Gregory Giuliani, Pierre Lacroix, Yaniss Guigoz, Lorenzo Bigagli, Nicolas Ray, Anthony Lehmann

Bringing GEOSS services into practice

Foreword

This material has been developed by researchers of the enviroSPACE lab. at the University of Geneva (http://www.unige.ch/envirospace), Laboratory of Earth and Space Science Informatics at the Institute of Atmospheric Pollution Research (IIA) of the National Research Council of Italy (CNR) (http://www.essi-lab.eu) and the GRID-Geneva office of the United Nations Environment Programme (http://www.grid.unep.ch).

This material has been supported by EU FP7 EnviroGRIDS (grant no. 226740), IASON (grant no. 603534) & EOPOWER (grant no. 603500) projects.

http://www.envirogrids.net http://www.iason-fp7.eu http://www.eopower.eu

Authors:

GIULIANI Gregory [gregory.giuliani@unige.ch], LACROIX Pierre [pierre.lacroix@unige.ch], GUIGOZ Yaniss [yaniss.guigoz@unepgrid.ch], BIGAGLI Lorenzo [lorenzo.bigagli@cnr.it], RAY Nicolas [nicolas.ray@unige.ch], LEHMANN Anthony [anthony.lehmann@unige.ch].

Citation: Giuliani G., Lacroix P., Guigoz Y., Bigagli L., Ray N., Lehmann A. 2014. Bringing GEOSS Services into Practice. GIS Open Source Workshop Material. University of Geneva, United Nations Environment Programme, National Research Council of Italy. 189 pages. Available at: http://www.geossintopractice.org

ISBN 978-2-8399-1380-5

Version: January 2014

"Bringing GEOSS services into practice" aims at teaching participants how to install, configure and deploy a set of open source software to publish and share data and metadata through the Global Earth Observation System of Systems (GEOSS) using OGC web services & ISO standards.

GEOSS has been created as an international voluntary effort that connects geospatial and Earth Observation and information infrastructures, acting as a gateway between producers of environmental data and end users. The aim of GEOSS is to enhance the relevance of Earth observations and to offer public access to comprehensive, near-real time data, information and analyses of the environment.

The teaching material of this workshop is composed of two components: an interactive ebook (also available in PDF format) and a Virtual Machine (an OVA file) that can be installed on any computer (Mac, Linux, Windows) using VirtualBox (https://www.virtualbox.org). The teaching material can be downloaded on the EOPOWER project website: http://www.eopower.eu. To open the virtual machine for the first time you need to proceed as follows:

- Right-click the OVA file
- Open it in Virtual Box
- (optional) Change the settings, e.g. the RAM
- Import

For more information, please refer to the Frequently Asked Questions (FAQ) page: http://www.unige.ch/sig/enseignements/GeossInPractice/FAQ.html

To connect to the Virtual Machine use the following credentials:

- Username: `geoss`
- Password: `geoss`

1

Introduction on SDI

Keywords: data sharing; Internet; interoperability; OGC; SDI; web services

What you will learn?

- What is a Spatial Data Infrastructure
- What is interoperability
- What is a web service

Recommended readings:

- SDI cookbook
- Interoperability and the Value of Standards
- OGC Reference Model

Responding to our changing environment, the need for data sharing

Today we are living in a globalized world where everything is changing rapidly (growing population, environmental deterioration,...) and where communication means have taken a remarkable place in our life. Everyday we access an enormous and continuous flow of information and much of it refer to a position or a specific place on the surface of our planet. This information is therefore, and by definition, georeferenced.

In the last 30 years, the amount of georeferenced data available has grown dramatically following the evolution of the communication means and due to the rapid development of spatial data capture technologies such as Global Positioning System (GPS), remote sensing images, sensors, etc (Philips et al., 1999). Over the last ten years, with the advent of applications like Google Earth, we have seen that geoinformation has been incorporated and routinely embedded into business and workflows of agencies at all levels of government, as well as in the private sector (Booz et al., 2005).

Despite the fact that administrations and governments are recognizing that spatial information is important and must be part of the basic information infrastructure that need to be efficiently coordinated and managed for the interest of all citizens (Ryttersgaard, 2001), this huge amount of geospatial data is stored in different places, by different organizations and the vast majority of the data are not being used as effectively as they should.

Moreover at the United Nations Conference on Environment and Development in Rio de Janeiro in 1992, the so-called Agenda 21 resolution fosters the importance of georeferenced information to support decision-making and management on the degradation and threats that are affecting the environment (GSDI, 2004). This means that there is a strong need for availability and access to appropriate information. The development of databases and exchange of information are the conditions for creating the basis for a sustainable development and to support the information management needs for implementing and monitoring sustainable development policies and goals like the UN Millennium Development Goals (UNGIWG, 2007).

Thus, geospatial information is a critical element underpinning decision making for many disciplines (Rajabifard and Williamson, 2001) and is indispensable to make sound decisions at all levels, from local to global. Experiences from developed countries show that more than two-third of human decision-making are affected by georeferenced information (Ryttersgaard, 2001).

However, geospatial information is an expensive resource, it is time consuming to produce, and for this reason it is of high importance to improve the access and availability of data, and

promote its reuse. Many of the decisions that different organization need to make depend on good and consistent georeferenced data, available and readily accessible (Rajabifard and Williamson, 2001).

In 1998, the former vice-president of the United States, Al Gore, presented its visionary concept of a Digital Earth (Gore, 1998), "a multi-resolution, three-dimensional representation of the planet, into which we can embed vast quantities of georeferenced data". As of today this vision is clearly not fully realized, but gives us an interesting support to our purpose as it is still actual.

Talking about geospatial data, Al Gore (1998) mentioned that the difficult part of taking advantage of this vast amount of information will be "making sense of it, turning raw data into understandable information" because at the moment we have more information than we can handle and it is stored in "electronic silos of data" , remaining mostly unused. He envisioned applications where "information can be seamlessly fused with the digital map or terrain data" allowing the user to move through space and time, but of course, to achieve this vision, a collaborative effort (from government, industry, academia and citizens) is needed.

All the technologies and capabilities required to realize this vision and to build a Digital Earth are already available:

- *Computational science:* even a simple desktop computer can process complex models and simulations. With the potential that technologies such as the Grid are offering new insights into the data are possible, giving us the ability to simulate phenomena that are impossible to observe.
- *Mass storage:* storing tera-bytes of data on a desktop computer is not a problem anymore.
- *Satellite imagery:* a lot of satellites are continuously observing the Earth offering high spatial and temporal observations.
- *Broadband networks:* are already a reality giving the ability to connect different databases together.
- *Interoperability:* this is a key point to allow communication and integration of distributed data, allowing the geospatial data generated by one software to be read by another.
- *Metadata:* are important as they describe the data, allowing a user to evaluate and discover the data before using them.

Even if all technologies are ready, organizations and agencies around the world are still spending billions of dollars every year to produce, manage and use geospatial data but without having the information they need to answer the challenges our world is facing (Rajabifard and Williamson, 2001). These authors also highlight the facts that most organizations and/or agencies need more data than they can afford, they often need data outside their jurisdictions, and the data collected by different organizations are often incompatible. This inevitably leads to inefficiencies and duplication of effort, and thus it is evident that countries can benefit both economically and environmen-

tally from a better management of their data (UNGIWG, 2007; GSDI, 2004).

In consequence, it is now essential to make these data easily available and accessible in order to give the opportunity to the user to turn them into understandable information with a clear and broad benefits for the society and the economy because “working together, we can help and solve many of the most pressing problems facing our society...” (Gore, 1998).

It is clear that there are a lot of challenges to face, both tangible and intangible, when we start sharing data but we have to overcome them in order to improve our knowledge, share our experiences and try to build a better informed society. Achieving the goal of a sustainable development requires the integration of a large number of different types of data from different sources. Through agreed common standards and a clear political will, these data can be interchanged and integrated in an interoperable way, leading to a new collaborative approach to decision-making.

In conclusion, for Arzberger et al. (2004), ensuring that data are easily accessible, so that they can be used as often and as widely as possible is a matter of sound stewardship of public resources. Availability should be restricted only in certain specific cases like national security. These authors argue that “publicly funded research data should be openly available to the maximum extent possible”, because publicly funded data are a public good, produced in the public interest.

Spatial Data Infrastructure

Definition, concepts and rationale

The term Spatial Data Infrastructure (SDI) if often used to describe the mechanisms or the enabling environment, that supports easy access to, and utilization of, geospatial data and information (UNECA, 2005). This definition is quite reductive as it gives the idea that SDIs are essentially technical. The primary objective of SDIs is to provide a basis for geospatial data discovery, evaluation, and application for users and providers within all levels of government, commercial and the non-profit sectors, academia and citizens (GSDI, 2004).

This means that SDIs are more than just data repositories. SDIs store data and their attributes, and their related documentation (metadata), offering a mean to discover, visualize, and evaluate their fitness to different purpose, and finally provide access to the data themselves. In addition to these basic services, there are often additional services or software supporting the use of the data. Finally, to make an SDI work efficiently, it is necessary to include all the organizational agreements needed to coordinate and administer it.

In consequence, following Masser (2005) and GSDI (2004), we can give a more complete definition of what are SDIs:

"A spatial data infrastructure supports ready access to geographic information. This is achieved through the co-ordinated actions of nations and organizations that promote awareness and implementation of complimentary policies, common standards and effective mechanisms for the development and availability of interoperable digital geographic data and technologies to support decision making at all scales for multiple purposes. These actions encompass the policies, organizational remits, data, technologies, standards, delivery mechanisms, and financial and human resources necessary to ensure that those working at the national and regional scale are not impeded in meeting their objectives".

Before going further in details, we have to explain the concepts underlying the rational of SDI, in particular geospatial data and information (also named geodata or georeferenced data). A geospatial data describes a location on Earth, giving through its attributes a comprehensive picture of the physical world both in term of spatial and/or temporal extent. Geospatial data are extremely valuable as users can build spatial relationships between features and data. For example, just after a flood event, one can overlay remote sensing images with existing georeferenced data of settlements to evaluate the extent of the damage and then focus humanitarian assistance. In consequence, geospatial data has a key role to play in our knowledge-based economy affecting directly or indirectly different sec-

tors like forestry, urban planning, security, telecommunication, environmental protection, (fig.1).

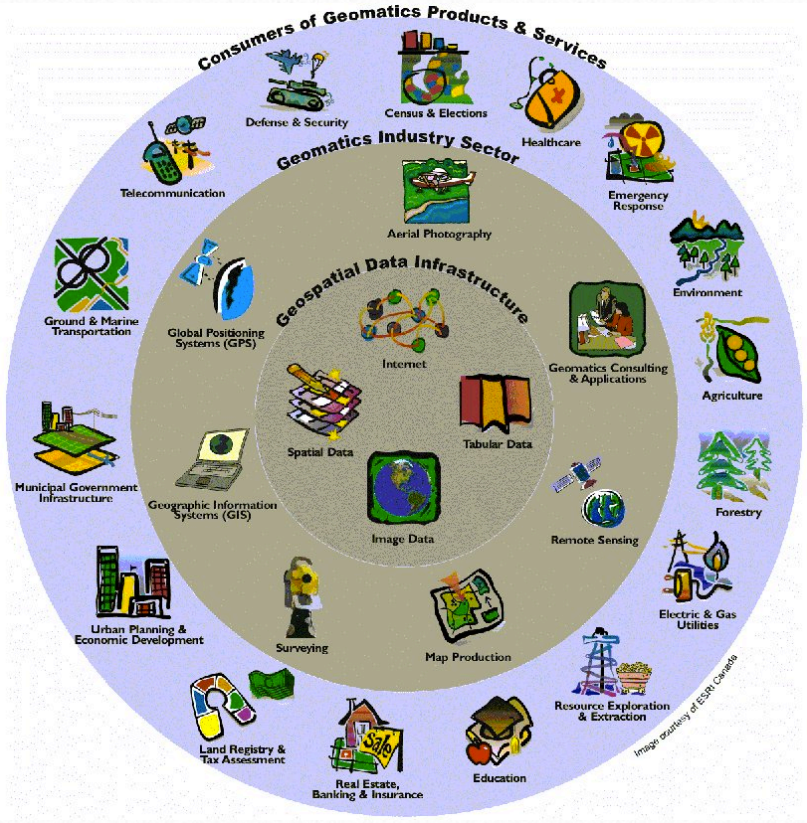


Fig.1: GIS and economy (Source: Geoconnections).

If, previously, geospatial information was mostly presented in the form of paper maps, with the increasing means to capture information in digital formats, geospatial data are nowadays used and viewed within a Geographical Information System (GIS). This computer system is capable of assembling, storing, manipulating, and displaying geographically referenced information (UNECA, 2005).

A GIS gives the ability to merge different existing information from different sources facilitating collaboration in creating and analyzing data. Due to these new possibilities of reusing existing data and working on collaboratively greater scale, new challenges arise. When someone wishes to create a new information layer based on different data sets or different formats, with different terminology, and perhaps different projection, it is quite difficult to bring them together. Harmonizing geospatial data is a complex, costly and time-consuming task, but could be achieved by agreeing among data capturers before the work begins.

The growing recognition that once a geospatial data set has been created it could be used for public and private sectors (Ryttersgaard, 2001), reinforces the need to store data into databases that are made accessible for different purposes (Philips et al., 1999). This leads to the concept that geospatial data could be a shared resource, which will be maintained continuously.

The advantage of having geospatial data in a digital form (UNECA, 2005; UNGIWG, 2007) are:

- Easy storage,
- Easy dissemination,
- Facilitation of data exchange/sharing,
- Faster and easier updates and corrections,
- Ability to integrate data from multiple sources,
- Customization of products and services.

As a result of the previous considerations, the concept of the SDI was developed in order to facilitate and coordinate the exchange and sharing of geospatial data (Rajabifard and Williamson, 2001), encompassing the data sources, systems, network linkages, standards and institutional issues involved in delivering geospatial data and information from many different sources to the widest possible group of potential users (Coleman et al., 1997).

The vision of an SDI incorporates different databases, ranging from the local to the national, into an integrated information highway and constitutes a framework, needed by a community, in order to make effective use of geospatial data (UNECA, 2005).

Objectives

Following Masser's definition (2005) and the different considerations highlighted in the previous section we can list different objectives underpinning SDIs:

1. The overall objective of an SDI is to maximize the reuse of geospatial data and information.

2. SDIs cannot be realized without coordination (especially by governments).

3. SDIs must be user driven, supporting decision-making for many different purposes.

4. SDIs implementation involves a wide range of activities, including not only technical topics such as data, standards, interoperability, and delivery mechanisms, but also institutional arrangements, policies, financial and human resources.

5. The term infrastructure is used to promote the idea of a reliable and supporting environment, analogous to a road or a telecommunication network, facilitating the access to geoinformation using a minimum set of common practices, protocols, and specifications (GSDI, 2004). This allows the movement of spatial information instead of goods.

6. SDIs are all about (UNGIWG, 2007):

7. Re-use: of data, technical capabilities, skills developed, invested effort and capital.

8. Sharing: “sharing-not-wearing” the costs of data, people, technology,... helping to realize more rapid returns on investment.

9. Learning from others: avoiding the pitfalls experienced by others.

10. Avoid duplication efforts and expenses and enables users to save resources, time and effort when trying to acquire or maintain datasets (Rajabifard and Williamson, 2001).

11. SDIs are “about working smarter, not harder” (UNGIWG, 2007).

12. Implies to scale from specific and monolithic (data-centric) towards independent and modular (service-oriented) information systems.

13. Integrate these systems together into an information highway which both links together environmental, socio-economic and institutional databases and provides a movement of information from local to national and global levels.

14. Encompass the sources, systems, network linkages, standards and institutional issues involved in delivering spatially-related information from many different sources to the widest possible group of potential users.

Altogether these objectives intend to create an environment that foster activities (fig.2) for using, managing and producing geospatial data and in which all stakeholders can co-operate with each other and interact with technology, to better achieve their objectives at different political/ institutional levels (Rajabifard and Williamson, 2004).

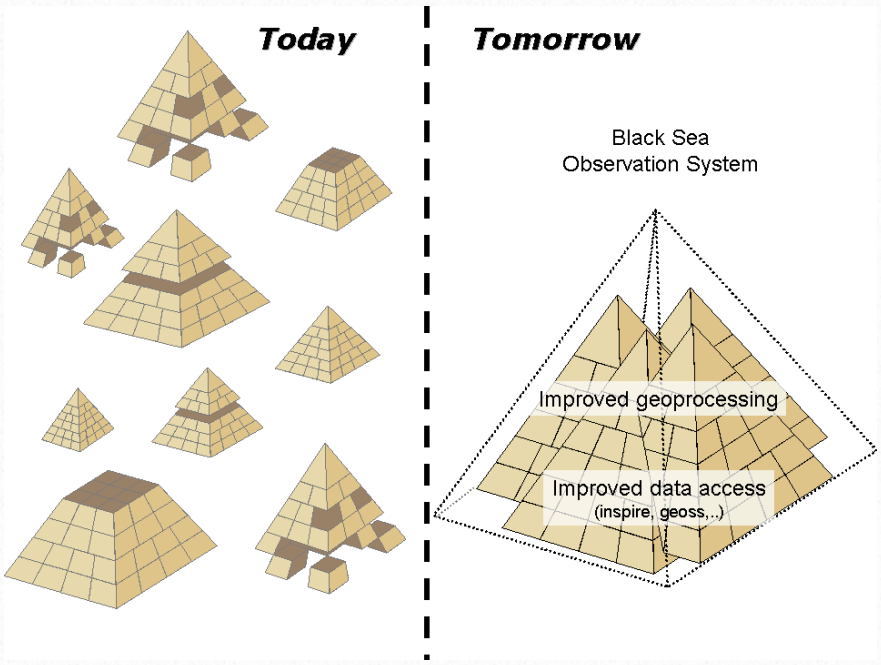


Figure 2: Integrated data and information

Components

Masser (2005) identifies the most important stakeholders with special interests in geoinformation/SDIs matters and shows the diversity both in terms of size and resource of the large numbers of players involved:

- Central government organizations,
- Local government organizations,
- Commercial sector,
- Non-for-profit/non-governmental organizations,
- Academics,
- Individuals.

Thus the temptation will be to create a centralized "one-size-fits-all" spatial database, in order to provide all the information needed by a country or a specific community of common interest. But as reported by UNECA (2005), UNGIWG (2007) and GSDI (2004) the existence of geospatial data and information does not alone ensure that it is used for decision-making. Different other factors are important to consider in order to ensure that information will be effectively used and reused:

- To be used, people need to know that the data exist, and where to obtain it.
- They need to be authorized to access and use the data.
- They need to know the history of the data capture, in order to interpret it correctly, trust it and be able to integrate it meaningfully with data coming from other sources.
- They need to know if the data depends on other data sets, in order to make sense of data.

Consequently, to leverage the full potential of geospatial data, an SDI must be made of different components to allow users to find, discover, evaluate, access and use these data, namely:

- A clearly defined core of spatial data.
- The adherence to known and accepted standards and procedures.
- Databases to store data and accessible documentation about the data, the so-called metadata.
- Policies and practices that promote the exchange and reuse of information.
- Adequate human and technical resource to collect, maintain, manipulate and distribute geospatial data.
- Good communication channels between people/organizations concerned with geospatial data, allowing the establishments of partnerships and share knowledge.

- The technology for acquiring and disseminating data through networks.
- Institutional arrangements to collaborate, cooperate and coordinate actions.

But as stated by Rajabifard and Williamson (2001), there is an important additional component represented by people. This include the users of geospatial data but also the providers and any other data custodians. For these authors, people are the key to transaction processing and decision-making. Facilitating the role of people and data in governance that appropriately supports decision-making and sustainable development objectives is central to the concept of SDI.

In order to meet the requirements of all stakeholders involved, an SDI must (Coleman et al., 1997):

- Be widely available,
- Be easy to use,
- Be flexible,
- Form the foundation for other activities.

In summary, Rajabifard and Williamson (2001) suggest that an SDI cannot be seen only as composed of geospatial data, services and users but instead involves other issues regarding interoperability, policies and networks.

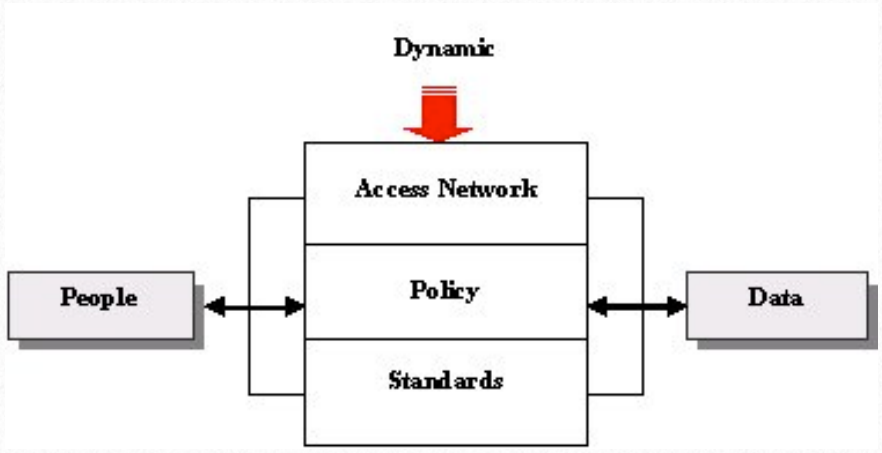


Figure 3: Nature and relations between SDI components (Source: GISCafe).

This shows that an SDI is by nature really dynamic as people who want to access data must interact with technological components (fig.3).

SDI hierarchy

As a result of the fact that SDI initiatives range from local to national and regional levels (Crompvoets, 2003; Masser 2007) and they all aim to promote economic development, to stimulate better government and to foster environmental sustainability (Masser, 2005), Rajabifard (2002) proposed a model of SDI hierarchy that is made of inter-connected SDIs developed at different levels (from local to global). Each SDI of a higher level is primarily formed by the integration of geospatial datasets developed and made available by the lower level (fig.4).

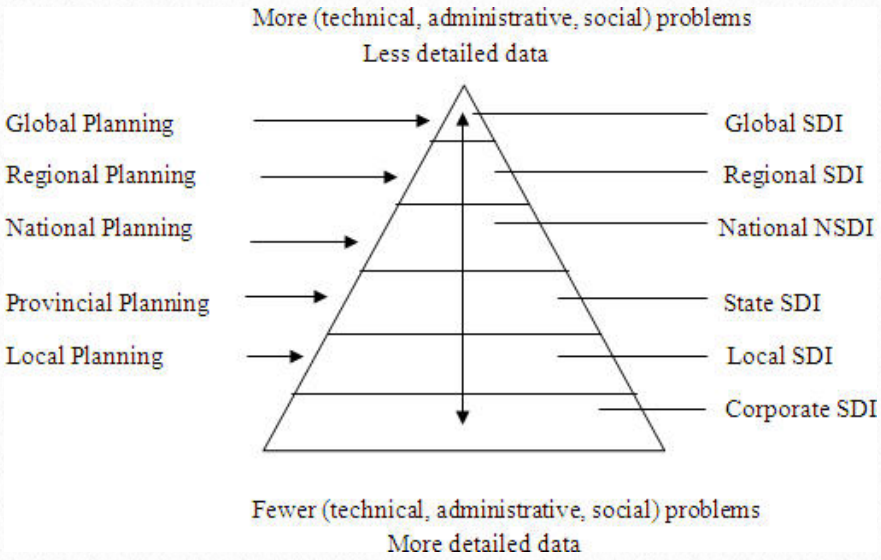


Figure 4: SDI hierarchy. (Source: GISCafe)

Such a hierarchy has two views: in one hand it is an umbrella in which the SDI at a higher level encompasses all the components of SDIs at levels below. On the other hand, it can be seen as the building block supporting the access of geospatial data needed by SDIs at higher levels. The SDI hierarchy allows to create an environment in which users working at any level can rely on data from other levels and integrate geospatial data from different sources (Mohammadi et al. 2007). Such a hierarchy is envisioned by regional and global initiatives like INSPIRE and GEOSS that will be further discussed.

For Masser (2006), the SDI hierarchy poses the challenge of a multistakeholder participation in SDI implementation because the bottom-up vision differs a lot from the top-down approach that is implicit in most of the SDI literature. The top-down approach emphasizes the need for standardization and uniformity while the bottom-up stresses the importance of diversity and heterogeneity due to the different aspirations of the various stakeholders. In consequence, it is necessary to find a consensus to ensure some measure of standardization and uniformity while recognizing the diversity and the heterogeneity of the different stakeholders performing different tasks at different levels.

SDI evolution and (emerging) trends

Different authors (Masser, 2005; Crompvoets, 2003) have studied the diffusion and evolution of SDI around the world and show that driving forces behind SDI initiatives are generally similar:

- Promoting economic development,
- Stimulating better government,
- Fostering environmental sustainability,
- Modernization of government,
- Environmental management.

They all agree on the fact that, as of today, a critical mass of SDI users has been reached as a result of the diffusion of SDI concepts during the last ten to fifteen years. This provides a basic network of people and organizations that is essential for future development of SDIs.

Rajabifard et al. (2001) find that the first generation of SDIs, based on a product model, gave away to a second generation at the beginning of the years 2000, the latest being characterized by a process model. Indeed, the first generation of SDIs were product-based, aiming to link existing and future databases while the second generation aim to define a framework to facilitate the management of information assets allowing reuse of collected data by a wide range of people and/or organizations for a great diversity of purposes at various times. For Masser (2005) this evolution emphasis the shift from the concerns

of data producers to those of data users and the shift from centralized structures to decentralized and distributed networks like the web.

The process-based model emphasizes the communication channel of knowledge infrastructure and capacity building towards the creation of an SDI facilitating cooperation and exchange of data and knowledge (Rajabifard and Williamson, 2001). They also highlight the fact that the characteristics of the social system strongly influence the approach taken to implement and develop a Spatial Data Infrastructure. They propose key issues and strategies to be considered for the design process:

- Development of a strategic vision and associated implementation strategy,
- Recognition that SDI is not an end in itself,
- Key institutional strategy is to have all coordinating processes administered by one group.

Today's effort on the technical development of SDI components clearly focus on the exchange of geospatial data in an interoperable way (Bernard and Craglia, 2005) through services that allow efficient access to spatially distributed data. The shift towards an infrastructure offering services to answer requests rather than a “simple” network allowing to find, view and exchange geospatial data is highlighted by the concept of web services and the related Service Oriented Architecture (SOA).

Web services are a “new paradigm” (Comert, 2004) where different systems or providers offer some services for certain user groups, allowing an easy access to distributed geographic data and geoprocessing applications. The web services emphasize the necessity that systems involved could talk to each other and the provision of this talk should be easy and cost-effective for businesses to profit. In other words, web services relay on interoperability.

Web services enable the possibility to construct web-based application using any platform, object model and programming language. A service is no more than a collection of operations that allows users to invoke a service, which could be as simple as requesting to create a map or complicated as processing a remote sensing image.

In summary, web services are for application-to-application communication over internet and are based, in general, on open standards like XML (Comert, 2004). SOA is the basic principle concept supporting web services development. It promotes loose coupling between software components so that they can be reused (Sahin and Gumusay, 2008). In a SOA, the key component is services. They are well defined set of actions, self contained, stateless, and does not depend on the state of other services.

There are three components on the web services architecture: service provider, service requester and service broker and three operations: publish, find and bind. A SOA relates the three components to the three operations to allow automatic discovery and use of services.

In a traditional scenario, a service provider hosts a web service and “publishes” a service description to a service broker. The service requester uses a “find” operation to retrieve the service de-

scription and uses it to “bind” with the service provider and invoke the web service itself (fig.5).

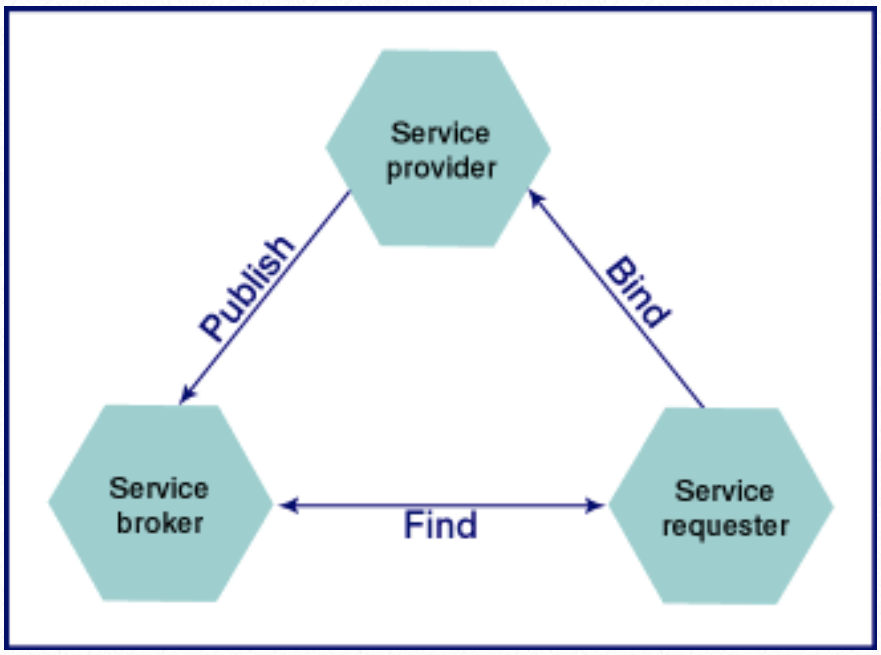


Figure 5: Basic operations in SOA (Source: IBM).

SOA is the underlying concept for an interoperable environment based on reusability and standardized components and thus is of high importance for SDIs allowing applications and related components to exchange data, share tasks, and automate processes over the Internet (OGC, 2004).

The OGC web services are, by far, the most important and relevant web services for our purpose and they will be discussed in details in upcoming sections of this tutorial.

With the advent of web services into the SDI community new trends/opportunities could be foreseen:

- Actual SDIs are lacking of analysis capabilities, an essential task to turn data into understandable information. This means that processing geospatial data is done in general on desktop computers and thus limit the analytical capacities caused by the huge execution time that geoprocessing tasks require to process a vast amount of data. With the recently introduced Web Processing Service and the promises of high storage and computing capacities offered by Grid infrastructures, new opportunities are emerging within geosciences communities (Padberg and Greve, 2009)

- Semantic web developed vocabularies (called ontologies) for geospatial data with the goal to increase understanding of such data by machines, allowing automated process through web services (Boes and Pavlova, 2008; Vaccari et al., 2009).

Web services are one aspect of the Web 2.0 revolution. The web 2.0 refers to a second generation of internet based services, that allow people to collaborate and share information (Boes and Pavlova, 2007). GIS is also taking advantage of the web 2.0 revolution, with a good example being Google Maps that opened some of the more straightforward capabilities of GIS to the general public (Goodchild, 2007) and allowing, with other tools, the general public to create and generate news sources of data and information. This phenomenon is also know as Volunteered Geographic Information (VGI) (Boes and Pavlova, 2007; Coleman et al., 2009, Craglia, 2007). VGI offers new opportunities and perhaps will influence the development of SDIs and the production of data in a near future.

Finally, it is necessary to mention that building an efficient SDI is almost impossible without partnership because a single agency is unlikely to have all resources, skills or knowledge to undertake the development of all aspects of SDI (UNECA, 2005; UNGIWG, 2007). This is why different authors (Williamson et al., 2003, Rajabifard and Williamson, 2004) stress the importance of the capacity building component in the SDI implementation process.

Benefits

To conclude, we can highlight some of the (expected) benefits that SDIs can offer:

- Universal (anywhere and anytime) access to geospatial data and related information,
- Services and applications to discover and access distributed data sources,
- Integration of different geospatial information to provide seamless visualization,
- Seamless combination (chaining) of data, services and related applications,
- Geospatial data update and maintenance made easy,
- Sharing and reuse capabilities,
- Collaborative activities,
- Wide-scale interoperability, agreeing on open and common standards,
- Development of partnerships, collaboration between different stakeholders.

The Canadian Geospatial Data Infrastructure (CGDI) claimed that developing applications using such an infrastructure allows to:

- Reduce costs: Applications can be built by reusing existing services.
- Reduce complexity: Service interfaces hide the underlying complexity.
- Permits less costly integration and interoperability: Standard interfaces simplify interconnection and integration.
- Allow direct access to current, authoritative source data.

Finally an effective and working SDI leads to:

- Inform decision-making: easy access to current information, knowledge and expertise.
- Increase efficiency: standards and specifications, as well as access to services, reduce duplication of effort.
- Enhance usability: providing reliable access to geospatial information to all levels, from the citizens to governments.
- Push for economic growth.

We would not depict SDIs as a “perfect tool” that can solve all problems. It is evident that SDIs represent a great opportunity and a framework with great perspective but as Masser reminds us (2005), SDIs can facilitate access to data to a wide range of users only if profound changes in “sharing spirit” take place. He also mentions the fact that building an SDI is a long term process. In order to be fully operational such a process depends on sustainability and commitment.

In addition, there are several others issues that could limit the implementation of SDI concepts such as: collaboration, funding, political stability, legislation, priorities, awareness, copyrights, privacy, licensing, capacity building and cultural issues (Table 1).

Technical	**Institutional**	**Policy**	**Legal**	**Social**
Computational heterogeneity	Collaboration models	Political stability	Rights, Restrictions	Cultural
Semantic	Funding model	Legislation	Copyright,	Capacity building
Reference sys.	Linkage between data units	Priorities/ Sustainable dev.	Intellectual Property rights	Equity

Table 1: Integration issues (Williamson et al., 2006)

To conclude this section, it is important to keep in mind that data sharing and related SDI developments rely first on individuals (Craig, 2005) that have many interests in common. First, their idealism, their sense that better data will lead to better decisions. Second, their self-interest: by sharing, they got something in return even if it is intangible, they are viewed as cooperative partners, and finally they are involved in a professional culture that honors serving society and cooperating with others.

Interoperability and standards

Definitions and concepts

Previously, we have seen that we are living in a world that is changing rapidly with communication means that are taking an increasing place in our everyday life. This communication revolution is mostly based on the Internet, whose successes are due to interoperability. Interoperability is “the ability of a system or a product to work with other systems or products without special effort on the part of the customer.” (OGC, 2004). This means that two or more systems or components are able to transmit or exchange information through a common system and to use the information that has been exchanged.

When systems are interoperable, it gives the user the ability to:

- Find what he needs,
- Access it,
- Understand and employ it,
- Have goods and services responsive to his need.

As of today, in a climate of economic constraint, interoperability and standardization have never been so important because a non-interoperable system impedes the sharing of data, information and computing resources (OGC, 2004), leading organizations to spend much more than necessary on data, software and hardware. Moreover being non-interoperable increases the risk for a system or an infrastructure to not deliver its expected benefit and in consequence to lead to a user disappointment and system failure.

In order to achieve interoperability, there are two approaches:

- Adhering to standards
- Making use of a "broker" of services that can convert one product's interface into another product's interface, "on the fly".

One good example of the first approach is the Web, where standards like HTTP, TCP/IP or HTML have been developed by organizations that wish to create standards to “meet everyone's needs without favoring any single company or organization” (OGC, 2005).

The great advantage of interoperability, and that is why it is an essential building block for the GIS and SDI industry, is that it describes the ability of locally managing and distributing heterogeneous systems to exchange data and information in real time to provide a service (OGC, 2004). This allows the users to maximize the value of past and future investments in geoprocessing systems and data.

As a response to the need of GIS standards to support interoperability, the OGC aims to tackle the non-interoperability caused by the diversity

of systems creating, storing, retrieving, processing and displaying geospatial data in different formats. In addition to this, software vendors often did not communicate among themselves to agree on how data should be structured and stored and how systems must exchange information, leading inevitably to a non-interoperable environment, isolating geospatial data in "electronic silos" and resulting in expensive duplication of data and difficulty in sharing and integrating information (OGC, 2004).

The OGC (2005) has pointed out the general user needs:

- Need to share and reuse data in order to decrease costs (avoid redundancy collection), obtain additional or better information, and increase the value of data holdings.
- Need to choose the best tool for the job and the related need to reduce technology and procurement risks (avoid being locked in to one vendor).
- Need to leverage investment in software and data, enabling more people to benefit from using geospatial data across applications without the need for additional training.

The OGC believes that responding to the users needs of interoperability will have a profound and positive impact in the public and private sectors, opening the doors of new business opportunities and new human activities.

In summary, interoperability enhances: communication, efficiency and quality for the benefit of all citizens allowing them to access data in a good, consistent and transparent way.

Types of interoperability

There are two types of interoperability (OGC, 2004):

- *Syntactic (or technical)*: when two or more systems are capable of communicating and exchanging data, they are exhibiting syntactic interoperability. Specified data formats and communication protocols are fundamental. In general, XML or SQL standards provide syntactic interoperability. Syntactical interoperability is required for any attempts of further interoperability.
- *Semantic:* Beyond the ability of two or more computer systems to exchange information, semantic interoperability is the ability to automatically interpret the information exchanged meaningfully and accurately in order to produce useful results as defined by the end users of both systems. To achieve semantic interoperability, both sides must defer to a common information exchange reference model. The content of the information exchange requests are unambiguously defined: what is sent is the same as what is understood (eg. explaining why INSPIRE is producing data specifications).

Different types of geoprocessing systems (vector, raster, CAD, etc.) producing different types of data, different vendors geoprocessing systems using internal data formats and producing proprietary formats, different vendors systems using proprietary libraries and interfaces and reducing the possibilities of communication between systems... are all causes of syntactic non-interoperability while different data producers using different metadata schemas and/or different naming convention lead to semantic non-interoperability.

The World Wide Web and its associated technologies offer a great opportunity to overcome both syntactic and semantic non-interoperability because it is an almost universal platform for dis-

tributed computing and it provides facilities to semantically process structured text. The web is thus a key enabler for interoperability, by increasing access to geospatial data and processing resources, which in consequence increases the value of those resources (OGC, 2004).

To ensure effective interoperability, it is not only a matter of technology but also and often it requires a change of philosophy, of spirit to go "open". This is classified under human or legal/policy on the following table summarizing the different types of interoperability (Table 2).

Technical	**Semantic**	**Human**	**Legal/Policy**
Machine to machine connections	Common understanding concepts, terms, ...	Cooperation	Digital rights, ownerships
Software interaction	Inter-disciplinary vocabularies	Training	Responsibility
APIs			
Formats			

Table 2: Different types of interoperability.

As expressed by the OGC (2004) *"if an organization does not fully embrace the tenets of interoperability and interoperable architectures, then long-term success in integrating geospatial processes into an organization's overall business processes may be problematic"*.

In consequence, organizations will need:

- Commitment to interoperability and geospatial standards,
- Commitment to collaboration,
- Commitment to define a geospatial interoperability and information framework that meets the requirements of the organization,
- Commitment to the collection and maintenance of geospatial metadata,
- Commitment to training and education.

Through all these commitments, an organization will be truly interoperable, maximizing the value and reuse of data and information under its control and will be able to exchange these data and information with other interoperable systems, allowing new knowledge to emerge from relationships that were not envisioned previously.

Interoperability enablers

To enable effective interoperability, we have already seen that the Internet and standards are probably the most important components at a technical level but here are a lot of other possible enablers, both human and technical, that could help an organization to promote its commitment to interoperability:

- Web and networks,
- Standards,
- Infrastructure,
- Metadata,
- Support for multiple: languages, views, data formats, projections, datums,...
- Sharing of best practices,
- Cooperation and collaboration,
- Business models,

- Business agreements,
- Policy framework,
- Copy and access rights,
- Authorization,
- Others.

Altogether they will contribute in a way or another to a successful implementation of the geospatial interoperability by reaching a consensus between the users' need for compatibility with the autonomy and heterogeneity of the interoperating systems (OGC, 2005).

Standards

Standards are documented agreements, used in public contracts or international trade, containing technical specifications or other precise criteria to used consistently as rules, guidelines, or definitions of characteristics, to ensure that materials, products, processes and services are fit for their purpose (Ostensen, 2001). In other words, standardization means agreeing on a common system (OGC, 2005).

The existence of non-harmonized standards for similar technologies contribute to the so-called “technical barriers to trade”, avoiding a user to share data, information or services.

Although developing standards is a long and complex process, involving many organizations, based on a consultative approach and aiming to find a consensus between all the parties involved (UNECA, 2005), organizations and agencies are increasingly recognizing that standards are essential for improving productivity, market competitiveness, export capabilities (GSDI, 2004), and lowering maintenance and operation costs over time (Booz et al., 2005; Craglia and Nowak, 2006; Almirall et al., 2008).

We can summarize the functions of standards as follow:

- help to ensure interoperability,
- promote innovation, competition, commerce and free trade,
- increase efficiency,
- make things work,
- affect every aspect of our life (widespread use of standards).

Benefits

After discussing what are standards and interoperability, we can give an overview of the (expected) benefits of a truly interoperable architecture.

1. Allow sharing and reusing of data: gives access to distributed and heterogenous sources of data.
2. Avoid data duplication: data are collected and maintained at the most appropriate place.
3. Reduce the costs: of maintenance, of operations and of course of production.
4. Integration: As Mohammadi et al. (2007) shows multi-source data integration could only be achieved with an effective interoperability.
5. Reduce the complexity: through common knowledge, standards offer a set of rules

that every data provider can follow, understand and become familiar with. Moreover when a user shares a data in a standardized way, another will be immediately able to use it.

6. Increase efficiency.

7. Vendor neutral: avoid the fact to be locked in to one vendor.

8. Improve decision-making: offering standardized access to a vast amount of data and information and to used them as effectively as they should.

9. New opportunities and knowledge: open the doors to new activities and relations that were not foreseen before.

Finally, as OGC (2005) stated: *"Changing internal systems and practices to make them interoperable is a far from simple task. But the benefits for the organization and for those who make use of information it publishes are incalculable"*.

2

How to store data?

Keywords: DBMS; PostGIS; PostgreSQL; SQL

What you will learn:

- Create a PostgreSQL/PostGIS database
- Load vector data in a geospatial database

Recommended readings:

- http://www.postgis.us
- PostGIS documentation

What is a spatial database?

PostGIS is a spatial database. Oracle Spatial and SQL Server 2008 are also spatial databases. But what does that mean; what is it that makes an ordinary database a spatial database?

The short answer, is. . .

Spatial databases store and manipulate spatial objects like any other object in the database.

The following briefly covers the evolution of spatial databases, and then reviews three aspects that associate spatial data with a database – data types, indexes, and functions.

1. Spatial data types refer to shapes such as point, line, and polygon
2. Multi-dimensional spatial indexing is used for efficient processing of spatial operations;
3. Spatial functions, posed in SQL, are for querying of spatial properties and relationships.

Combined, spatial data types, indexes, and functions provide a flexible structure for optimized performance and analysis.

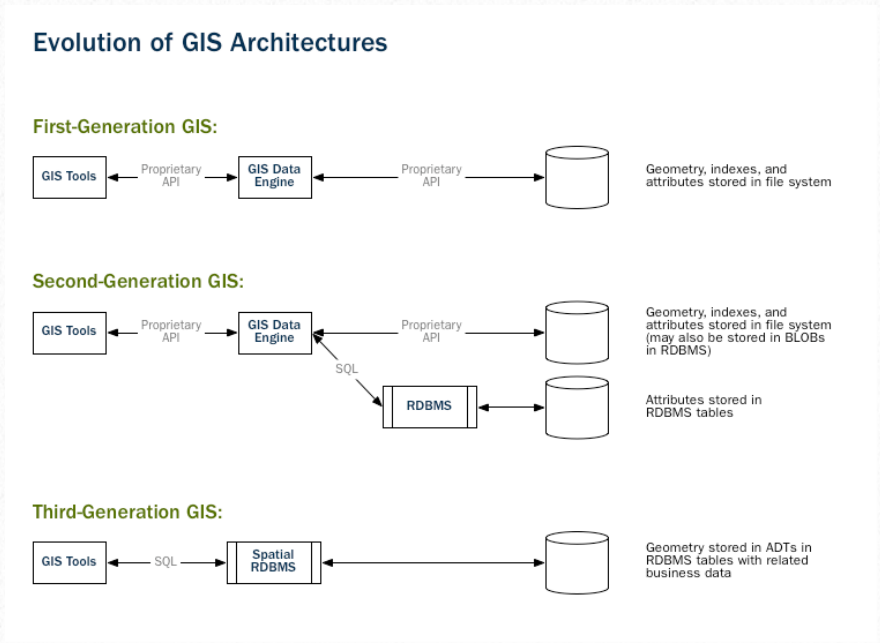
In the Beginning

In legacy first-generation GIS implementations, all spatial data is stored in flat files and special GIS software is required to interpret and manipulate the data. These first-generation management systems are designed to meet the needs of users where all required data is within the user's organizational domain. They are proprietary, self-contained systems specifically built for handling spatial data.

Second-generation spatial systems store some data in relational databases (usually the "attribute" or non-spatial parts) but still lack the flexibility afforded with direct integration.

True spatial databases were born when people started to treat spatial features as first class database objects.

Spatial databases fully integrate spatial data with an object relational database. The orientation changes from GIS-centric to database-centric.

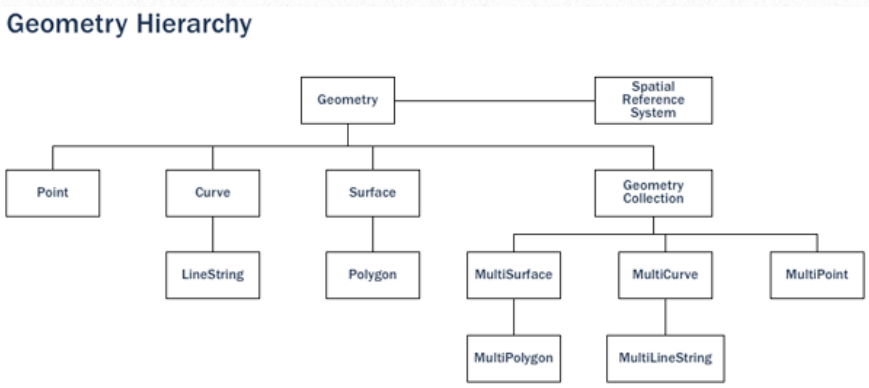


Note: A spatial database management system may be used in applications besides the geographic world. Spatial databases are used to manage data related to the anatomy of the human body, large-

scale integrated circuits, molecular structures, and electro-magnetic fields, among others.

Spatial Data Types

An ordinary database has strings, numbers, and dates. A spatial database adds additional (spatial) types for representing geographic features. These spatial data types abstract and encapsulate spatial structures such as boundary and dimension. In many respects, spatial data types can be understood simply as shapes.



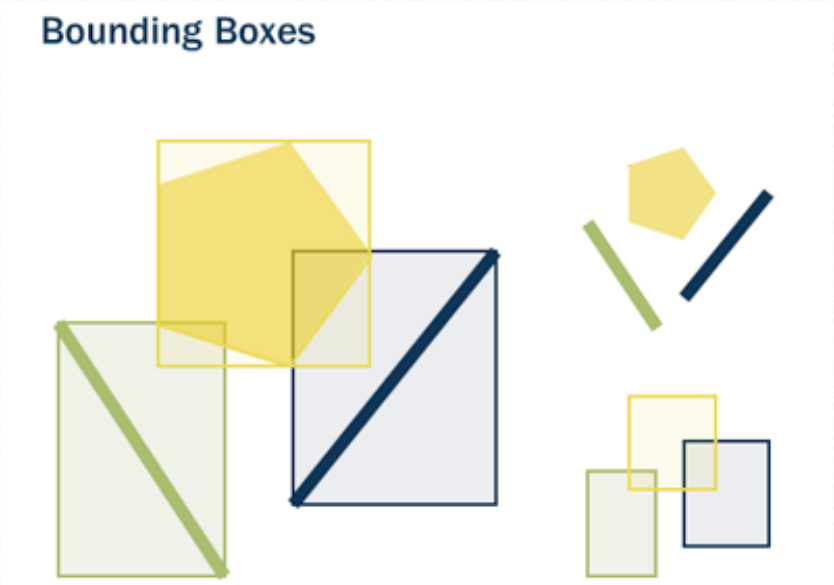
Spatial data types are organized in a type hierarchy. Each sub-type inherits the structure (attributes) and the behavior (methods or functions) of its super-type.

Spatial Indexes and Bounding Boxes

An ordinary database provides “access methods” – commonly known as indexes – to allow for fast and random access to subsets of data. Indexing for standard types (numbers, strings, dates) is usually done with B-tree indexes. A B-tree partitions the data using the natural sort order to put the data into a hierarchical tree.

The natural sort order of numbers, strings, and dates is simple to determine – every value is less than, greater than or equal to every other value. But because polygons can overlap, can be contained in one another, and are arrayed in a two-dimensional (or more) space, a B-tree cannot be used to efficiently index them. Real spatial databases provide a “spatial index” that instead answers the question “which objects are within this particular bounding box?”.

A bounding box is the smallest size rectangle capable of containing a given feature.



Bounding boxes are used because answering the question “is A inside B?” is very computationally intensive for polygons but very fast in the case of rectangles. Even the most complex polygons and linestrings can be represented by a simple bounding box.

Indexes have to perform quickly in order to be useful. So instead of providing exact results, as B-trees do, spatial indexes provide approximate results. The question “what lines are inside this polygon?” will be instead interpreted by a spatial index as “what lines have bounding boxes that are contained inside this polygon’s bounding box?”

The actual spatial indexes implemented by various databases vary widely. The most common implementation is the R-tree (used in PostGIS), but there are also implementations of Quad-

trees, and grid-based indexes in shipping spatial databases.

Spatial Functions

For manipulating data during a query, an ordinary database provides functions such as concatenating strings, performing hash operations on strings, doing mathematics on numbers, and extracting information from dates. A spatial database provides a complete set of functions for analyzing geometric components, determining spatial relationships, and manipulating geometries. These spatial functions serve as the building block for any spatial project.

The majority of all spatial functions can be grouped into one of the following five categories:

- *Conversion*: Functions that convert between geometries and external data formats.
- *Management*: Functions that manage information about spatial tables and PostGIS administration.
- *Retrieval*: Functions that retrieve properties and measurements of a Geometry.
- *Comparison*: Functions that compare two geometries with respect to their spatial relation.
- *Generation*: Functions that generate new geometries from others.

The list of possible functions is very large, but a common set of functions is defined by the OGC SFSQL and implemented (along with additional useful functions) by PostGIS.

What is PostGIS?

PostGIS turns the PostgreSQL Database Management System into a spatial database by adding adding support for the three features: spatial types, indexes, and functions. Because it is built on PostgreSQL, PostGIS automatically inherits important “enterprise” features as well as open standards for implementation

But what is PostgreSQL?

PostgreSQL is a powerful, object-relational database management system (ORDBMS). It is released under a BSD-style license and is thus free and open source software. As with many other open source programs, PostgreSQL is not controlled by any single company, but has a global community of developers and companies to develop it.

PostgreSQL was designed from the very start with type extension in mind – the ability to add new data types, functions and access methods at run-time. Because of this, the PostGIS extension can be developed by a separate development team, yet still integrate very tightly into the core PostgreSQL database.

Why choose PostgreSQL?

A common question from people familiar with open source databases is, “Why wasn’t PostGIS built on MySQL?”.

PostgreSQL has:

- Proven reliability and transactional integrity by default (ACID),
- Careful support for SQL standards (full SQL92),
- Pluggable type extension and function extension,
- Community-oriented development model,

- No limit on column sizes (“TOAST”able tuples) to support big GIS objects,
- Generic index structure (GiST) to allow R-Tree index,
- Easy to add custom functions.

Combined, PostgreSQL provides a very easy development path to add new spatial types. In the proprietary world, only Illustra (now Informix Universal Server) allows such easy extension. This is no coincidence; Illustra is a proprietary reworking of the original PostgreSQL code base from the 1980’s.

Because the development path for adding types to PostgreSQL was so straightforward, it made sense to start there. When MySQL released basic spatial types in version 4.1, the PostGIS team took a look at their code, and the exercise reinforced the original decision to use PostgreSQL. Because MySQL spatial objects had to be hacked on top of the string type as a special case, the MySQL code was spread over the entire code base. Development of PostGIS 0.1 took under a month. Doing a “MyGIS” 0.1 would have taken a lot longer, and as such, might never have seen the light of day.

Why not Shapefiles?

The shapefile (and other file formats) have been the standard way of storing and interacting with spatial data since GIS software was first written. However, these “flat” files have the following disadvantages:

- Files require special software to read and write. SQL is an abstraction for random data access and analysis. Without that abstraction, you will need to write all the access and analysis code yourself.
- Concurrent users can cause corruption. While it’s possible to write extra code to ensure that multiple writes to the same file do not corrupt the data, by the time you have solved the problem and also solved the associated performance problem, you will have written the better part of a database system. Why not just use a standard database?
- Complicated questions require complicated software to answer. Complicated and interesting questions (spatial joins, aggregations, etc) that are expressible in one line of SQL in the database take hundreds of lines of specialized code to answer when programming against files.

Most users of PostGIS are setting up systems where multiple applications will be expected to access the data, so having a standard SQL access method simplifies deployment and development. Some users are working with large data sets; with files, they might be segmented into multiple files, but in a database they can be stored as a single large table.

In summation, the combination of support for multiple users, complex ad hoc queries, and performance on large data sets are what sets spatial databases apart from file-based systems.

Note: PostGIS supports also raster file since version 2.0. More information on raster data handling in PostGIS can be found at: http://postgis.net/docs/RT_reference.html

Creating a spatial database

In this exercise you will create a PostGIS database and import vector data into this database. These data will be published on GeoServer and documented with ISO metadata later in this course.

Starting up PostGIS

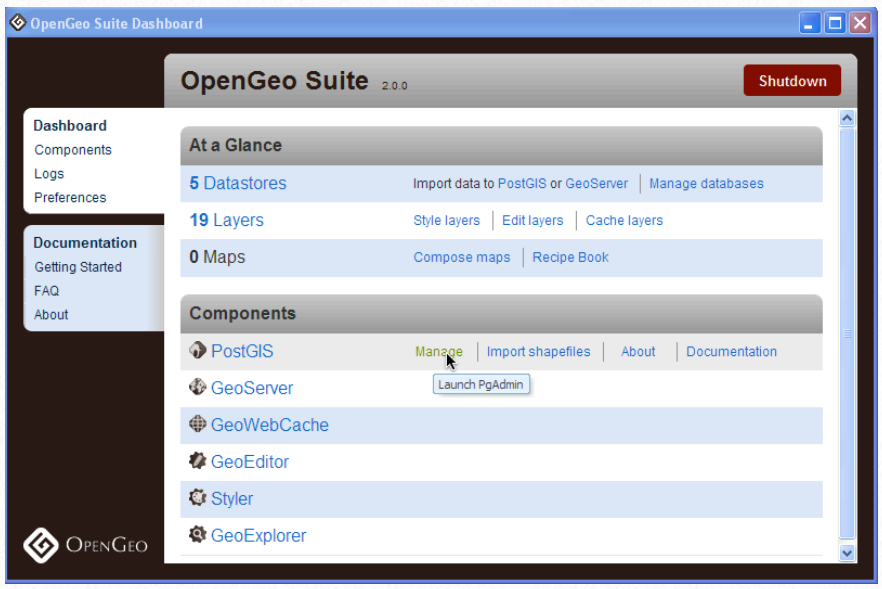
First, we need to start up PostGIS.

- Click **Applications menu** at the upper left of your virtual machine:

- Select **Development > pgAdminIII**.

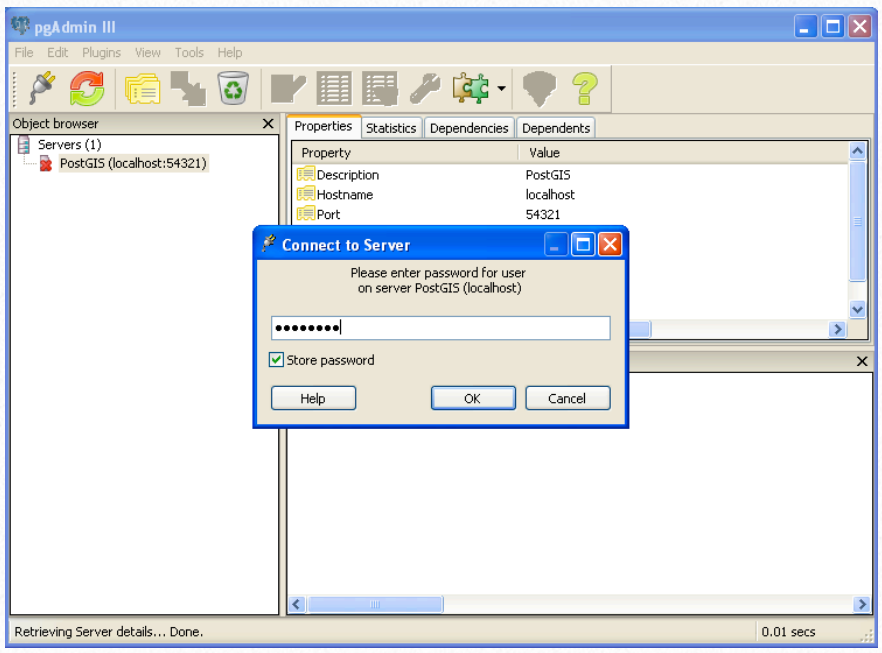
`pgAdmin` is the PostgreSQL administration interface which allows you many operations for managing data, e.g. importing shapefiles into PostGIS.

Note: PostgreSQL has a number of administrative front-ends. The primary is `psql` a command-line tool for entering SQL queries. Another popular PostgreSQL front-end is the free and open source graphical tool `dasboard` (accessible from the `start.html` file on the desktop of your virtual machine) which also lets you access other softwares such as GeoServer and GeoExplorer, which you will discover later in this course.



If this is the first time you have run pgAdmin, you should have a server entry for PostGIS (`localhost:5432`) already configured in `pgAdmin`.

- **Double click** the **localhost:5432** entry, and enter **opengeo** as password to connect to the database.



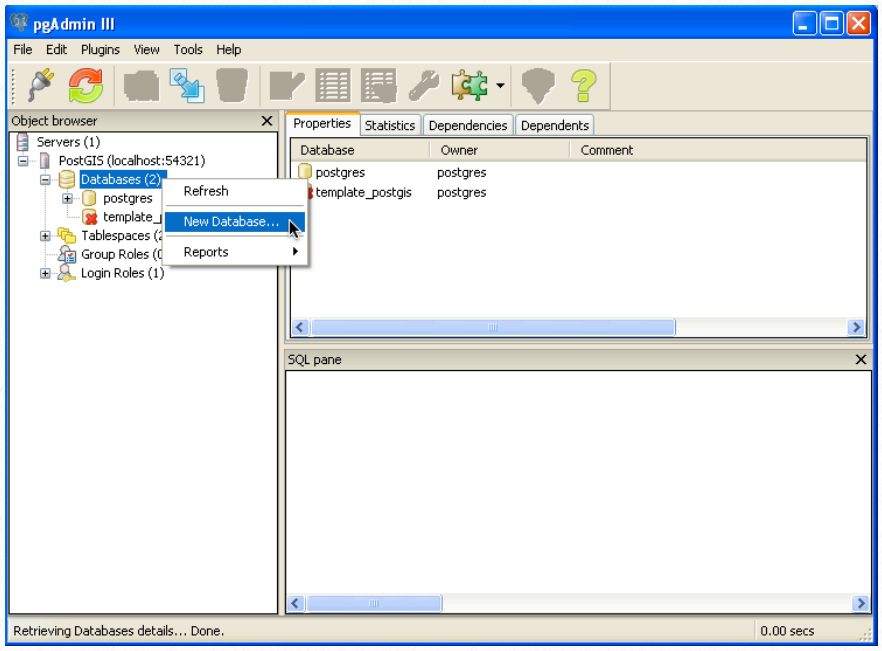
Note: If you have a previous installation of PgAdmin on your computer, you might not have another entry for `localhost:5432`. You will need to create a new connection. Go to `File > Add Server`, and register a new server at localhost and port `54321` (note the non-standard port number) in order to connect to the PostGIS bundled with the OpenGeo Suite.

Creating a Database

PostgreSQL has the notion of a template database that can be used to initialize a new database. The new database automatically gets a copy of everything from the template. When you installed PostGIS, a spatially enabled database called `template_postgis` was created. If we use `template_postgis` as a template when creating our new database, the new database will be spatially enabled.

- Open the **Databases** tree item and have a look at the available databases: the `postgres` database is the user database for the default `postgres` user and is not too interesting to us. The `template_postgis` database is the one that we are going to use to create spatial databases.

- **Right-click** on the **Databases** item and select **New Database**.

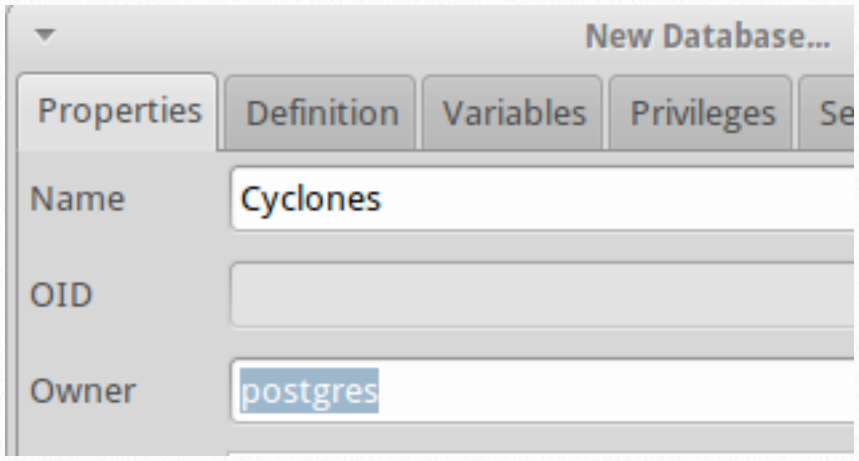


Note: If you receive an error indicating that the source database (`template_postgis`) is being accessed by other users, this is likely because you still have it selected. In this case you will need (1) to right-click on the `localhost:5432` item, (2) to select `Disconnect`, `(3)` `to` double-click the same item to reconnect.

- **Fill** in the `New Database` form **as shown below**:

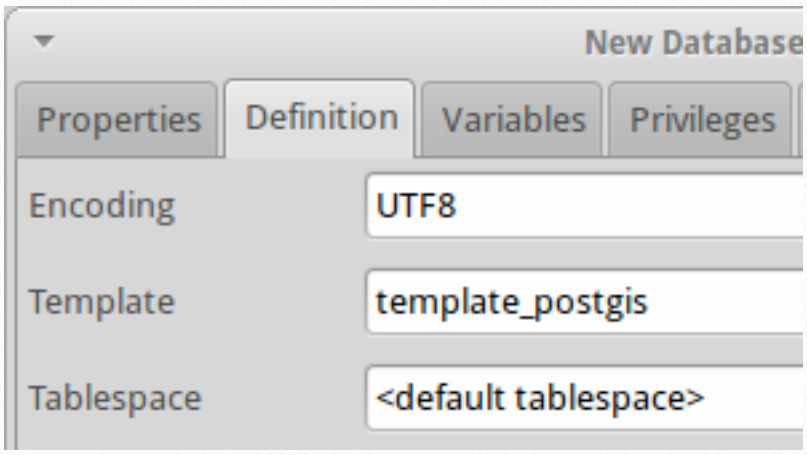
Name: **Cyclones**

Owner: **postgres**



Encoding: **UTF8**

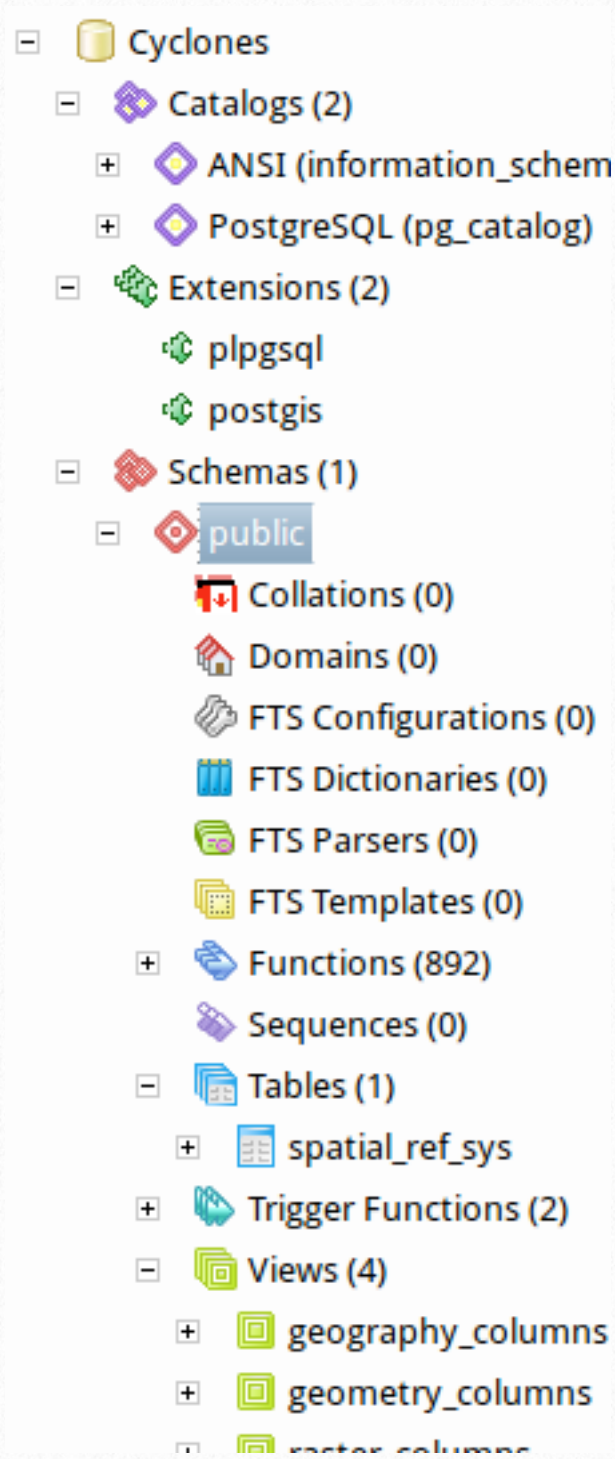
Template: **template_postgis**



- Click **OK**.

- **Select** the new **Cyclones** database and **open** it up to display the tree of objects.

You will see the public schema, and under it a couple of PostGIS-specific metadata tables such as `spatial_ref_sys`.

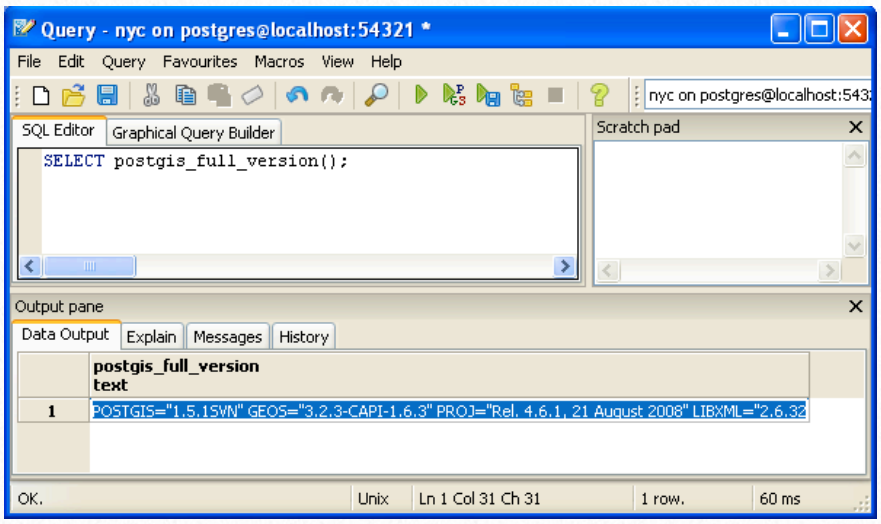


- Click on the **SQL query button** indicated below (or go to Tools > Query Tool).

- **Enter** the following query into the query text field:
  SELECT postgis_full_version();

Note: This is your first SQL query. `postgis_full_version()` is a management function that returns version and build configuration.

- Click the Play button in the toolbar (or press F5) to "**Execute the query**" The query will return the following string, confirming that PostGIS is properly enabled in the database.



Congratulation! You have successfully created a PostGIS spatial database!!

- You can now **close the Query window**. When prompted **do not save**.

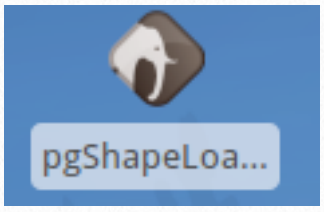
`pgAdmin` should still be open.

Loading spatial data

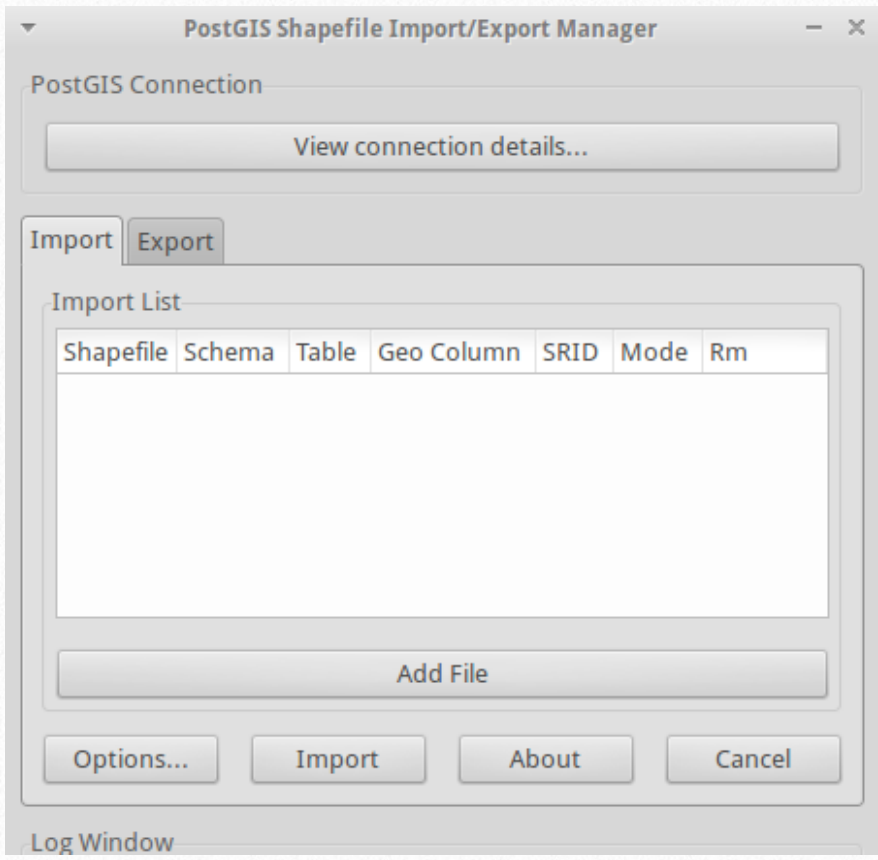
Load a shapefile into your new database

Supported by a wide variety of libraries and applications, PostGIS provides many options for loading data. This section will focus on the basics – loading shapefiles using the PostGIS shapefile loading tool.

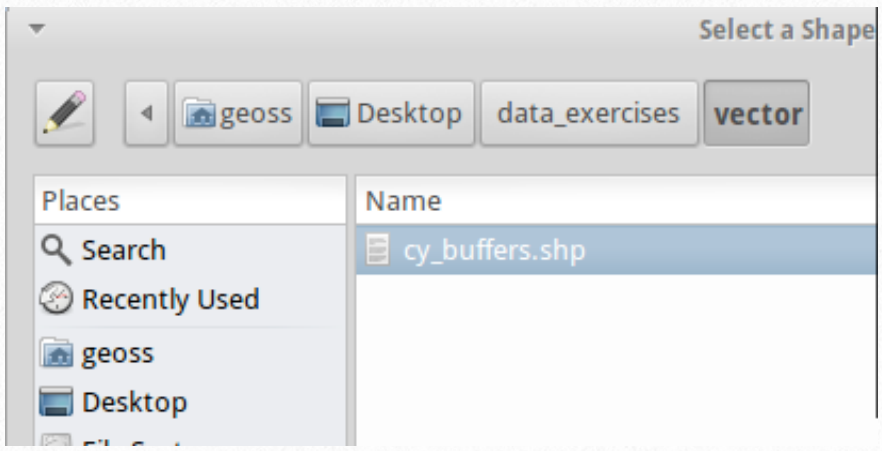
- First, **double click** the **pgShapeLoader** from the desktop of your virtual machine:



The `PostGIS Shapefile Import/Export manager` opens:



- Click **Add File** and **navigate to** the **data_exercices/vector** directory



- **Select** the **cy_buffers.shp** file.
- Click **Open**.

- Click **View Connection Details**.
- Fill in the details for the PostGIS Connection section as follows:

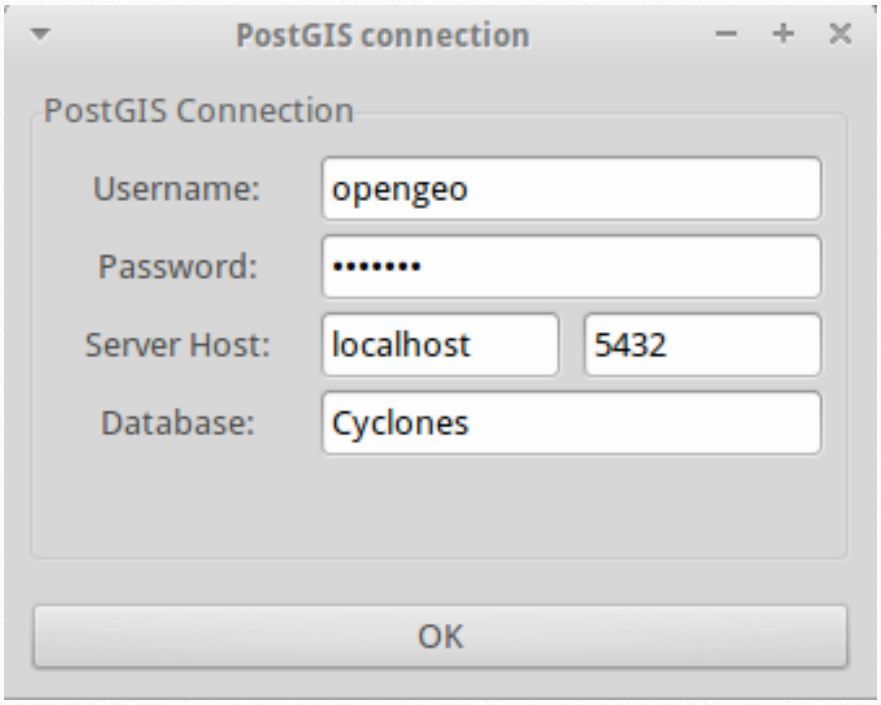
Username: **opengeo**

Password: **opengeo**

Server host: **localhost**

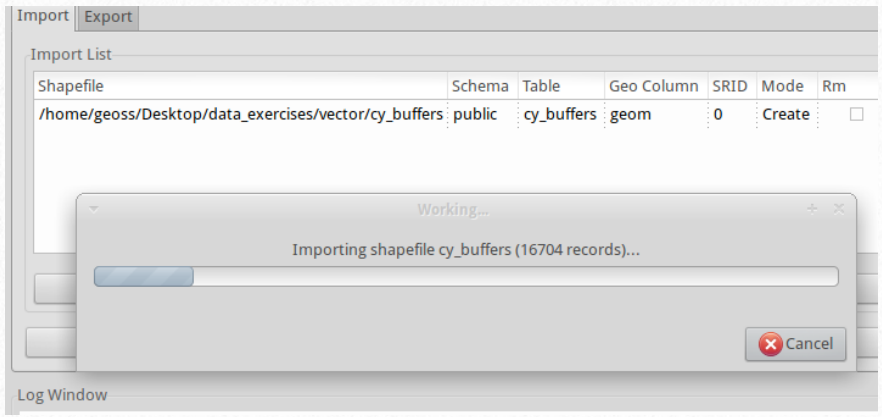
Port: **5432**

Database: **Cyclones**

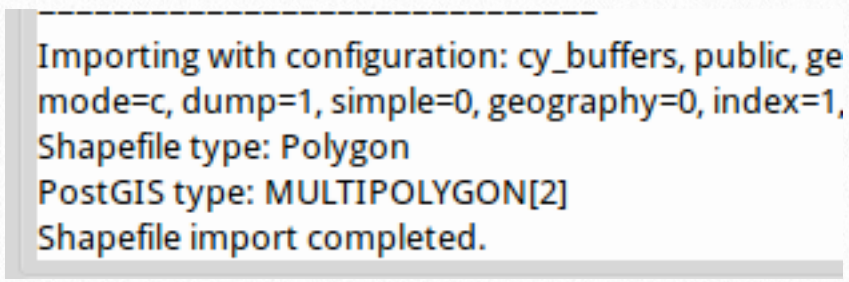


- Click **OK**.
- Click **Import** to load the cyclones buffers into the Cyclones database.

The upload should take a few minutes, but this is the largest file that you will load during this course. A progress bar indicates you the progress of the operation:

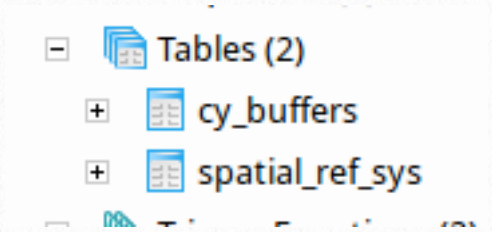


At the end of the operation a message should appear indicating that the import is complete:



- **Close** the **PostGIS Shapefile Import/Export manager**.
- From the **pgAdmin** console select **Tables(1)** and click the **Refresh** button to update the tree view:

You should see a new table showing up in the `Tables` section of the tree:



This means that you have successfully converted the cyclone buffers shapefile to the PostGIS format. This will allow gaining in performance when publishing this data in GeoServer later.

Shapefiles? What's that?

You may be asking yourself – *"What's this shapefile thing?"* A shapefile commonly refers to a collection of files with `.shp`, `.shx`, `.dbf`, and other extensions on a common prefix name (e.g., `cy_buffers`). The actual shapefile relates specifically to files with the `.shp` extension. However, the `.shp` file alone is incomplete for distribution without the required supporting files.

Mandatory files:

`.shp` — shape format; the feature geometry itself

`.shx` — shape index format; a positional index of the feature geometry

`.dbf` — attribute format; columnar attributes for each shape, in dBase III

Optional files include:

`.prj` — projection format; the coordinate system and projection information, a plain text file describing the projection using well-known text format

In order to analyze a shapefile in PostGIS, you need to convert a shapefile into a series SQL commands. By running `pgShapeLoader`, a shapefile converts into a table that PostgreSQL can understand.

SRID 4326? What's with that?

Most of the import process is self-explanatory, but even experienced GIS professionals can trip over an SRID.

An "SRID" stands for "Spatial Reference IDentifier." It defines all the parameters of our data's geographic coordinate system and projection. An SRID is convenient because it packs all the information about a map projection (which can be quite complex) into a single number.

You can see the definition of our workshop map projection by looking it up either in an online database, http://spatialreference.org/ref/epsg/4326/ or directly inside PostGIS with a query to the `spatial_ref_sys` table.

```
SELECT srtext FROM spatial_ref_sys
WHERE srid = 4326;
```

Note: The PostGIS `spatial_ref_sys` table is an OGC-standard table that defines all the spatial reference systems known to the database. The data shipped with PostGIS, lists over 3000 known spatial reference systems and details needed to transform/re-project between them.

In both cases, you see a textual representation of the 4326 spatial reference system:

```
GEOGCS["WGS 84",
    DATUM["WGS_1984",
        SPHEROID["WGS 84",6378137,298.257223563,
            AUTHORITY["EPSG","7030"]],
        AUTHORITY["EPSG","6326"]],
    PRIMEM["Greenwich",0,
        AUTHORITY["EPSG","8901"]],
    UNIT["degree",0.01745329251994328,
        AUTHORITY["EPSG","9122"]],
    AUTHORITY["EPSG","4326"]]
```

A common problem for people getting started with PostGIS is figuring out what SRID number to use for their data. Some of them have is a `.prj` file which stores a similar information as the one presented above. But how do humans

translate a `.prj` file into the correct SRID number?

The easy answer is to use a computer. Plug the contents of the .prj file into http://prj2epsg.org. This will give you the number (or a list of numbers) that most closely match your projection definition. There aren't numbers for every map projection in the world, but most common ones are contained within the `prj2epsg` database of standard numbers.

- You can now **close pgAdmin**.

The data that you uploaded in this Chapter will be published and processed later.

3

How to publish data?

Keywords: GeoServer; KML; WCS, WFS; WMS; SLD

What you will learn:

- Publish vector data as OGC services with GeoServer
- Publish raster data as OGC services with GeoServer
- Apply styles to the data
- Handle main GeoServer functions

GeoServer: learning basics

Overview

GeoServer is an open source software server written in Java that allows users to share and edit geospatial data. Designed for interoperability, it publishes data from any major spatial data source using open standards.

This section will discuss the basic concepts related to GeoServer and web mapping, including OGC protocols and useful terminology.

Servers basics

GeoServer is a server of geospatial data through the web. If you are familiar with how a web server functions, feel free to skip to the next section. For those used to working only with applications/clients, a brief discussion of what a web server is and how it works may be in order.

Web servers

A web server is a program that serves content (web pages, images, files, data, etc.) using HTTP (Hypertext Transfer Protocol). When you use your browser to connect to a website, you contact a web server. The web server takes the request, interprets it, and returns a response, which the browser renders on the screen.

For example, when you request a web page, your request takes the form of a URL:

```
http://example.com/some/path/page.html
```

The web server looks at its file system, and if that request points to a valid file (if `page.html` exists in `some/path`), the contents of that file will be returned via HTTP. Usually these calls come from a browser, in which case the result is rendered in the browser.

It is possible to request many different kinds of files through HTTP, not just HTML pages:

```
http://example.com/some/path/image.jpg
http://example.com/some/path/archive.zip
http://example.com/some/path/data.xml
```

If your browser is configured to display the type of file, it will be displayed, otherwise you will usually be asked to download the file to your host system.

The most popular web servers used today are Apache HTTP Server and Internet Information Services (IIS).

Web mapping servers

A web mapping server is a specialized subset in the web server model. Like a web server, requests are sent to the server which are interpreted and responded.

However, a web mapping server utilizes different protocols. Protocols that can be employed are Web Map Service (WMS), Web Feature Service (WFS), plus many other open and proprietary protocols. The protocols are designed specifically for the transfer of geographic information, whether it be raw feature data, geographic attributes, or map images.

Some popular web mapping servers:

- GeoServer
- MapServer
- Mapnik
- ArcGIS Server

Other web-based map services such as Google Maps have their own server technology as well, but clients have limited access via published APIs.

Data sources

GeoServer can read from many different data sources, from files on the local disk to external databases. Through the medium of web protocols, GeoServer acts as an abstraction layer, allowing a standard method of serving geospatial data regardless of the source data type.

The following is a list of the most common data formats supported by GeoServer. This list is by no means exhaustive.

- Files: Shapefile, GeoTIFF, ArcGrid, JPEG2000, GDAL formats,
- Databases: PostGIS, ArcSDE, Oracle Spatial, DB2, SQL Server.

OGC protocols

GeoServer implements standard open web protocols established by the Open Geospatial Consortium (OGC), a standards organization. GeoServer is in fact the reference implementation of the OGC Web Feature Service (WFS) and Web Coverage Service (WCS) standards, and contains as well a high performance certified compliant Web Map Service (WMS). It is through these protocols that GeoServer can serve data and maps in an efficient and powerful way.

Web Map Service (WMS)

One fundamental component of the web map (and the simplest to understand) is the map image. The Web Map Service (WMS) is a standard protocol for serving georeferenced map images generated by a map server. In short, WMS is a way for a client to request map tiles from a server. The client sends a request to a map server, then the map server generates an image based on parameters passed to the server in the request and finally returns an image.

It is important to note that the source material from which the image is generated does not need to be an image. The WMS generates an image from whatever source material is requested, which could be vector data, raster data, or a combination of the two.

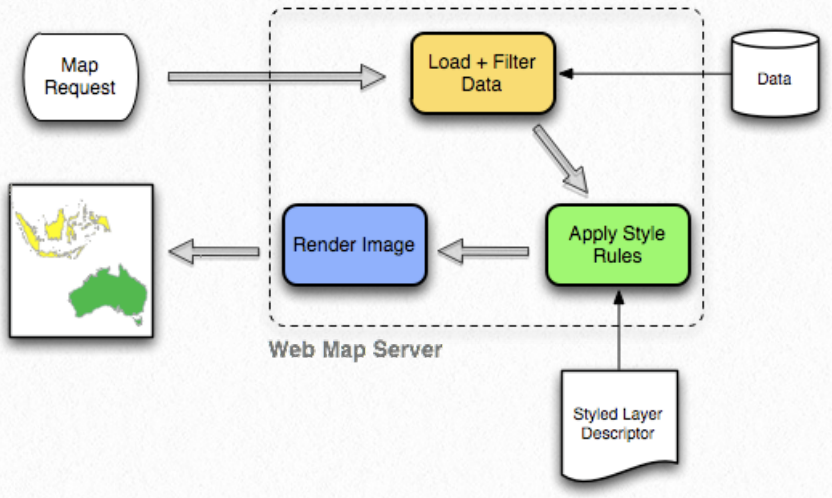


Diagram showing how a WMS turns data into a map image

Sample WMS request

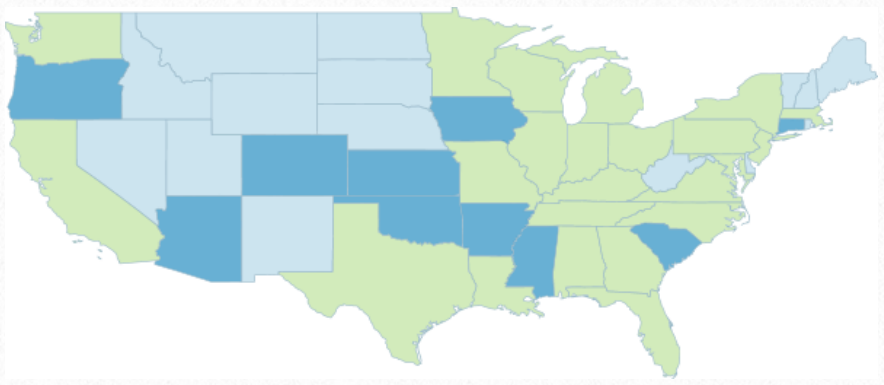
The following is a sample WMS request, rendered as a HTTP GET request (with line breaks added for clarity) to a hosted GeoServer instance:

```
http://suite.opengeo.org/geoserver/w
ms?
  SERVICE=WMS&
  VERSION=1.3.0&
  REQUEST=GetMap&
  LAYERS=usa:states&
  SRS=EPSG:4326&
  BBOX=24.956,-124.731,49.372,-66.97&
  FORMAT=image/png&
  WIDTH=600&
  HEIGHT=255
```

While the details of the WMS protocol are beyond the scope of this course, a quick scan of this request shows that the following information is being requested:

- Server details (a WMS 1.3.0 request)
- Request type (a WMS GetMap request)
- Layer name (usa:states)
- Projection (EPSG:4326)
- Bounding box (in this case, latitude/longitude coordinates)
- Image properties (600x255 PNG)

If you paste the request into a browser, the result would be:



An image generated by a WMS request

Other WMS requests

A WMS request can ask for more than just a map image ("`GetMap`"). An example of another such request is a request for information about the WMS server itself. The request is called GetCapabilities, and the response is known as the capabilities document. The capabilities document is an XML response that details the supported image formats, projections, and map layers being served by that WMS.

The following is a WMS GetCapabilities request given to the same WMS used above:

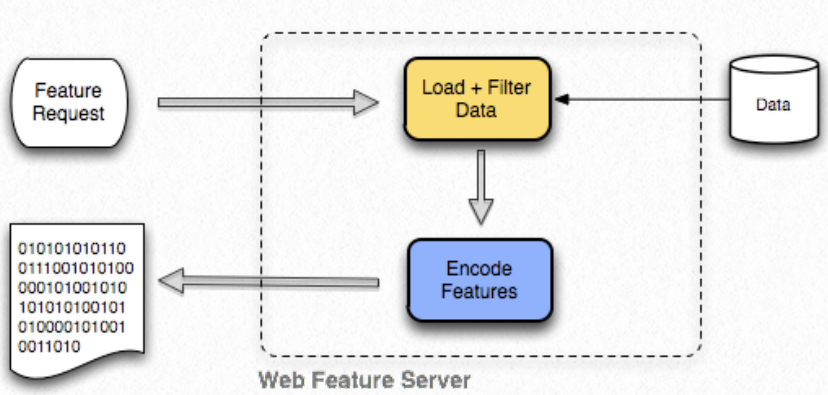
```
http://suite.opengeo.org/geoserver/w
ms?
    SERVICE=WMS&
    VERSION=1.3.0&
    REQUEST=GetCapabilities
```

You can paste this request into a browser to see the result.

For more information, please see the OGC site on the Web Map Service protocol at http://www.opengeospatial.org/standards/wms.

Web Feature Service (WFS)

A web mapping server can also return the actual geographic data that comprise the map images. One can think of the geographic data as the "source code" of the map. This allows users to create their own maps and applications from the data, to convert data between certain formats, and be able to do raw geographic analysis of served data. The protocol used to return geographic feature data is called Web Feature Service (WFS).



A diagram showing how a WFS turns a request into a response

Sample WFS request

The following is a sample WFS request, rendered as a HTTP GET request (with line breaks added for clarity) to a hosted GeoServer instance:

```
http://suite.opengeo.org/geoserver/wfs?
    SERVICE=wfs&
    VERSION=1.1.0&
    REQUEST=GetFeature&
    TYPENAME=usa:states&
    FEATUREID=states.39
```

While the details of the WFS protocol are beyond the scope of this course, a quick scan of this request shows that the following information is being requested:

- Server details (WFS 1.1.0 request)
- Request type (GetFeature)
- Layer name (usa:states)
- Feature ID (states.39)

This particular request polls the WFS for a single feature in a layer.

Note: A feature is a single unit of geographic data, such as a polygon or point. The above feature requested is the multipolygon for the state of New York.

Paste the request into a browser to see the result. The response contains the coordinates for each vertex in the feature in question, along with the attributes associated with this feature. Scroll down to the bottom to see the feature attributes.

XML generated by a WFS request

While XML is difficult to read, it is easy for computers to parse, which makes WFS responses

ideal for use in software. GeoServer offers other output formats as well, such as JSON, CSV, and zipped shapefile.

Other WFS requests

A WFS request can ask for much more than just feature data. An example of another such request is to request information about the WFS server. The request is called GetCapabilities, and the response is known as the capabilities document. The capabilities document is an XML response that details the supported data layers, projections, bounding boxes, and functions available on the server.

The following is a sample WFS GetCapabilities request:

```
http://suite.opengeo.org/geoserver/wfs?
    SERVICE=WFS&
    VERSION=1.1.0&
    REQUEST=GetCapabilities
```

You can paste this request into a browser to see the result.

For more information, please see the OGC site on the Web Feature Service protocol at http://www.opengeospatial.org/standards/wfs.

Other protocols

While beyond the scope of this workshop, it is worth noting that GeoServer offers limited support for other protocols in addition to Web Map Service (WMS) and Web Feature Service (WFS).

Web Coverage Service (WCS)

The Web Coverage Service is a service that enables access to the underlying raster (or "coverage") data. In a sense, WCS is the raster analog to WFS, where you can access the actual raster data stored on a server.
GeoServer contains full support for WCS versions up to 1.1.1.

For more information on WCS, please see the OGC site on the Web Coverage Service protocol at http://www.opengeospatial.org/standards/wcs.

Web Processing Service (WPS)

The Web Processing Service (WPS) is a service for the publishing of geospatial processes, algorithms, and calculations. WPS extends the web mapping server to provide geospatial analysis. WPS in GeoServer allows for direct integration with other GeoServer services and the data catalog. This means that it is possible to create processes based on data served in GeoServer, including the results of a process to be stored as a new layer. In this way, WPS acts as a full browser-based geospatial analysis tool, capable of reading and writing data from and to GeoServer.
A much newer development in GeoServer, WPS is available only as an extension. Development continues support for this protocol, though, and eventually support will be added to the standard build.

For more information on WPS, please see the OGC site on the Web Processing Service proto-

c o l a t
http://www.opengeospatial.org/standards/wps.

GeoServer concepts

GeoServer uses lots of terminology, and this can sometimes be confusing, especially if you are unaccustomed to web mapping. This section will introduce some of the most common terms used in GeoServer. We will be using all of this information in the upcoming sections.

Workspace

Note: A workspace is also sometimes known as a “namespace”.

A workspace is the name for a notional container to group similar data together. It is designed to be a separate, isolated space relating to a certain project. Using workspaces, it is possible to use layers with identical names without conflicts.

Workspaces are usually denoted by a prefix to a layer name or store name. For example, a layer called streets with a workspace prefix called nyc would be referred to by `nyc:streets`.

Stores and layers must all have an associated workspace.

Note: Technically, the name of a workspace is a URI, not the short prefix. A URI is a Uniform Resource Identifier, which is similar to a URL, but does not need to resolve to a web site. In the above example, the full workspace could have been `http://opengeo.org/nyc` in which case the full layer name would be `http://opengeo.org/nyc:streets`

GeoServer intelligently replaces the workspace prefix with the full workspace URI, but it can be useful to know the difference.

Store

Note: A store is also sometimes known as a “datastore” when referring to vector (or feature) data, and “coveragestore” when referring to raster (or coverage) data.

A store is the name for a container of geographic data. A store refers to a specific data source, be it a shapefile, database, or any other data source that GeoServer supports.

A store can contain many layers, as in the case of a database that contains many tables. A store can also have a single layer, such as in the case of a shapefile or GeoTIFF, which can only contain one layer. A store must contain at least one layer.

GeoServer saves the connection parameters to each store (such as a path to the shapefile, or credentials to connect to the database). Each store must also be associated with one (and only one) workspace.

Layer

Note: A layer is also sometimes known as a “feature type”.

A layer is a collection of geospatial features or a coverage. Typically a layer contains one type of data (points, lines, polygons, raster) and has a single identifiable content (streets, houses, country boundaries, etc.). Aside from individual fea-

tures, a layer is the smallest grouping of geospatial data.

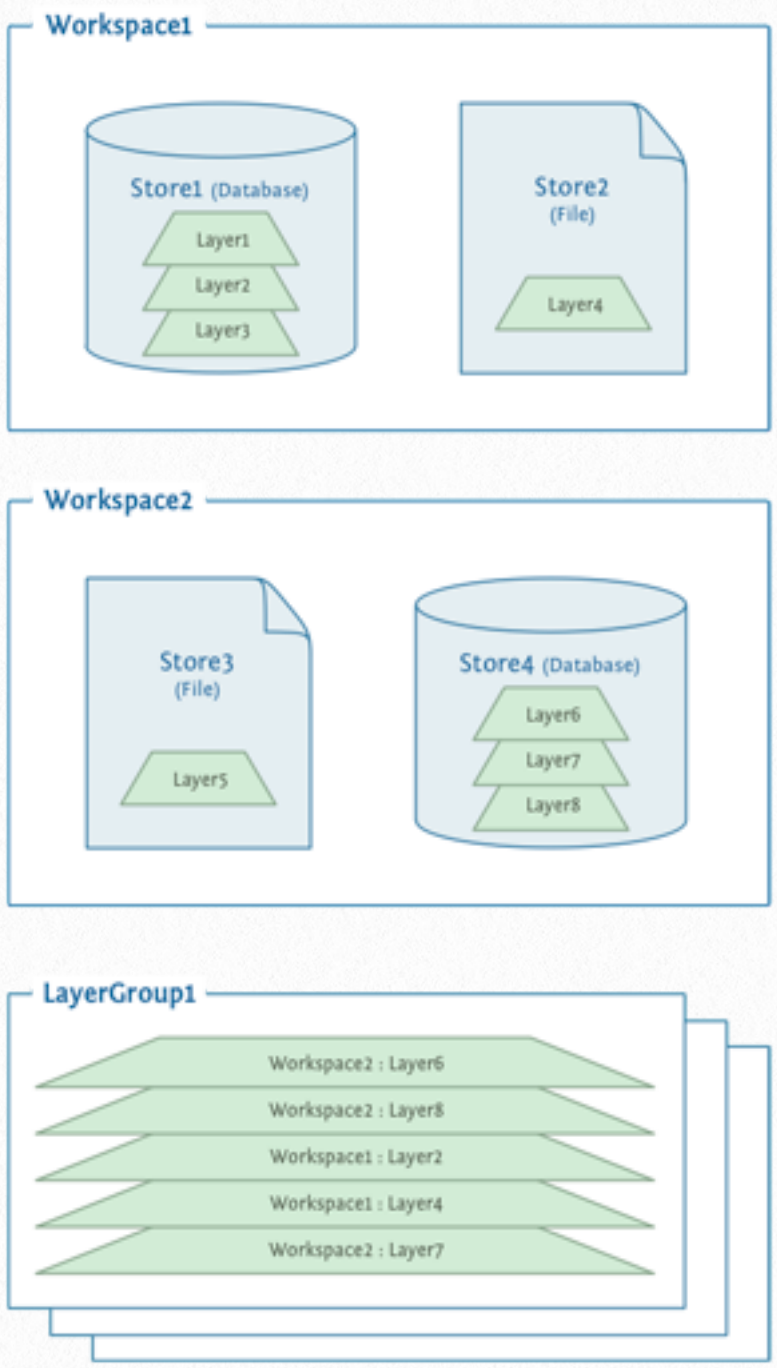
A layer corresponds to a table or view from a database, or an individual file.

GeoServer stores information associated with a layer, such as projection information, bounding box, associated styles, and more. Each layer must be associated with one (and only one) workspace.

Layer group

A layer group, as its name suggests, is a collection of layers. A layer group makes it possible to request multiple layers with a single WMS request. A layer group contains information about the layers that comprise the layer group, the order in which they are rendered, the projection, associated styles, and more. This information can be different from the defaults for each individual layer.

Layer groups do not respect the concept of workspace, and are relevant only to WMS requests.



Relationships between workspaces, stores, layers, and layer groups

Style

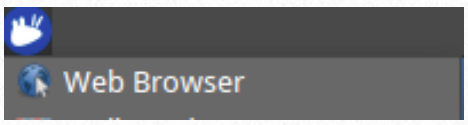
A style is a visualization directive for rendering geographic data. A style can contain rules for color, shape, and size, along with logic including attribute-based rules and zoom-level-based rules.

Every layer must be associated with at least one style. GeoServer recognizes styles in Styled Layer Descriptor (SLD) format.

GeoServer Web Administration

GeoServer includes a web-based administration interface. Most GeoServer configuration can be done through this interface, without the need to edit configuration files by hand.

This section will give a brief overview of the web interface. Subsequent sections will use the web interface in greater detail. In order to follow this section, open the web browser of your virtual machine



and on the GeoServer link of the start page:

GeoServer: http://localhost:8080/geoserver

Viewing

The default location of the GeoServer admin interface is `http://localhost:8080/geoserver`. The initial page is called the Welcome page.

GeoServer Welcome page

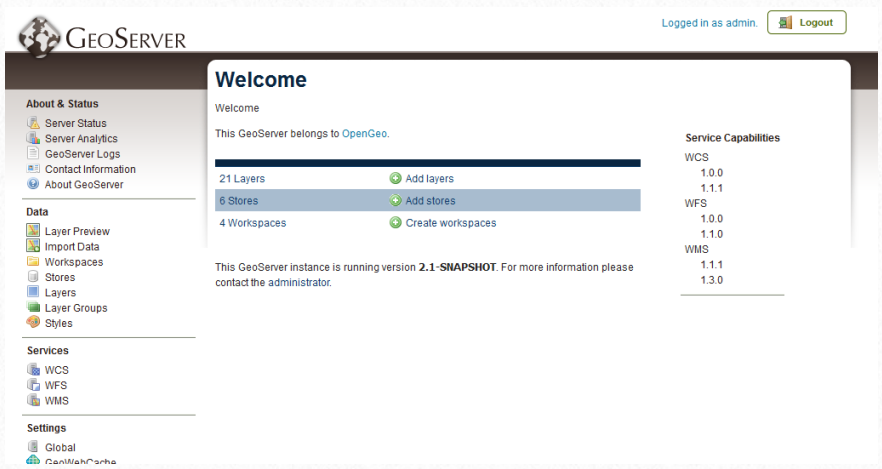
Authentication

For security reasons, most GeoServer configuration tasks require you to be logged in first. By default, the GeoServer administration credentials are `admin` and `geoserver`, although this can be changed.

Log in using the default credentials.

Logging in with default credentials

After logging in, many more options will be displayed.



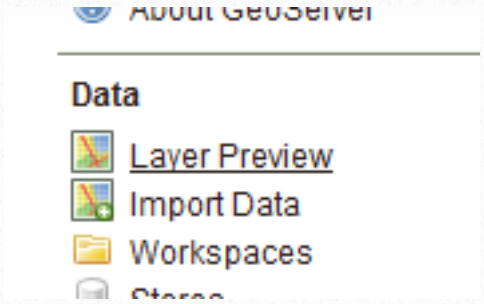
GeoServer welcome page with administrative options

Navigation

Use the links on the left side column to manage GeoServer, its services, data, security settings, and much more. Also on the main page are direct links to the capabilities documents for each service (WFS, WMS, WCS). We will be using the links on the left under Data (Workspaces, Stores, Layers, etc.) very often in this workshop, so it is good to familiarize yourself with their location.

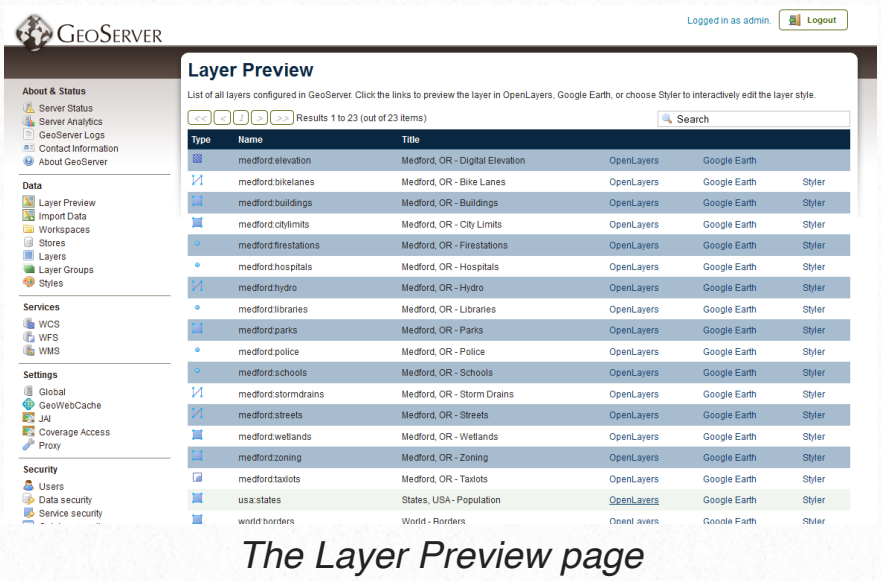
Layer Preview

You can use the Layer Preview link to easily view layers currently being served by GeoServer. The Layer Preview pages includes quick links to viewing layers via OpenLayers, along with other services.

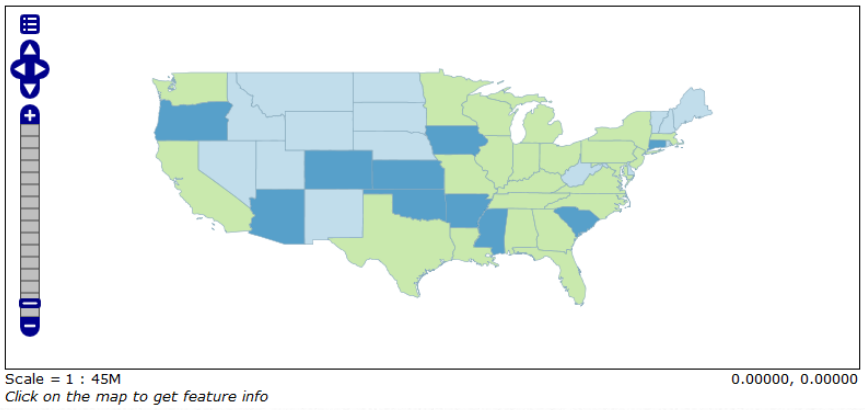


Navigating to the Layer Preview page

Preview a few layers by clicking on the `OpenLayers` link next to each layer.



The Layer Preview page



Viewing the `usa:states` layer

Note: Take a look at the contents of the URL bar when viewing an OpenLayers map. It is similar in construction to the sample WMS requests made in the Web Map Service (WMS) section. The salient difference is the use of format=application-/openlayers as the output format.

Logs

GeoServer displays the contents of the logs without the need to drop to a command line. Reading the logs is helpful when troubleshooting. To view the logs, **click on `GeoServer Logs`** on the left in the About & Status section.

View the GeoServer application logs

Working with data

Loading and publishing data is the core of GeoServer. Next section will detail how to set up a new project in GeoServer, as well as load data from multiple sources in different ways. After the data is loaded, a layer group will be created.

GeoServer: publishing geospatial data

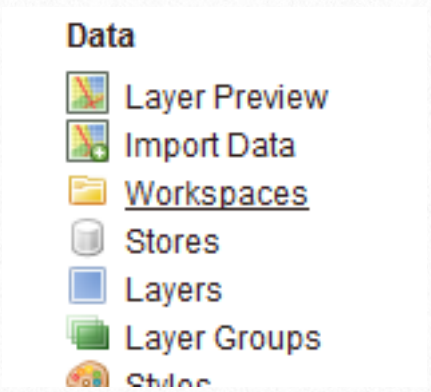
1. Adding a workspace

The first step in data loading is usually to create a workspace. This creates a virtual container for your project. Multiple layers from multiple sources can all be contained inside a workspace, with the primary constraint being that each layer name be unique.

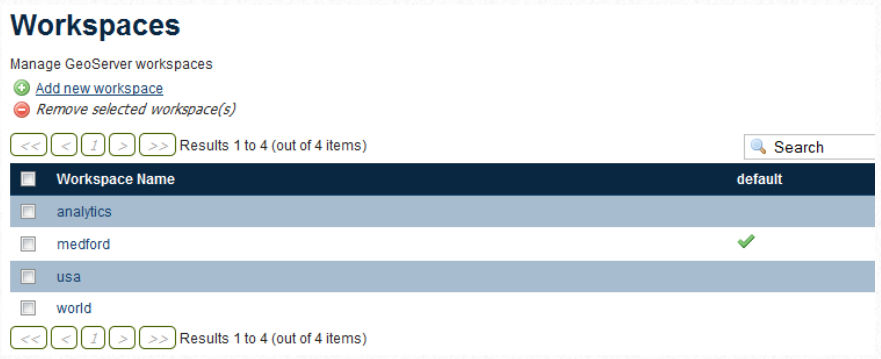
- Navigate to the main GeoServer Web Administration page: **http://localhost:8080/geoserver**

- **Login** as **admin** / **geoserver**

- Click on the **Workspaces** link on the left column, in the `Data` section.



- Click on the **Add new workspace** link at the top center of the page:

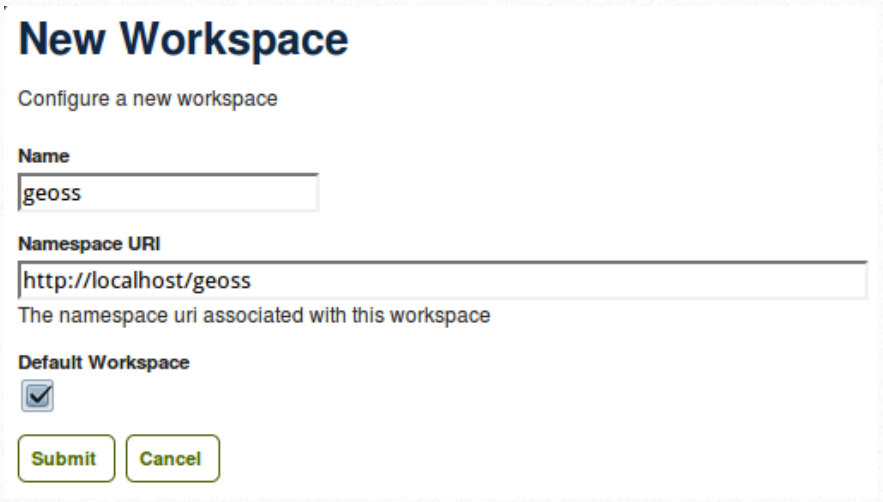


- A workspace is comprised of a Name (also sometimes known as a “namespace prefix”), represented by a few characters, and a Namespace URI. These two fields must uniquely identify the workspace. Fill in the following information:

  - Name: **geoss**

  - Namespace URI: **http://localhost/geoss**

  - Default workspace: **Checked**



- When done, **click Submit**.

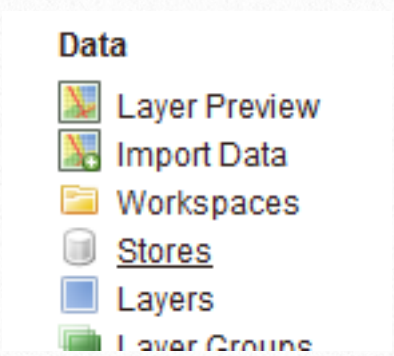
2. Publishing layers in GeoServer

2.1. Publishing a PostGIS layer

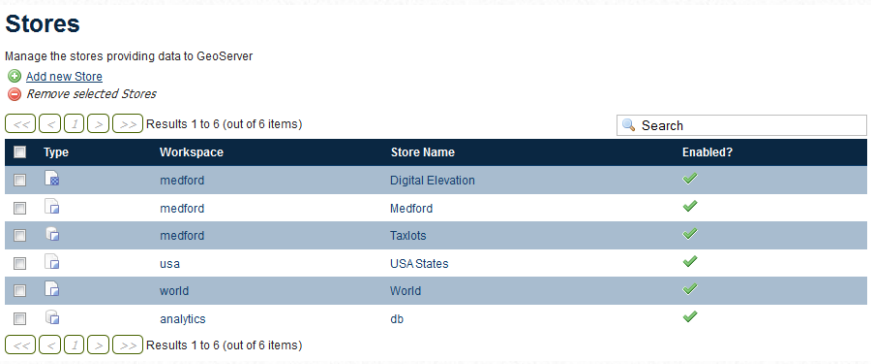
Adding a single dataset to GeoServer is one of the simplest data loading tasks. To start our discussion of data loading, we will load a PostGIS dataset showing cyclone buffers.

First we need to load a PostGIS store. In GeoServer terminology, a PostGIS store is a store that contains a single layer. We must add the store to GeoServer before we can publish a layer.

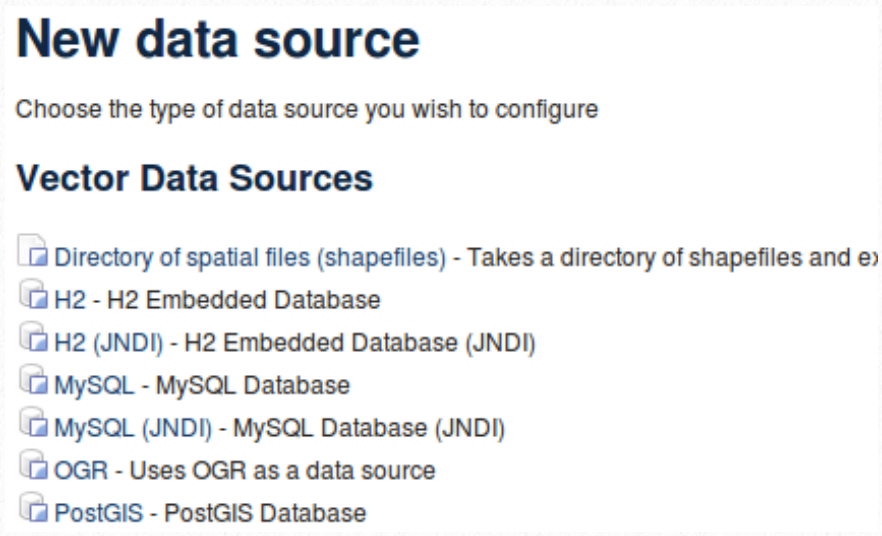
- From the GeoServer Web Administration page, **click** the **Stores** link on the left side, in the `Data` section.



- Click on **Add new store**.



- Select **PostGIS - PostGIS Database** under `Vector Data Sources`.



- Fill out the following form, leaving all other fields as the default:
  - Workspace: **geoss** should be the default.
  - Data Source Name: **cy_buffers**

    This can be anything, but it makes sense to match this with the name of the PostGIS database.
  - Description: **Cyclone buffers PostGIS**
  - Enabled: **Checked**

    Ensures the layer is published. Unchecking will save configuration information only.
  - host: **localhost**
  - port: **5432**
  - database: **`Cyclones`**
  - `schema:` **`public`**
  - `user:` **`opengeo`**
  - `password:` **`opengeo`**

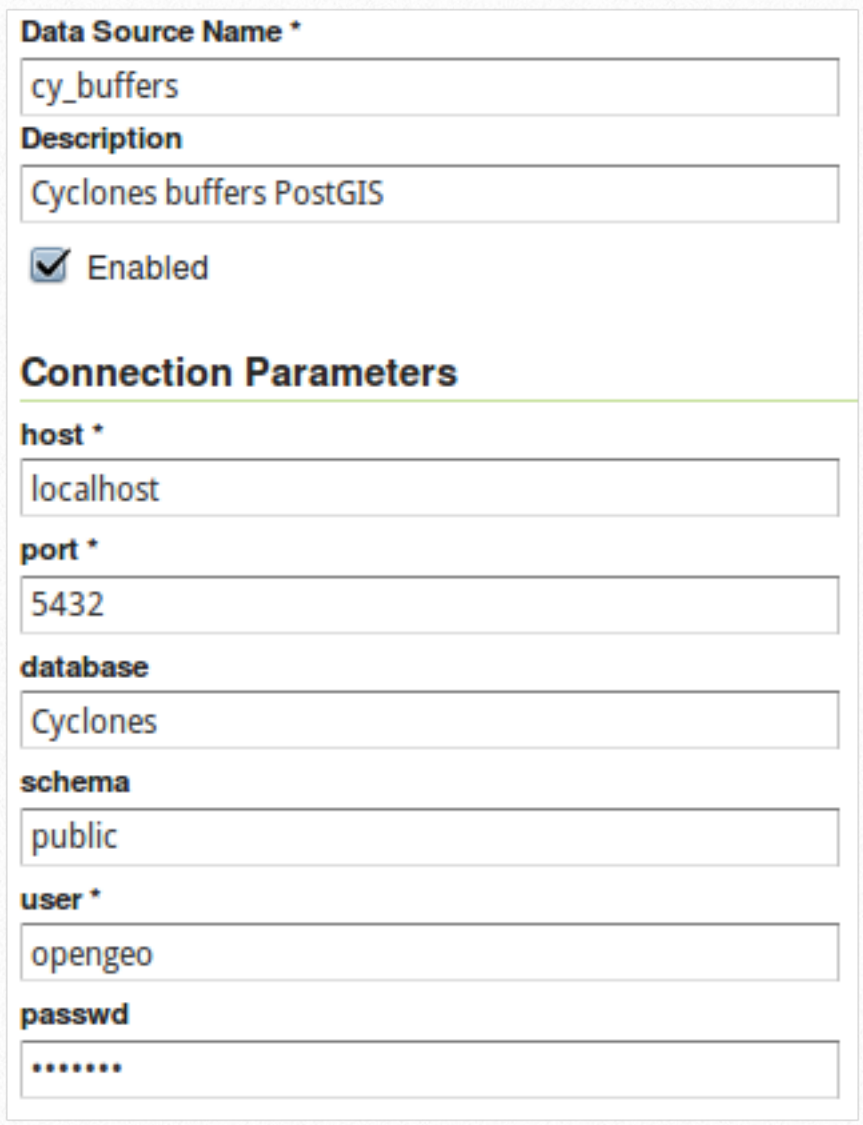
- When finished, click on **Save** at the bottom of the page.

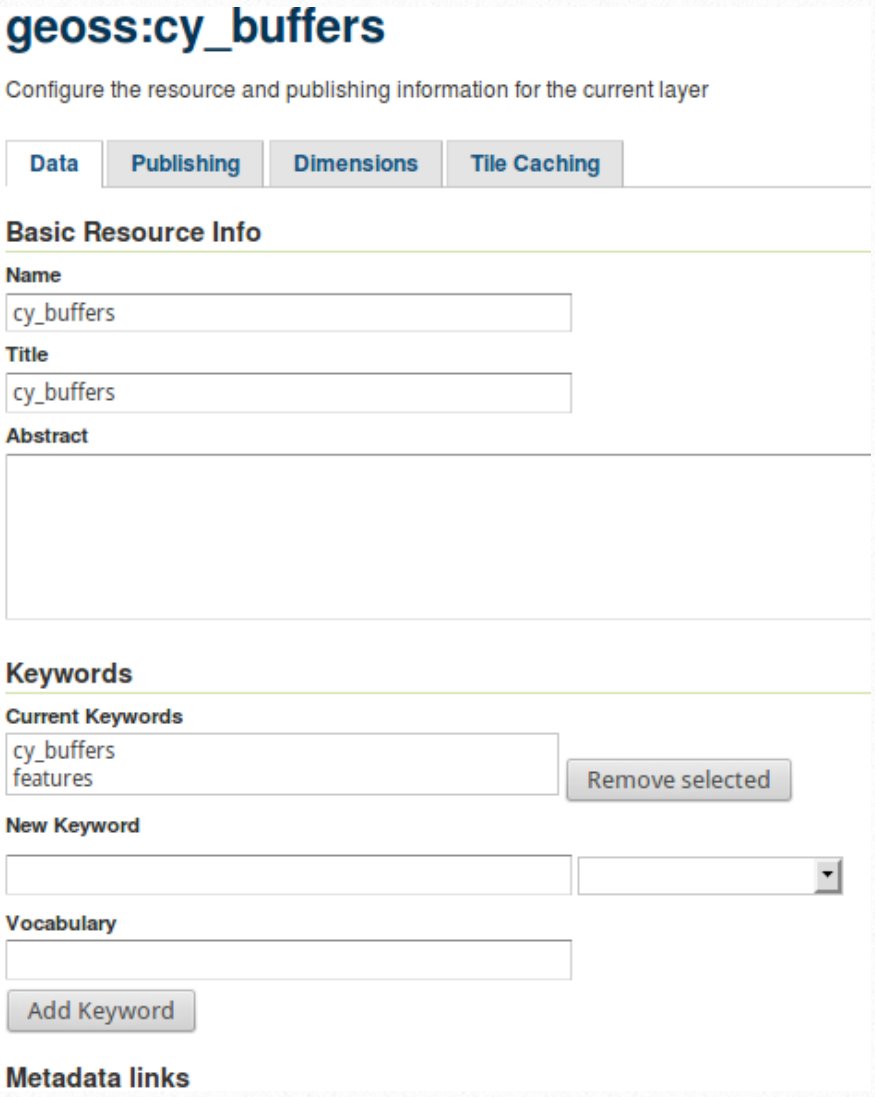
Publishing a layer

We have loaded a store, but our layer has yet to be published. We will do that now.

- Click **Publish**:

This opens the layer configuration page. There are many settings on this page; we don't need to work with most of them right now. We will return to some of these settings later. The following information should already be filled out:

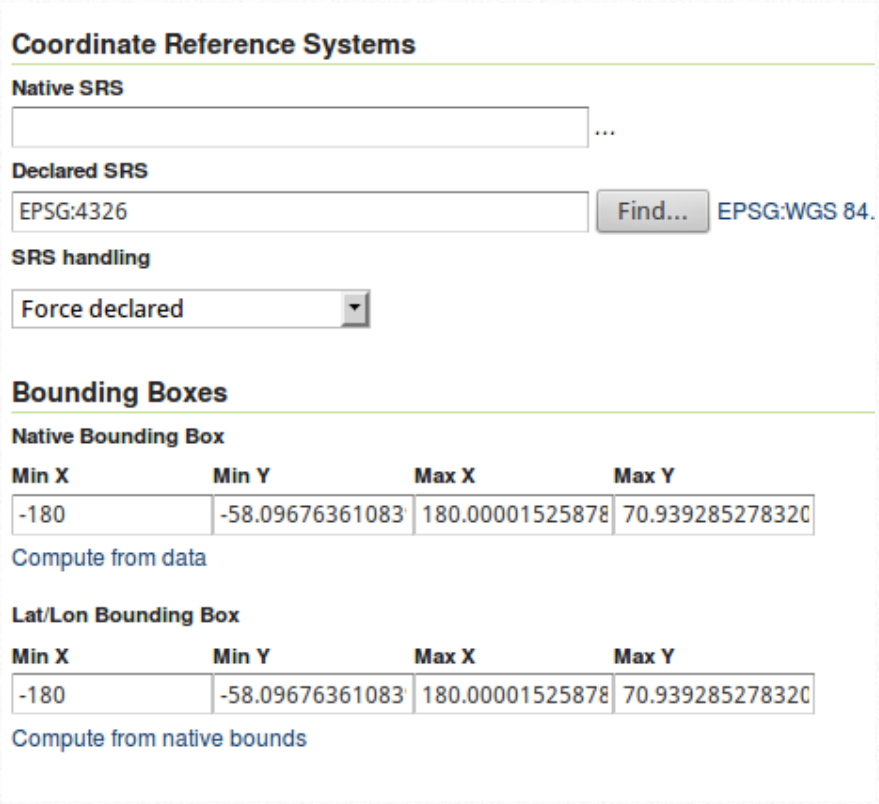




- Set the **Declared SRS** to **EPSG:4326** if not already done.

- Set the **SRS handling** to **Force declared** if not already done.

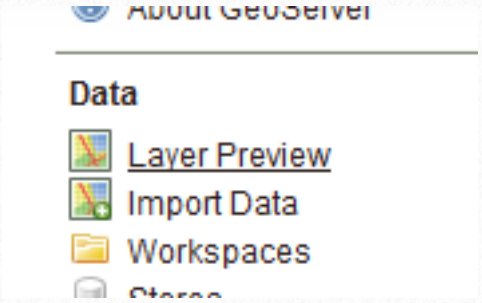
- In the `Bounding Boxes` section, click on the **Compute from data** and **Compute from native bounds** links to set the bounding box of the layer.



- When finished, click **Save**. Your layer is now published in GeoServer!

You can now view the layer using the integrated OpenLayers client (using WMS).

- Click on the **Layer Preview** link.



- A list of published layers is displayed. Find the layer in the list, and click the OpenLayers **Go** link next to the entry in the list.

Note: Lists in GeoServer are paged at 25 items at a time. If you can't find the layer, you may need to click the [2] or [>] buttons.

A new tab in your browser will open up, showing your layer inside an OpenLayers application. Play around with this window; you can use your mouse to zoom and pan, and can also click on the layer features to display attribute information.

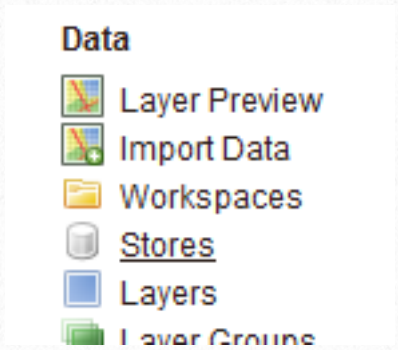
Note: If you're wondering where the style/color is coming from, this will be discussed in the Styling section.

2.2. Publishing a shapefile

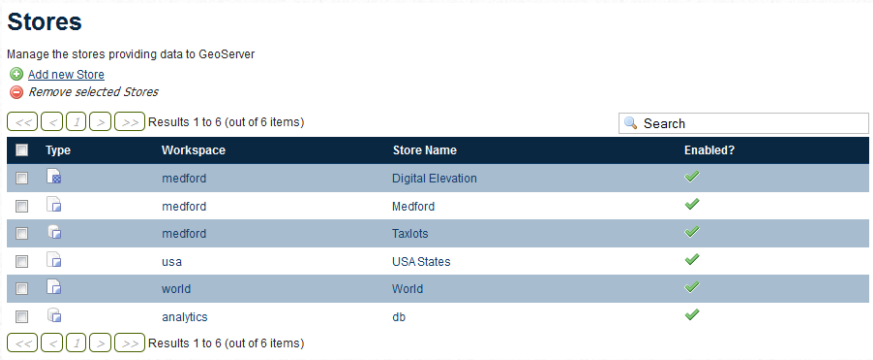
Publishing a shapefile is similar to publishing a PostGIS table. The only difference it that instead of using a PostGIS store, you will use a shapefile store.

Note: the data used in this Section was provided by http://naturalearthdata.com . See the `ne_10m_coastline.REAME.html` file in the `/Home/geoss/Desktop/data_exercises/vector` directory.

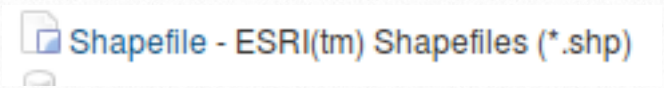
- Click the **Stores** link at the left of the page



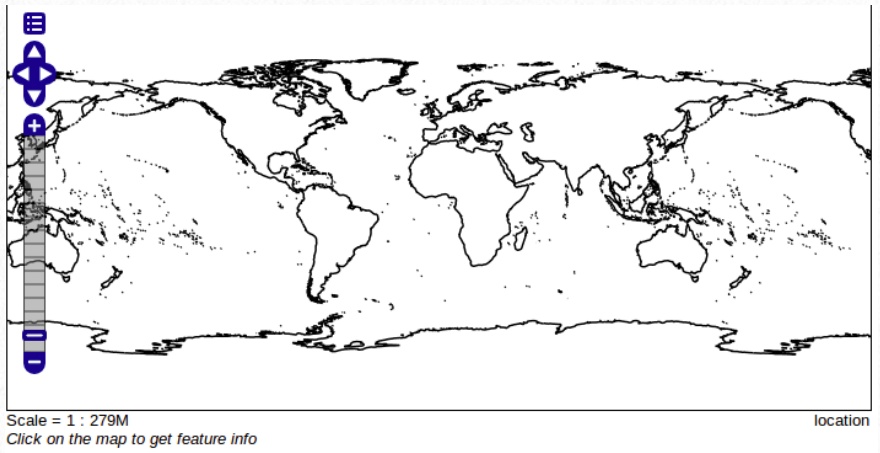
- Click **Add new store**.



- Select **Shapefile - Esri(tm) Shapefiles (.shp)**:



- Fill the shapefile form as follows and `save`.
    - Workspace: **geoss**
    - Data Source Name: **Coastlines**
    - Enabled: **check**
- Shapefile location: browse to **Home/geoss/Desktop/data_exercises/vector**
- Click **Save**.
- Click **Publish**.
- **Fill the form** exactly as you did for the PostGIS layer. The coordinate system is the same: `EPSG:4326` which corresponds to WGS 84.
- Click **Save**.
- From **Layer preview** scroll down to `geoss:ne_10m_coastline` at the bottom of the page and click **Go**:



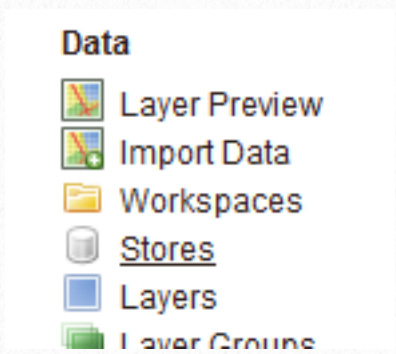
The layer is now published in GeoServer and is displayed as WMS from your local server.

2.3. Publishing a GeoTiff

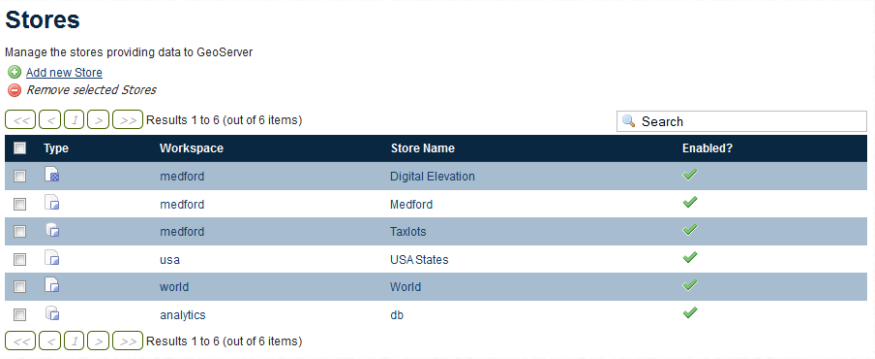
In the previous section, we published a shapefile. Shapefiles contain vector information, that is points, lines, or polygons. GeoServer can also publish raster imagery. This could be simple georeferenced-images (such as blue marble imagery) all the way to multi-band DEM (digital elevation model) data. In this section, we will load a simple GeoTIFF showing a shaded relief at the global level. This raster will be useful later when you will create a layer group.

The procedure for adding a store for a GeoTIFF is very similar to that of a shapefile. A GeoTIFF, like a shapefile, is a store that contains a single layer.

- From the GeoServer Web Administration page, click on the **Stores** link on the left side, under Data.



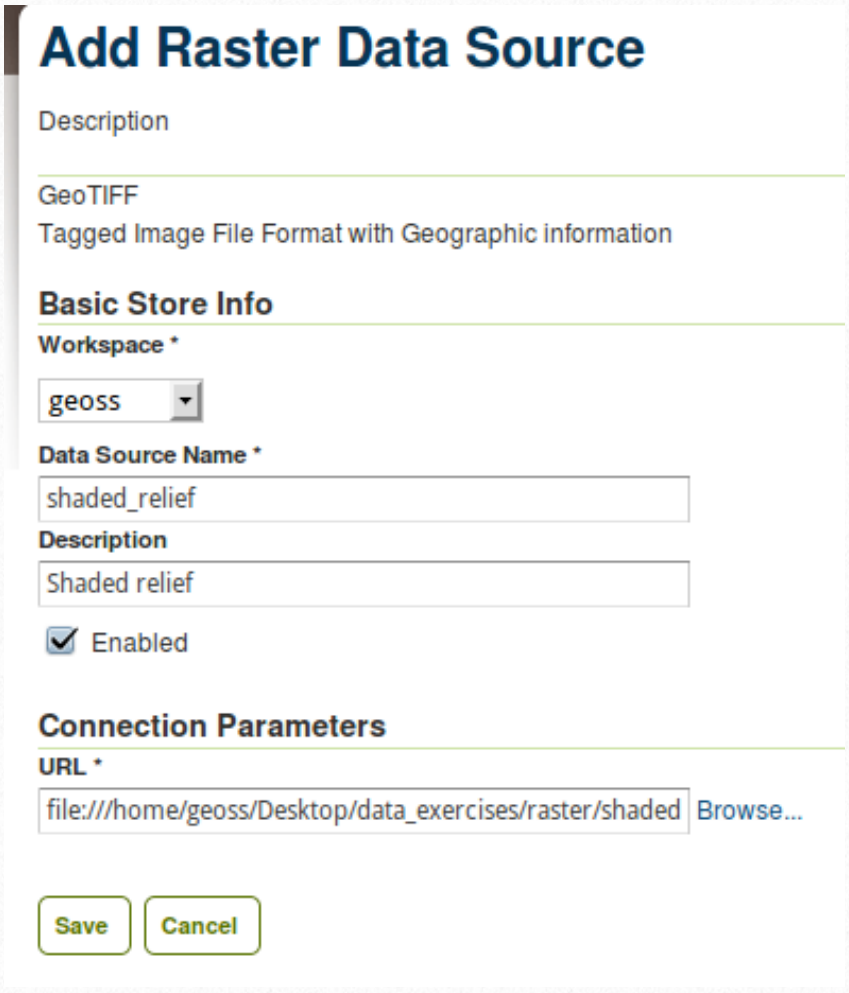
- Click on **Add new store**.



- Select **GeoTIFF** under `Raster Data Sources`.

- Fill out the form as follows:
  - Workspace: **geoss**
  - Data Source Name: **shaded_relief**
  - Enabled: **check**

- In the box marked URL, type in the full path to the GeoTIFF, or click the Browse button to navigate to the **/Home/geoss/Desktop/data_exercises/raster/shaded_relief.tif** file



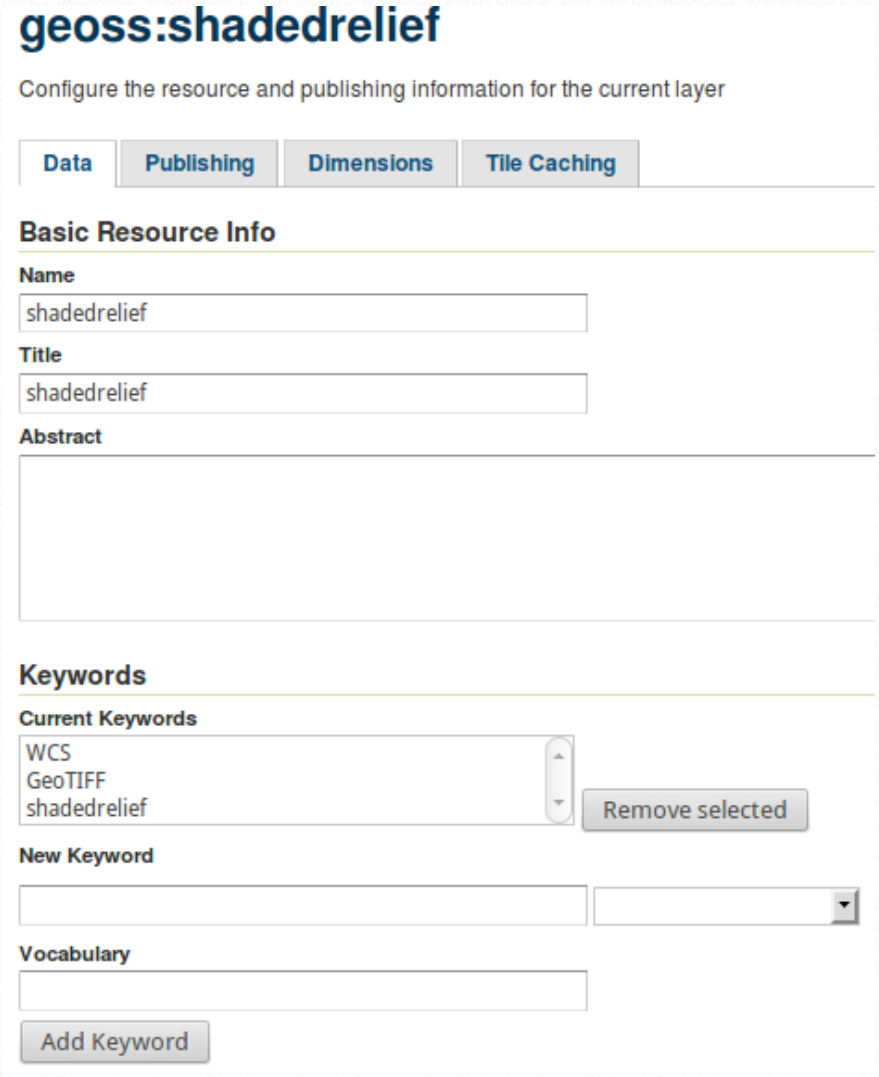
- When finished, click **Save**.

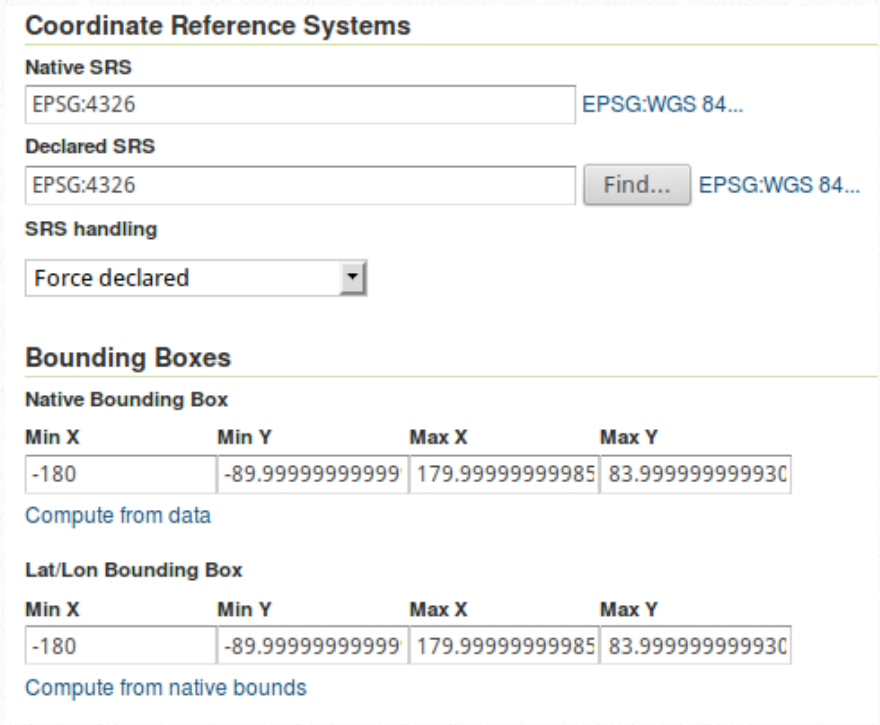
As with the shapefile, after the store is loaded, we now need to configure the layer itself.

- As for vector data you need to **Publish** the new store to configure the layer.

This opens the layer configuration page. There are many settings on this page, most of which we don't need to work with just now. We will return to some of these settings later.

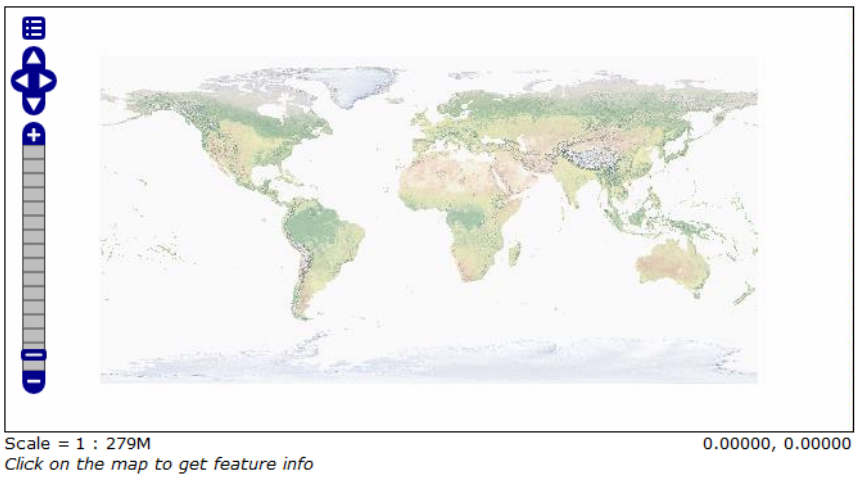
- **Fill out the form** with the following info (if not already done by GeoServer):





- Set the **Declared SRS** to **EPSG:4326** if not already there.
- Set the **SRS handling** to **Force declared** if not already there.
- In the `Bounding Boxes` section, click on the **Compute from data** and **Compute from native bounds** links to set the bounding box of the layer (if `Min X`, `Min Y`, `Max X` and `Max Y` are not already filled out):

- When finished, click **Save**. Your GeoTIFF is now published in GeoServer!
- You can now view the layer using the integrated OpenLayers client (using WMS) via the **Layer Preview** as in previous sections. Clicking on the map will display the RGB values for that particular point.



Your GeoTIFF has been successfully published in GeoServer!

2.4. Publishing multiple layers

So far we have loaded and published three different files in GeoServer. While the process is not complicated, it isn't very efficient when trying to load many layers. Fortunately, there are ways to load multiple files into GeoServer.

2.4.1. Directory of shapefiles

In our list of possible data sources (in the `Add new store` page), there is an option for `Directory of spatial files` (shapefiles). This allows you to load a directory of shapefiles as a single store, with each individual file inside the directory being a publishable layer. Using a single store has its advantages, but each layer still needs to be configured manually, so it can still be inefficient for many layers.

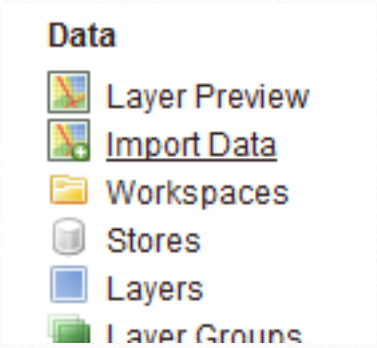
2.4.2. REST

GeoServer also has a full RESTful API for loading and configuring GeoServer. With this interface, one can create scripts (via bash, PHP, etc) to batch load any number of files.

The REST interface is beyond the scope of an introductory workshop, but those interested can read the REST section of the GeoServer documentation at http://docs.geoserver.org/stable/en/user/restconfig/index.html .

2.4.3. Layer Importer

The OpenGeo Suite includes a custom extension to GeoServer called the `Layer Importer`. This tool automates the process of uploading and configuring data into GeoServer. The importer will automatically determine projection information and bounding box, and will generate a unique style for each layer. The importer works with shapefiles, database tables such as PostGIS, Oracle, and ArcSDE, as well as with raster mosaics. This function is accessible from the `Import Data` link.



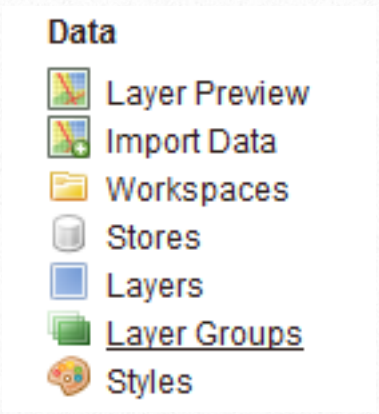
2.4.4. Creating a layer group

A layer group, as its name suggests, is a group of layers that acts as a single layer. This is useful when creating a base map, or other situations when more than one separate layer needs to be requested simultaneously or frequently. Since layers typically contain only a single type of geometry, using a layer group also allows you to combine data types in one single WMS request.

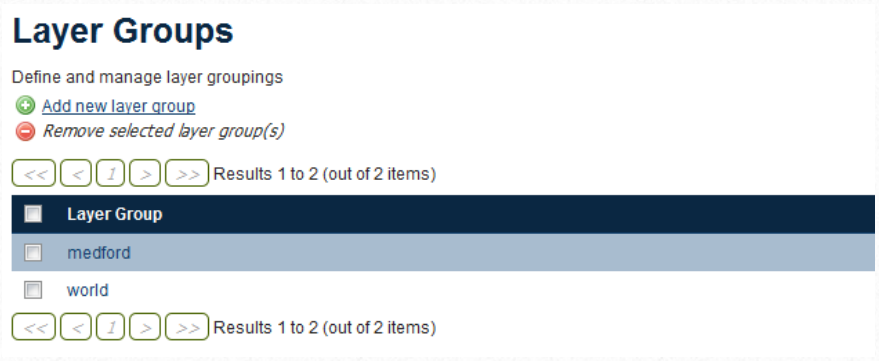
Note: Take care not to get confused between a workspace, which is a notional grouping of layers (think "container"), and a layer group, which is a group of layers for WMS requests (think "image group").

In the previous section, we loaded a few layers. Now we'll use a layer group to combine them.

- From the GeoServer Web Administration page, click on the **Layer Groups** link, under `Data` on the left side of the page.

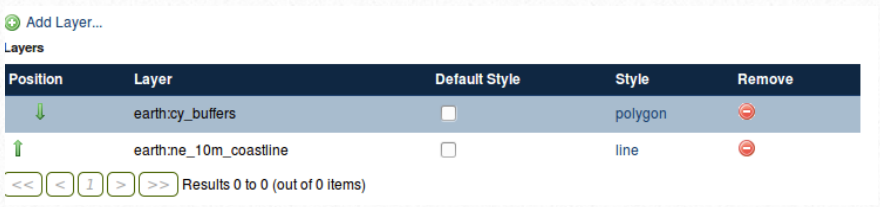


- Click **Add new layer group** at the top of the page.



- Enter **geoss map** in the `Name` field.
- Enter **geoss map** in the `Title` field.
- Choose the Workspace **`geoss`**

- Now we will add layers to our layer group (skipping the Bounds and other form fields for now). Click on the **Add Layer** link.
- Select each of the following layers so that they appear in this order:
  - **shaded_relief**
  - **cy_buffers**
  - **ne_10m_coastline**



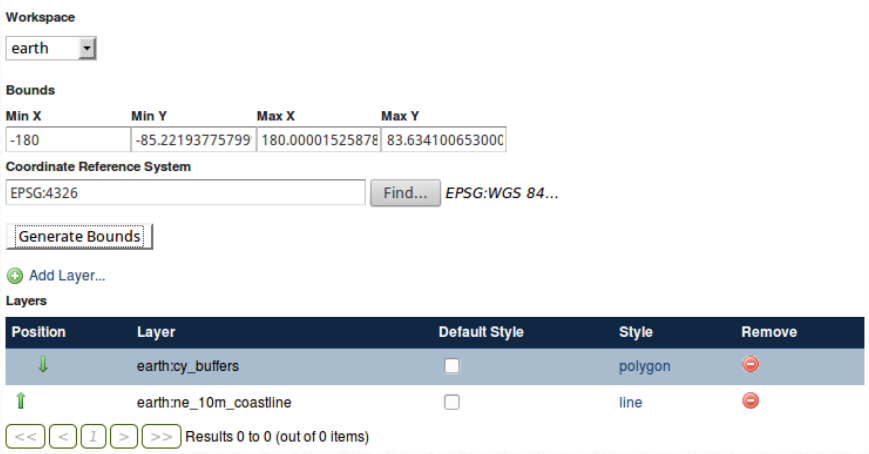
Note: To make it easier to find the appropriate layers, type `geoss` in the search box to narrow the

listing.

Layer order is important. The top layer in the list will be drawn first, the bottom last. Make sure to match the order of the above list. Reorder the layers if necessary by clicking on the Position arrows for each layer.

- Check the **Default Style** box for all the layers.

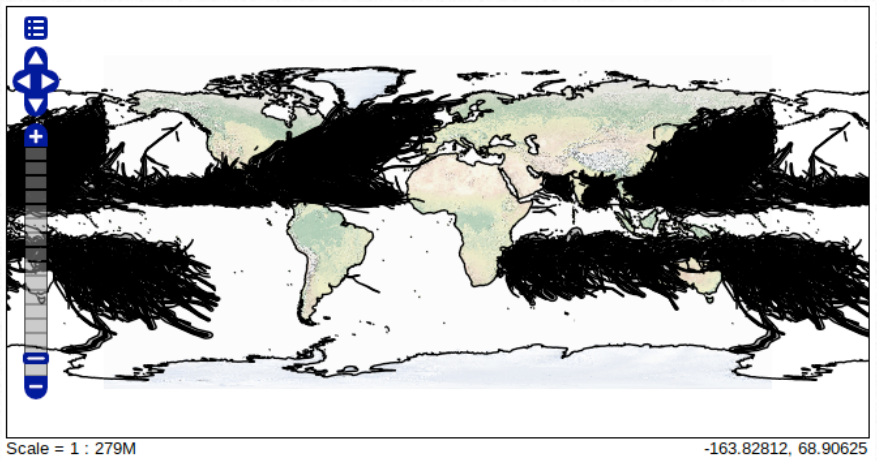
- Now click the **Generate Bounds** button to determine the bounding box for the entire layer group. This button will also determine the projection of the layer group, equal to the top layer's projection by default. If the projection is not found automatically, enter EPSG:4326.



- Click **Save** when done.

- Preview the layer group by going to the **Layer Preview**. This might take a few seconds before displaying all layers:

Note: Lists in GeoServer are paged at 25 items at a time. If you can't find the layer, you may need to click the [2] or [>] buttons.



Even though the Layer Importer generated unique styles for each layer, this layer group does not look very nice. The following section will discuss the next important step of making maps: styling.

3. Styling

GeoServer can render geospatial data as images and return them for viewing in a browser. This is the heart of WMS. However, geospatial data has no inherent visualization. Therefore additional information, in the form of a style, needs to be applied to data in order to visualize it.

GeoServer uses the Styled Layer Descriptor (SLD) markup language to describe geospatial data. In this section, we will first explain basic SLD syntax and then show how to create and edit styles manually in GeoServer. Finally, we will introduce Styler, a graphical style editor.

3.1. Styled Layer Descriptor (SLD)

GeoServer uses Styled Layer Descriptor (SLD) to visualize geospatial data. SLD is a XML-based standard created by the Open Geospatial Consortium (OGC). For more information on the SLD

schema, please see the OGC page on Styled Layer Descriptor at http://www.opengeospatial.org/standards/sld

3.1.1. SLD structure

An SLD file contains the following hierarchical structure:

Header

FeatureTypeStyles

Rules

Symbolizers

The header of the SLD contains metadata about XML namespaces, and is usually identical among different SLDs. The details of the header are beyond the scope of this workshop.

A FeatureTypeStyle is a group of styling rules. (recall that a feature type is another word for a layer.) Grouping by FeatureTypeStyle affects rendering order; the first FeatureTypeStyle will be rendered first, followed by the second, etc.

A Rule is a single styling directive. It can apply globally to a layer, or it can have filter logic associated with it so that the rule is conditionally applied. These conditions can be based on the attributes of the data, or based on the scale (zoom) level of the data being rendered.

A Symbolizer is the actual style instruction. There are five types of symbolizers:

- PointSymbolizer
- LineSymbolizer
- PolygonSymbolizer
- RasterSymbolizer
- TextSymbolizer

There can be one or more FeatureTypeStyles per SLD, one or more Rules per FeatureTypeStyles, and one or more Symbolizers per Rule.

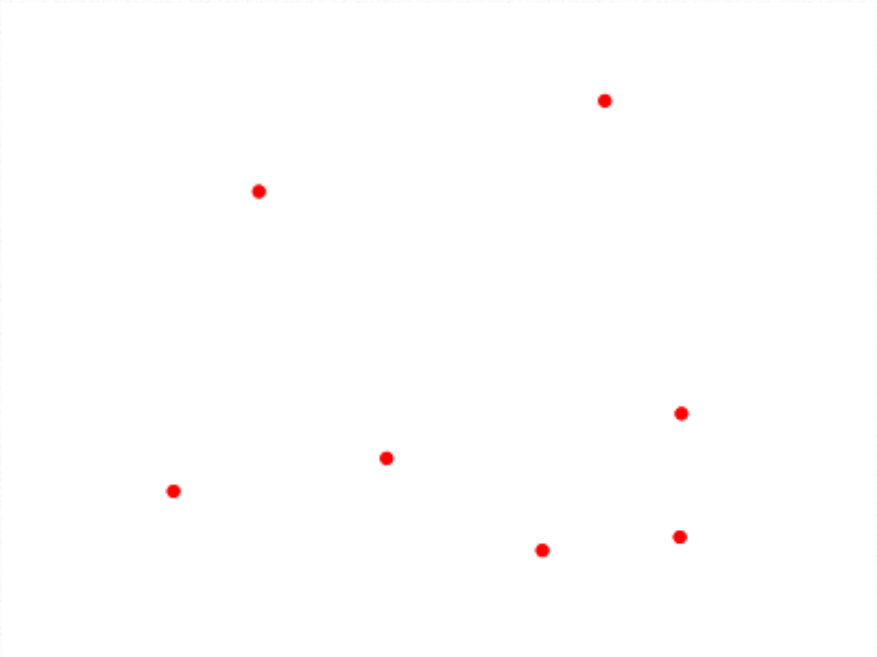
3.1.2. Simple SLD

The following example draws a simple 6-pixel red circle for each data point.

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <StyledLayerDescriptor version="1.0.0"
3   xsi:schemaLocation="http://www.opengis.net/sld StyledLayerDescriptor.xsd"
4   xmlns="http://www.opengis.net/sld"
5   xmlns:ogc="http://www.opengis.net/ogc"
6   xmlns:xlink="http://www.w3.org/1999/xlink"
7   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
8    <NamedLayer>
9      <Name>Simple Point</Name>
10     <UserStyle>
11       <Title>Simple Point</Title>
12       <FeatureTypeStyle>
13         <Rule>
14           <PointSymbolizer>
15             <Graphic>
16               <Mark>
17                 <WellKnownName>circle</WellKnownName>
18                 <Fill>
19                   <CssParameter name="fill">#FF0000</CssParameter>
20                 </Fill>
21               </Mark>
22               <Size>6</Size>
23             </Graphic>
24           </PointSymbolizer>
25         </Rule>
26       </FeatureTypeStyle>
27     </UserStyle>
28   </NamedLayer>
29 </StyledLayerDescriptor>
```

The first 11 lines are the header, which contain XML namespace information, as well as the Name and Title of the SLD. The actual styling happens inside the <FeatureTypeStyle> tag (lines 12-26), of which there is only one in this example. The tag contains one <Rule> (lines 13-25) and the rule contains one symbolizer, a <PointSymbolizer> (lines 14-24). The symbolizer directive creates a graphic mark of a “well known name”, in this case a circle (line 17). This shape has a <Fill> parameter of #FF0000 (line 19), which is an RGB color code for 100% red. The shape also has a <Size> of 6 (line 22), which is the diameter of the circle in pixels.

When applied to a hypothetical layer, the result would look like this:



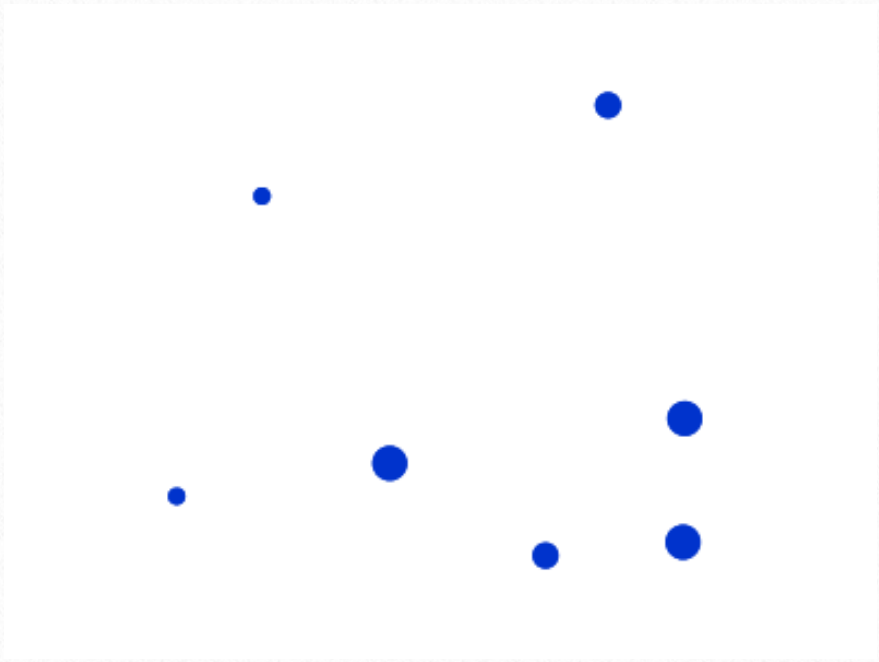
3.1.3. Another SLD example

Here is an example of an SLD that includes attribute-based styling. The SLD also contains three rules. Each rule has an attribute-based condition, with the outcome determining the size of the shape being rendered. The attribute in question is called “pop”, and the three rules are “less than 50000“, “50000 to 100000“, and “greater than 100000“. The result is a blue circle with a size of 8, 12, of 16 pixels, depending on the rule.

Rule name	Population	Size
SmallPop	< 50,000	8
MediumPop	50.000 to 100.000	12
LargePop	> 100.000	16

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<StyledLayerDescriptor version="1.0.0"
 xsi:schemaLocation="http://www.opengis.net/sld StyledLayerDescriptor.xsd"
 xmlns="http://www.opengis.net/sld"
 xmlns:ogc="http://www.opengis.net/ogc"
 xmlns:xlink="http://www.w3.org/1999/xlink"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <NamedLayer>
    <Name>Attribute-based point</Name>
    <UserStyle>
      <Title>Attribute-based point</Title>
      <FeatureTypeStyle>
        <Rule>
          <Name>SmallPop</Name>
          <Title>1 to 50000</Title>
          <ogc:Filter>
            <ogc:PropertyIsLessThan>
              <ogc:PropertyName>pop</ogc:PropertyName>
              <ogc:Literal>50000</ogc:Literal>
            </ogc:PropertyIsLessThan>
          </ogc:Filter>
          <PointSymbolizer>
            <Graphic>
              <Mark>
                <WellKnownName>circle</WellKnownName>
                <Fill>
                  <CssParameter name="fill">#0033CC</CssParameter>
                </Fill>
              </Mark>
              <Size>8</Size>
            </Graphic>
          </PointSymbolizer>
        </Rule>
        <Rule>
          <Name>MediumPop</Name>
          <Title>50000 to 100000</Title>
          <ogc:Filter>
            <ogc:And>
              <ogc:PropertyIsGreaterThanOrEqualTo>
                <ogc:PropertyName>pop</ogc:PropertyName>
                <ogc:Literal>50000</ogc:Literal>
              </ogc:PropertyIsGreaterThanOrEqualTo>
              <ogc:PropertyIsLessThan>
                <ogc:PropertyName>pop</ogc:PropertyName>
                <ogc:Literal>100000</ogc:Literal>
              </ogc:PropertyIsLessThan>
            </ogc:And>
          </ogc:Filter>
          <PointSymbolizer>
            <Graphic>
              <Mark>
                <WellKnownName>circle</WellKnownName>
                <Fill>
                  <CssParameter name="fill">#0033CC</CssParameter>
                </Fill>
              </Mark>
              <Size>12</Size>
            </Graphic>
          </PointSymbolizer>
        </Rule>
        <Rule>
          <Name>LargePop</Name>
          <Title>Greater than 100000</Title>
          <ogc:Filter>
            <ogc:PropertyIsGreaterThanOrEqualTo>
              <ogc:PropertyName>pop</ogc:PropertyName>
              <ogc:Literal>100000</ogc:Literal>
            </ogc:PropertyIsGreaterThanOrEqualTo>
          </ogc:Filter>
          <PointSymbolizer>
            <Graphic>
              <Mark>
                <WellKnownName>circle</WellKnownName>
                <Fill>
                  <CssParameter name="fill">#0033CC</CssParameter>
                </Fill>
              </Mark>
              <Size>16</Size>
            </Graphic>
          </PointSymbolizer>
        </Rule>
      </FeatureTypeStyle>
    </UserStyle>
  </NamedLayer>
</StyledLayerDescriptor>
```

It is helpful to break the SLD down into components when it gets large. There are three rules in this style, all of which are contained inside of one FeatureTypeStyle. Looking at the first rule (lines 13-33), there is a filter tag (<ogc:Filter>). This filter specifies that if the attribute value of “pop” for a given feature is less than 50000, then the condition is true and the feature is displayed. The second rule (lines 34-60) has a compound filter that specifies that the attribute value must be both greater than or equal to 50000 and less than 100000 in order for the feature to be rendered. Finally, the third rule (lines 61-77) has a filter that specifies that the attribute value must be greater that or equal to 100000 in order for the feature to be rendered.



3.1.4. SLD Cookbook

The GeoServer documentation (available at http://docs.geoserver.org) contains a collection of styles called the SLD Cookbook, designed for those wishing to learn SLD, or those who want a quick reference. The SLD Cookbook is available at http://docs.geoserver.org/stable/en/user/styling/sld-cookbook/index.html. The above SLD examples were taken from the SLD Cookbook.

3.2. Styles in GeoServer

Every layer published in GeoServer must have a style associated with it. When manually loading layers as done in the Publishing a shapefile and Publishing a GeoTIFF sections, GeoServer will look at the geometry of the data and assign a generic style based on that data type. When using the Loading multiple layers, a unique style will be generated for each layer. We will now look at how GeoServer handles styles.

3.2.1. Viewing existing style

- Preview the **geoss:cy_buffers** layer to see its visualization by navigating to the **Layer Preview**, then clicking on the **OpenLayers** link next to that layer.

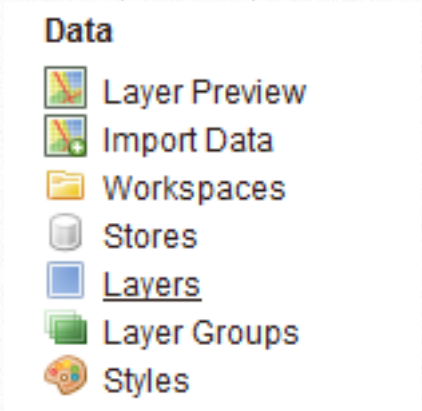
- **Leave this preview window open**.
- In **another tab of your browser** go back to the main **GeoServer Web Administration page**.

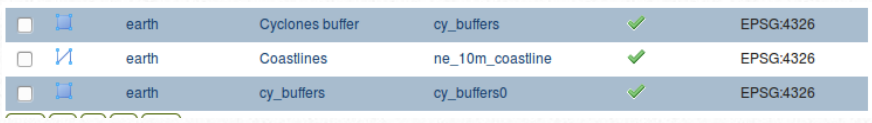
Note: Remember that the GeoServer web admin page is typically available at `http://localhost:8080/geoserver`

In order to view the SLD for this layer, we need to find out which style is associated with this layer.

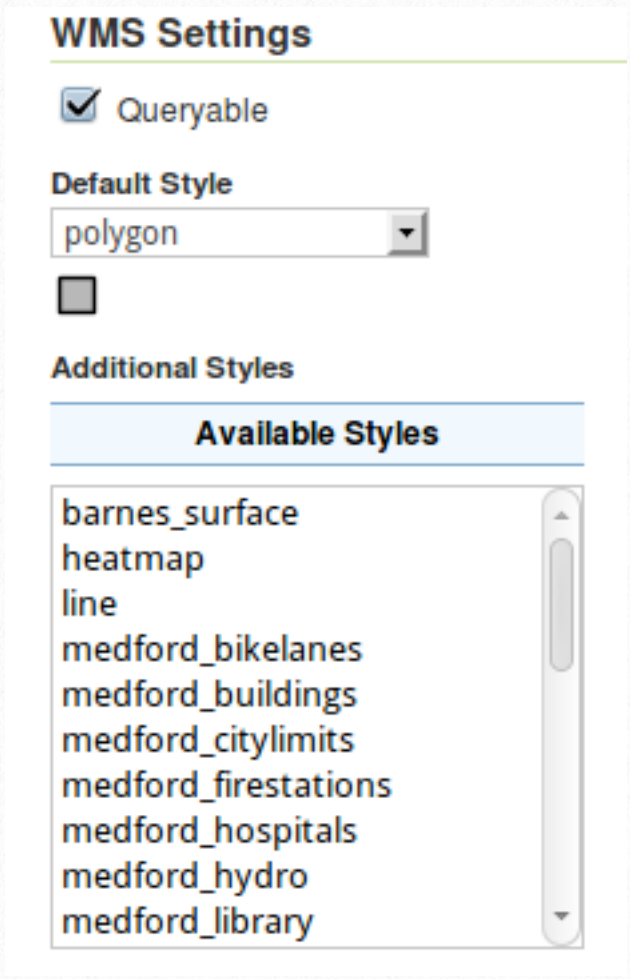
- To do so, click on **Layers** under `Data` on the left side of the page, then click on the Layer Name **cy_buffers**.

Note: If your session has been inactive for a while, you may need to log in again.



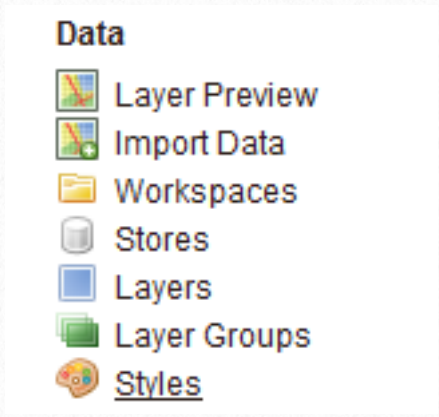


- You are now back at the layer configuration page. Notice there are four tabs on this page, `Data` (the default), `Publishing`, `Dimensions` and `Tile caching`. Click on the **Publishing** tab, then scroll down to the entry that says Default Styles. Note that the current style for the cyclone buffers is a dark grey `polygon`.



Now that we know the name of the style, we can view the style's code.

- Click on the **Styles** link, under `Data` on the left side of the page:



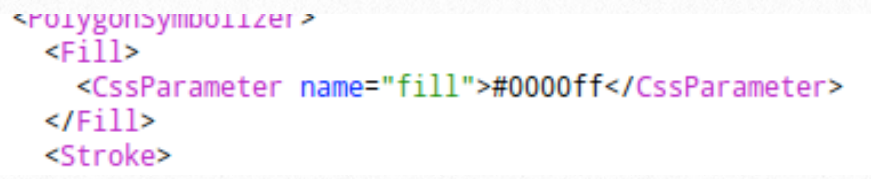
- Click on the style name as determined above: **polygon**.

A web-based text editor will open up, displaying the SLD code for this style:

3.2.2. Editing existing style

It is helpful when learning about SLD to edit existing styles rather than creating new ones from scratch. We will now do this with the style that is open in the web-based text editor.

- Make a change to an RGB color value in a <CssParameter> value. For example, find the line that starts with <CssParameter name="fill"> and change the RGB code to #**0000ff** (blue).



- When done, click **Validate** to make sure that the changes you have made are valid (there should be some text highlighted in green at the top of the page). If you receive an error, go back and check your work.

- Click **Submit** to commit the style change.
- Now go back to the browser tab that contains the **OpenLayers preview map**.
- **Refresh** the page, and you should see the color change to blue:

Note: GeoServer and your browser will sometimes cache images. If you don't see a change immediately, zoom or pan the map to display a new area.

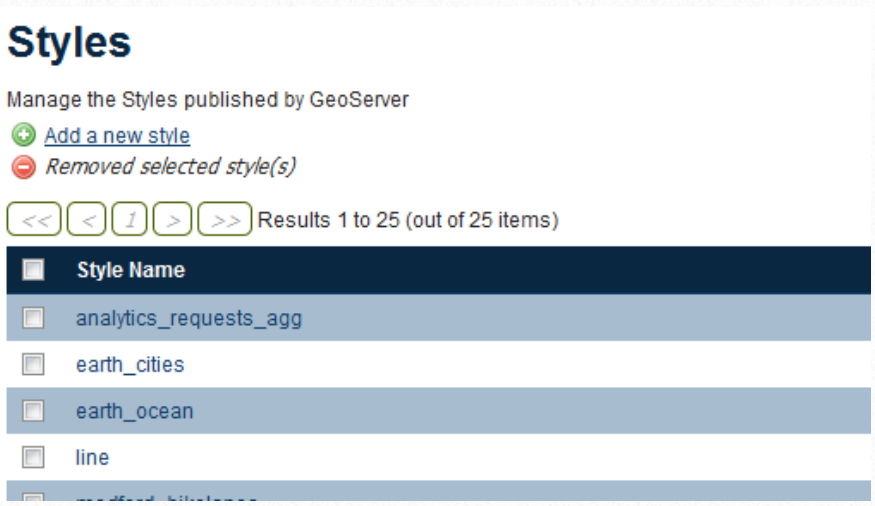
Warning: It is also important to note that it is easy (and dangerous!) to modify GeoServer default styles.

3.2.3. Loading new styles

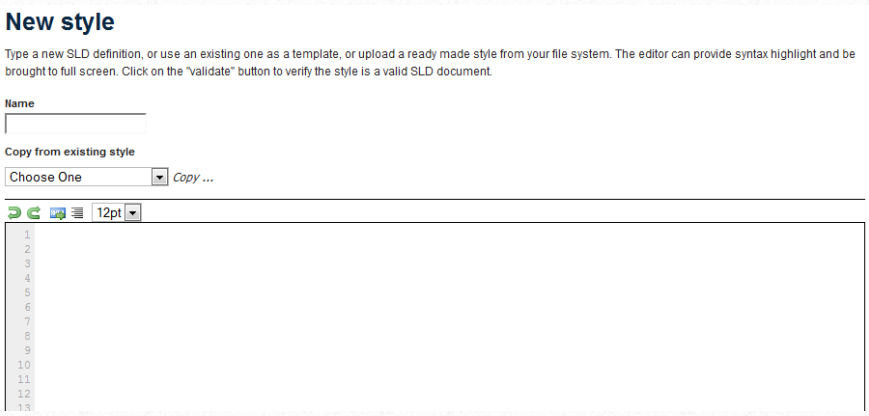
If you have an SLD saved as a text file, it is easy to load it into GeoServer. We will now load the styles saved in the workshop styles folder.

- Navigate back to the **Styles** page by clicking on Styles under `Data` on the left side of the page.

- Click on **Add a new style**.



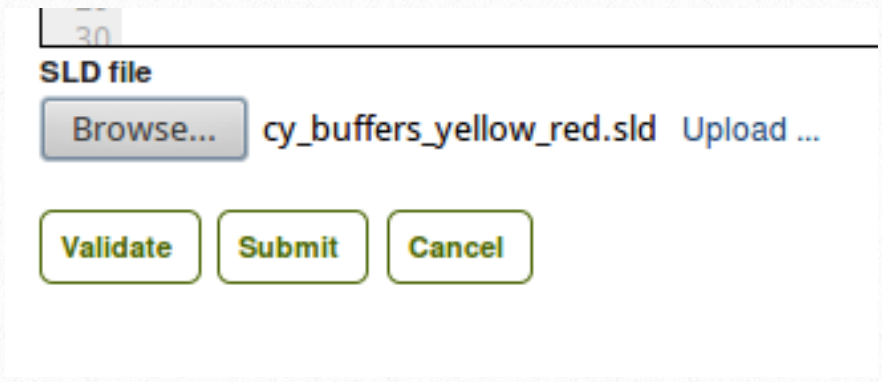
A blank text editor will open.



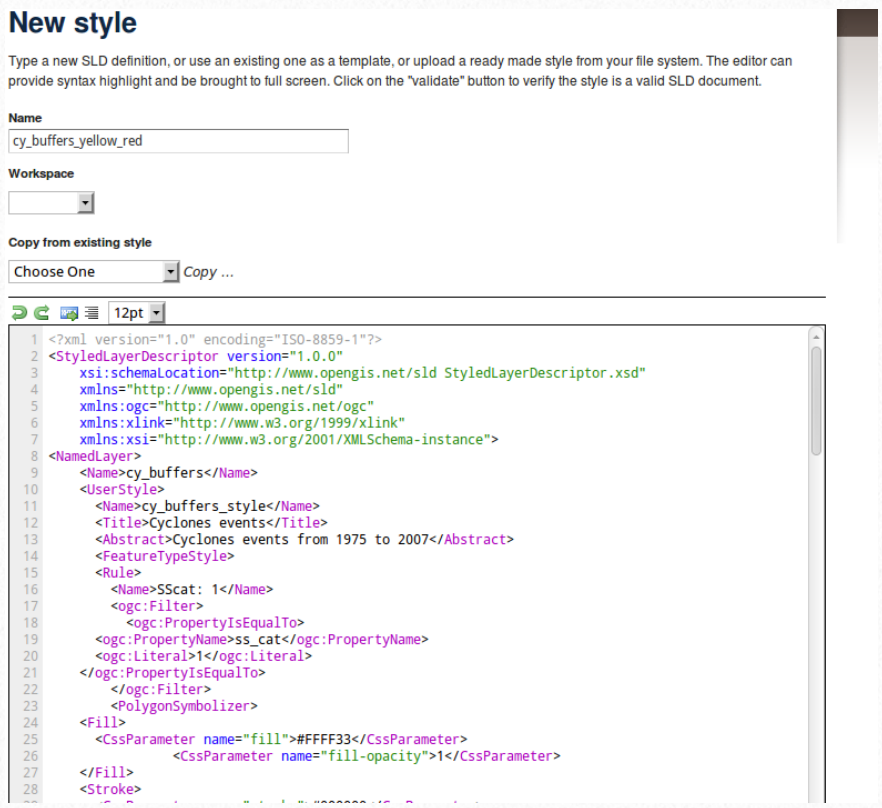
- At the very bottom of the page, below the text editor, there is a form named SLD file. Click **Browse...** to navigate to and select **`/Home/geoss/Desktop/data_exercises/cy_buffers_yellow_red.sld`**

This style has been prepare for you and will allow rendering cyclones buffers (1) in a more attractive way and (2) according to variations of an attribute.

- Click **Upload...** to load this SLD into GeoServer.



The SLD will display in the text editor. The name of the style will be automatically generated, although you can change this if you'd like.



- Click **Validate** to ensure that the SLD is valid, select the **geoss** workspace and click **Submit**.

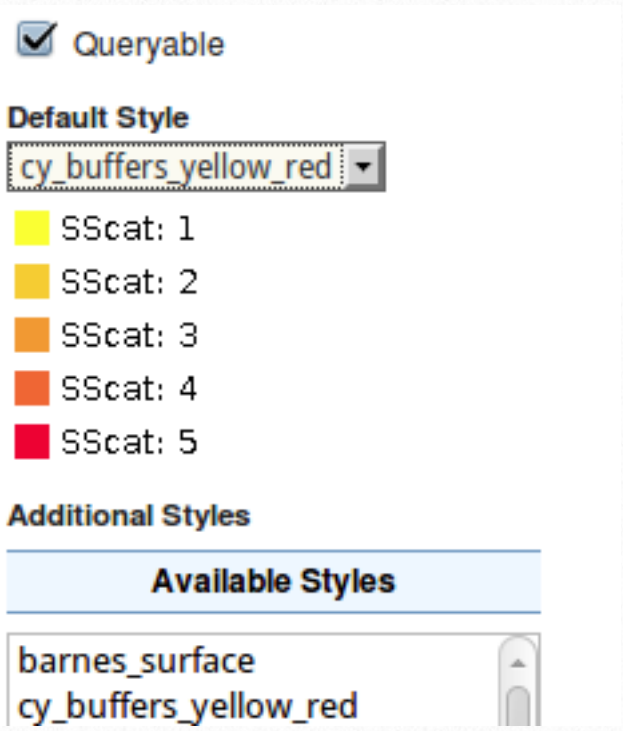
We will leave the coastlines layer with the default style as well as the shaded relief image.

Associating styles with layers

Once the styles are loaded, you can associate them with existing layers. That is what you are going to do now with cyclones buffers.

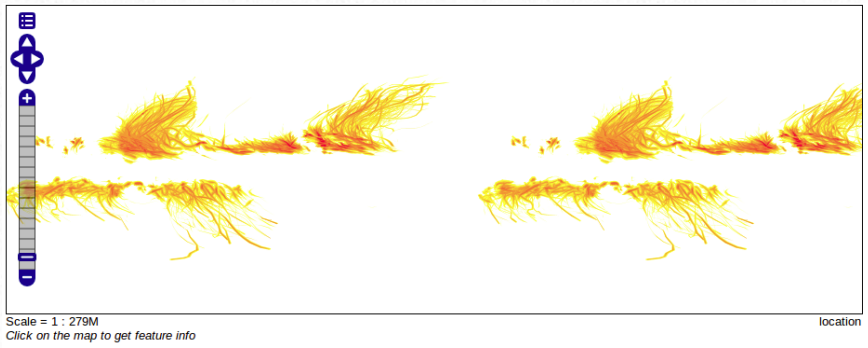
Warning: If an SLD has references that are specific to a certain layer (for example, attribute names or geometries), associating that style with another layer may cause unexpected behavior.

- Navigate to the `Layers` page by clicking on **Layers** under `Data` on the left side of the page.
- Click on the **geoss:cy_buffers** layer to edit its configuration.
- Click on the **Publishing** tab
- Scroll down to the `Default style` drop down list. Change the entry to display the **cy_buffers_yellow_red** style. you should notice the legend change.



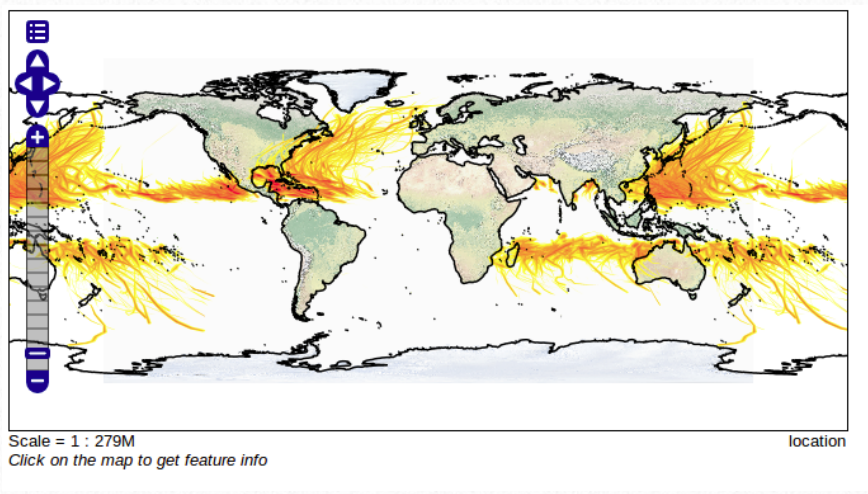
Cyclones events are now displayed in 5 categories, depending on the values stored in the `SScat` attribute.

- Click **Save** to commit the change.
- **Verify the change** by going to the layer's Layer Preview page. Zoom in the see the behavior change based on zoom level.



3.2.4. Revisiting the layer group

You can now preview your layer group (made in the Creating a layer group section) by going to Layer Preview. It should look quite different now!



Note: If for some reason, the layer group fails to update with the new styles, go back the Layer Group page and verify that the Default Style box is checked for every layer.

4. GeoExplorer

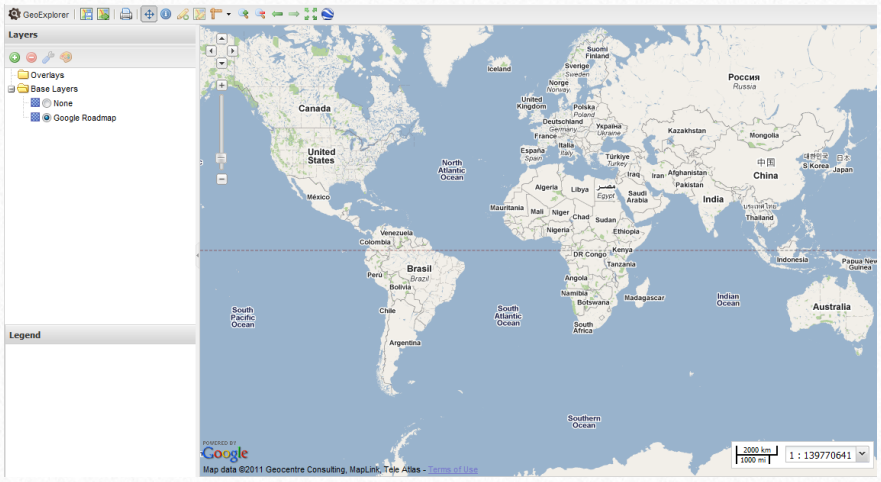
Creating SLD files by hand can be a difficult and time-consuming process. Fortunately, there is a tool called GeoExplorer which is a graphical style editor. With GeoExplorer, you can create rules and symbolizers without ever needing to view SLD code.

GeoExplorer replaces the deprecated Styler application. All the functionality of Styler has been added to GeoServer, making for a simpler edit/view workflow.

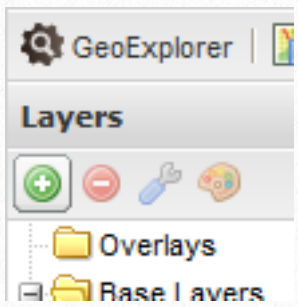
Note: GeoExplorer does not currently implement all of the features of the SLD specification. In addition, GeoServer "vendor options" are not yet supported.

4.1. Using GeoExplorer

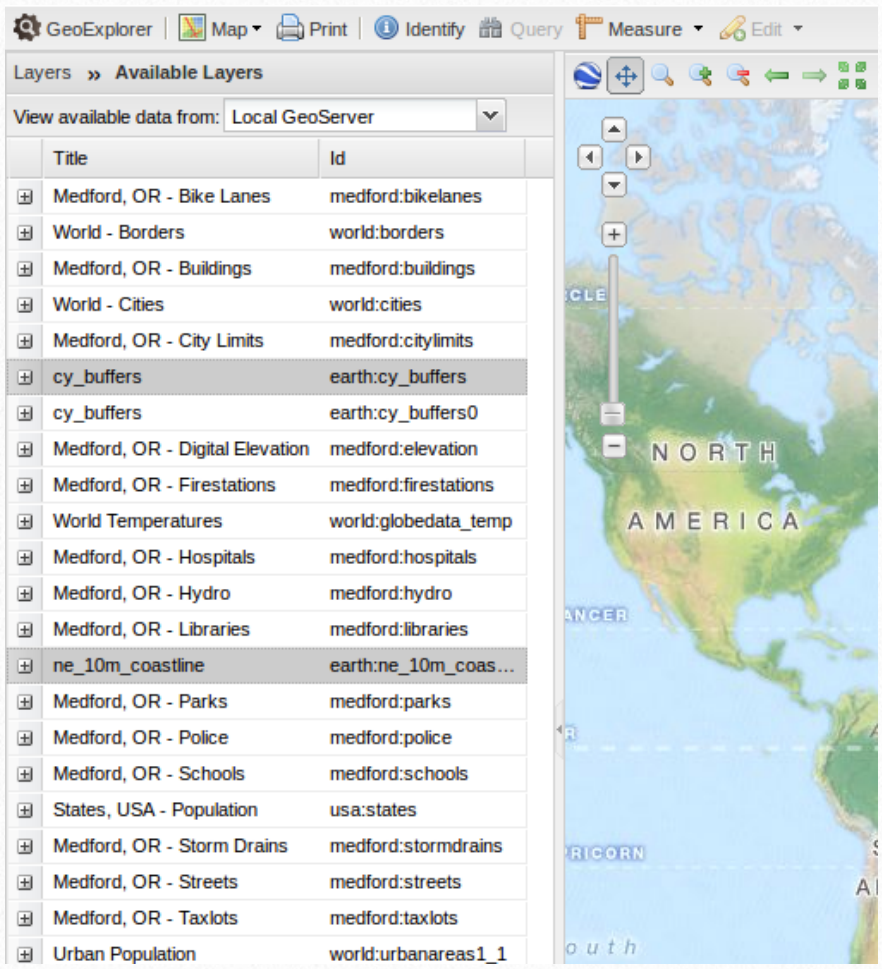
- **Launch GeoExplorer**. By default, GeoExplorer is located at `http://localhost:8080/geoexplorer`.



- By default, the only layer that is displayed is an OpenStreetMap base layer. Click the green plus icon the top left of the screen to add layers.



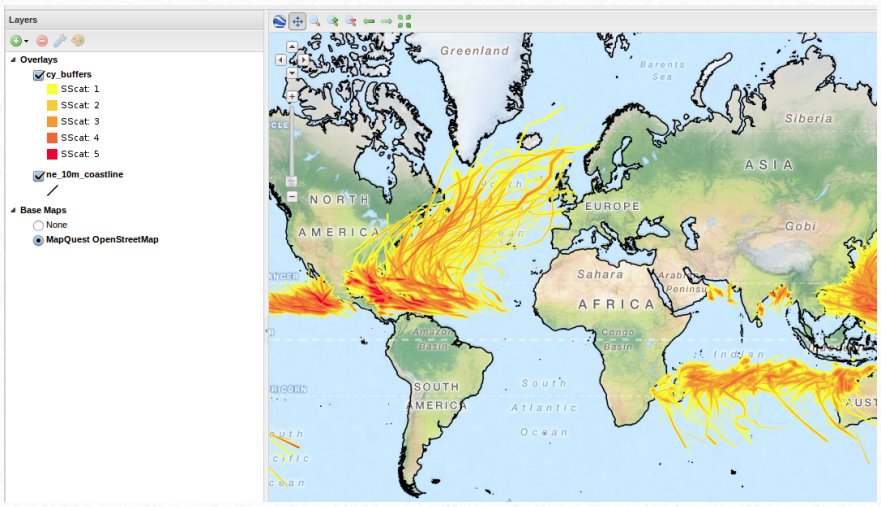
- Select the two vector layers used in this workshop and click **Add layers** at the bottom of the page:



- Click **Done** to return to the main map.

The check boxes determine which layers are being viewed (with the order of the layers determining the rendering order). The bottom left corner holds the Legend, which is a list of all the Rules in the styles of the visible layer(s). Finally, the bulk of the window is taken up by the map itself.

Note: Layer groups cannot be styled with GeoExplorer.

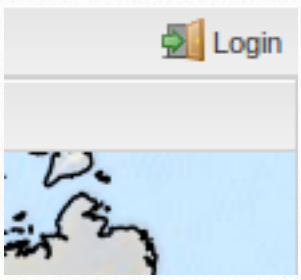


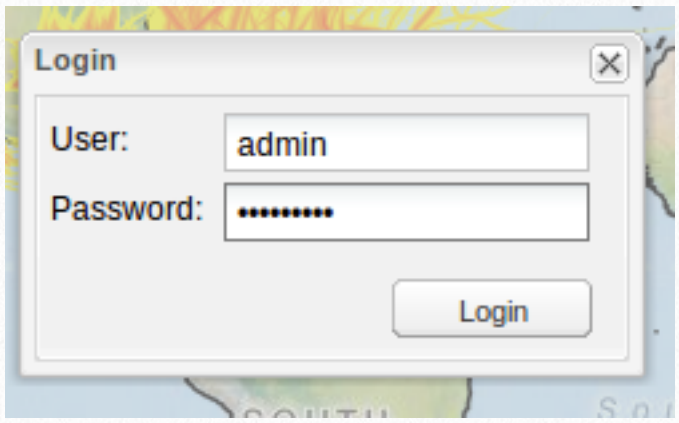
4.2. Editing an existing style

Warning: GeoExplorer makes changes directly to an SLD, and will completely recreate the SLD file. An SLD file may look very different after being edited by GeoExplorer. It is always a good idea to make a backup copy of your SLDs before using GeoExplorer.

Before we can make any changes to styles, we have to log in to GeoExplorer.

- Click the **login** button at the very top right of the window and enter your GeoServer admin credentials: **admin** / **geoserver** .



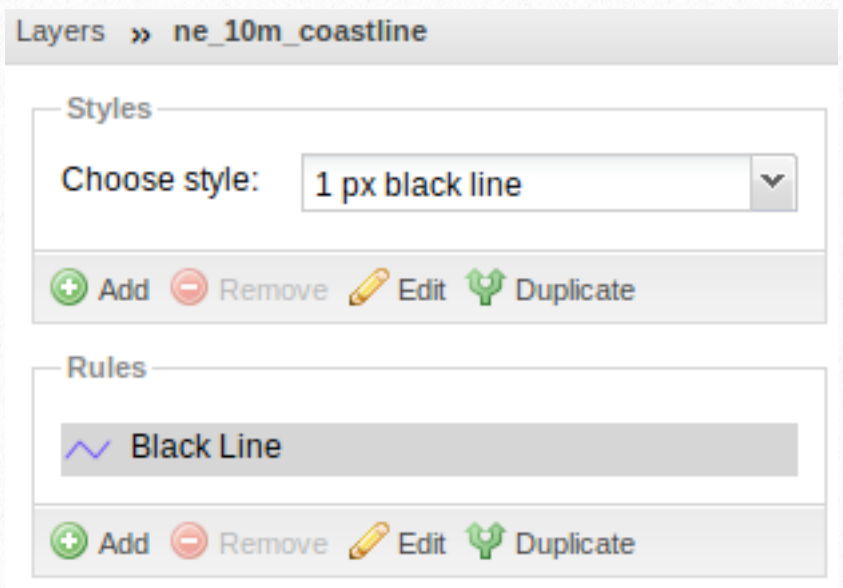


Once logged in, the editing tools will be enabled.

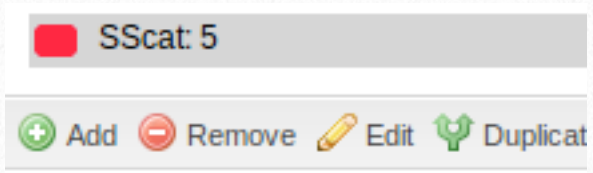
- Select the **ne_10m_coastlines** layer by single clicking on it in the layer list. Then click on the **palette** icon right above the layer list to Edit Styles:

Note: If the icon is disabled, make sure that you have logged in successfully and that you have selected the correct layer.

- Click **Black Line**:



- Click **Edit at the bottom**:



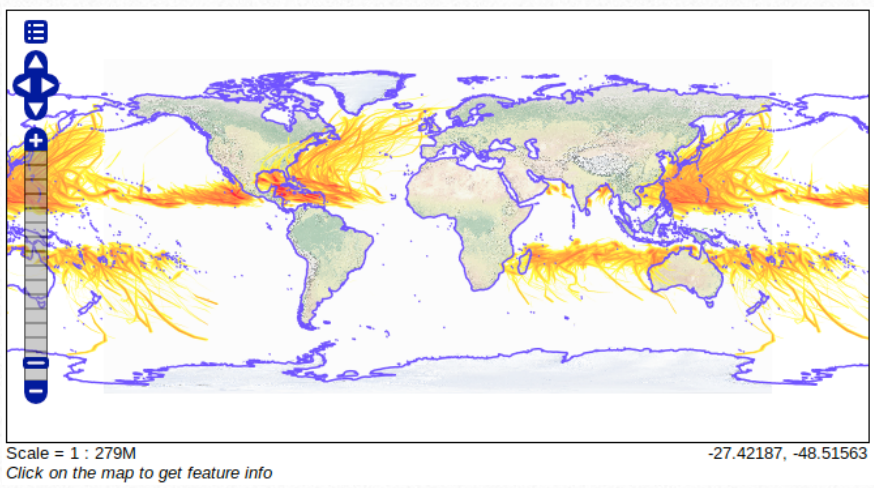
A style editor will display.

- Here you can edit the style. Make the following edit to the rule and see how it updates in real time. For example **enter #5C33FF**

- Once you have modified the style of the category, click on the **Save** button:

You can do the same for the other categories.

If you go back to Layers preview window, you will see the changes that you just did (in the picture below coastlines are purple; they were black before):



You can take a look at the SLD code that GeoExplorer has created in the style you just created (the one with the style's name and the number at the end). For that you can navigate back to GeoServer, to the `Styles` menu, and then to the `line` style to view changes: the <CSSParameter name="stroke"> value is now **#5C33FF**.

If you have time you can make edits to the cy_buffers categories. For example select the 5th category (red) and change its color.

- You can now **quit GeoExplorer** (no need to save the map composition).

5. Google Earth

Google Earth is a powerful 3D map viewer. GeoServer integrates with Google Earth by providing native KML output, allowing any layer served by GeoServer to be loaded into Google Earth. In addition, there are additional visualization features that are made possible through Google Earth, such as filters, legends, and extrudes.

5.1. Viewing layers

GeoServer natively outputs data in KML (Keyhole Markup Language). This is the markup language that is used by Google Maps and Google Earth. In this way, it is easy to convert shapefiles or any geospatial data to a format that Google understands.

There are two ways to view data in Google Earth. The first is by statically loading a KML file. The second is by using a Network Link and connecting to a KML stream.

For this exercise, we will use a new layer showing the Large Marine Areas called **lmz_po_shp**

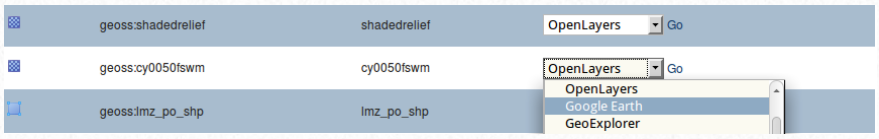
that you will find in the `/home/geoss/Desktop/data_exercises/vector` folder.

- **Publish it in geoserver** as you have learnt to do in the previous sections.

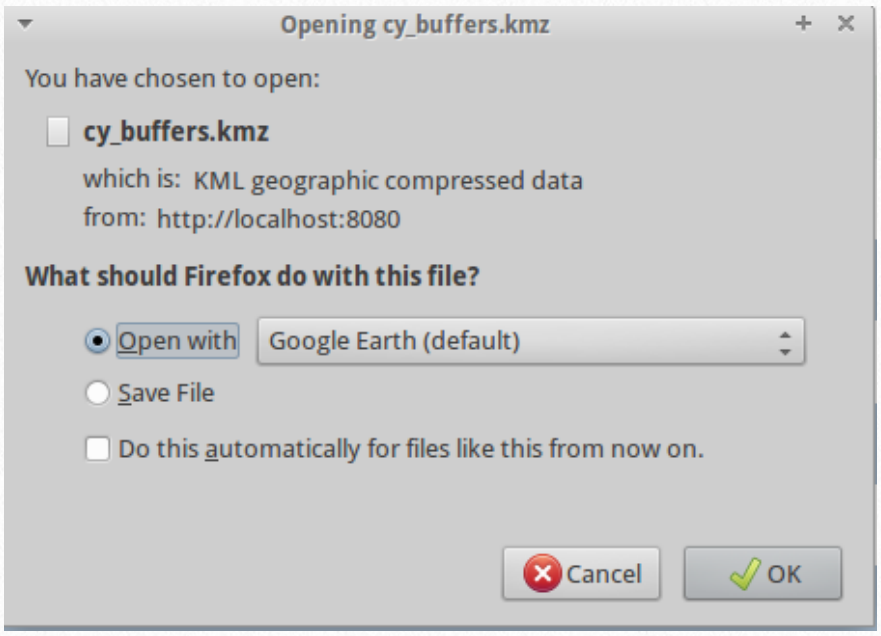
5.2. Loading a KML file

- Navigate to the **Layer Preview** menu.

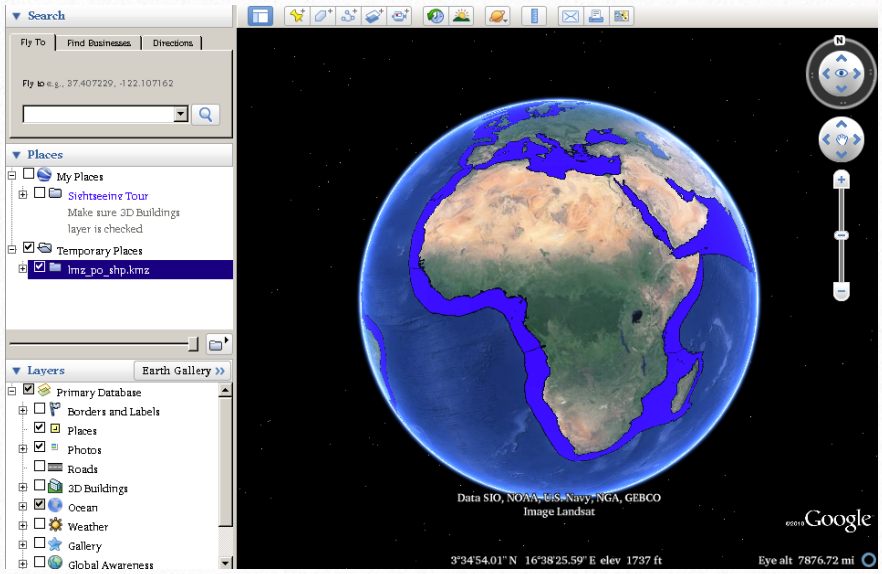
- Select the **geoss:lmz_po_shp** layer, select **Google Earth** in the dropdown list and click **Go**.



- You will be asked to download a file. Select **Open with Google Earth** and click **OK**.

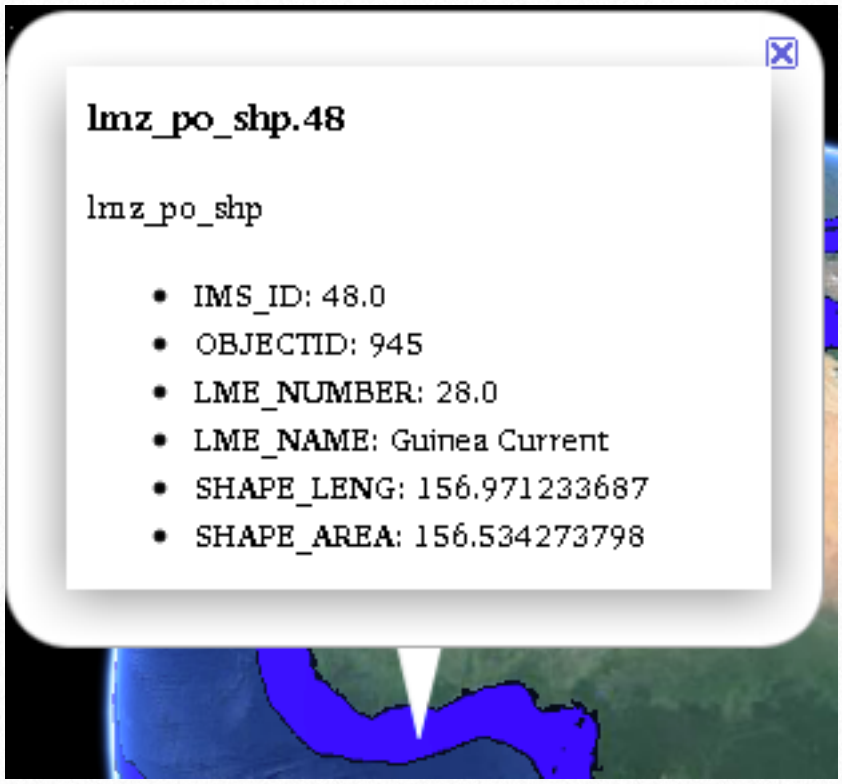


- The layer will open in Google Earth.



Large marine areas might be difficult to visualize if they are in blue color.

- **Click** on **one of the layer's points** to view its placemark description.



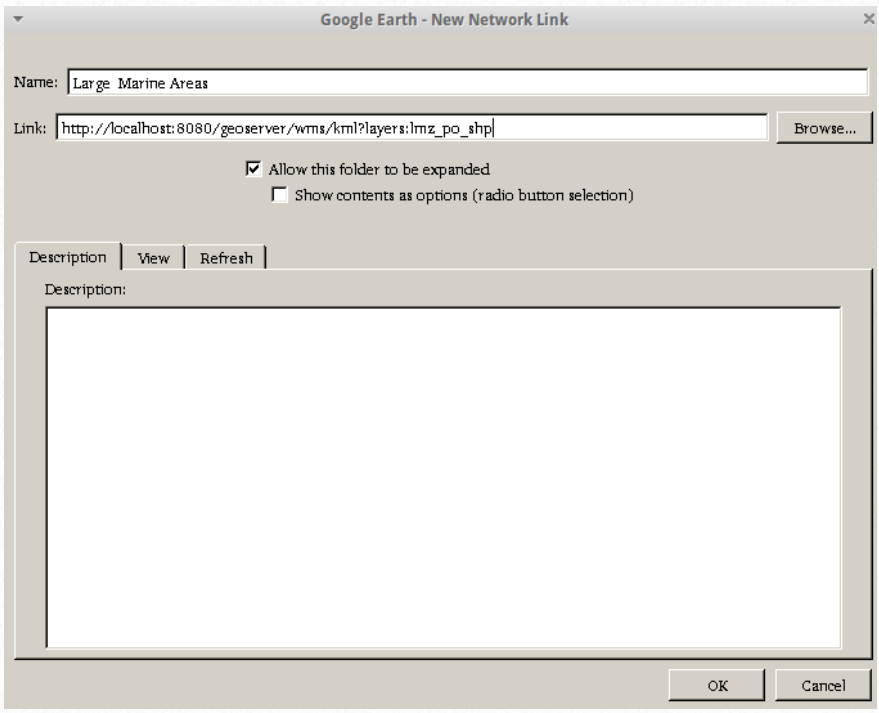
If is possible to customize this placemark description through GeoServer, including adding custom HTML.

5.3. Network Link

Now we will connect Google Earth to a GeoServer KML stream via a Network Link. This

allows for the view in Google Earth to be dynamically updated.

- Remove the entry called **lmz_po_shp.kmz** in the Places list (**Right-click > delete**).
- Add a new Network Link by navigating to the **Add** menu and selecting **Network Link**.
- In the dialog that appears, enter **Large Marine Areas** in the `Name` field.
- In the `Link` field, enter the following URL: **http://localhost:8080/geoserver/wms/kml?layers=geoss:lmz_po_shp**



- Click **OK** when done.

The output should be the same as before. The difference is that a Network Link is dynamic, which means that we can alter the stream and refresh the view without having to export a new KML file. We'll put this to use in the next section.

5.4. Filtering layers

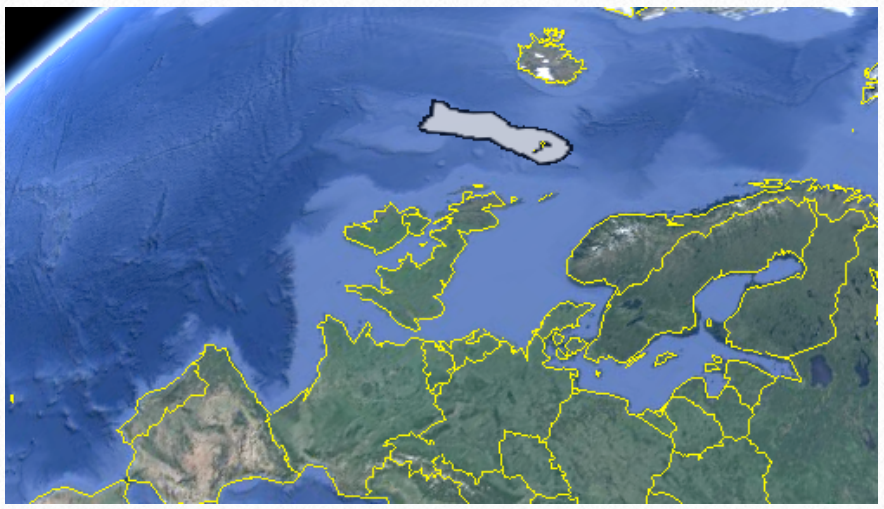
Often a layer contains too much information and it is desirable to filter what is displayed. In this section we will filter a KML stream coming from GeoServer using CQL. The `cql_filter` parameter is a way to specify a predicate based on attribute values or spatial orientation.

- **Right-click Large Marine Areas** and select **Properties**.

- After http://localhost:8080/geoserver/wms/kml?layers=geoss:lmz_po_shp add:

&cql_filter=LME_NAME='Faroe Plateau'

- Click **OK**:



You will see that only the marine area around Faroe Island in North-Atlantic will show up.

- You can now **close Google Earth**. When prompted to save, choose **Discard**.

4

How to document and search data?

Keywords: CSW; GeoNetwork; ISO19115; ISO19139; metadata; XML

What you will learn:

- To create an online catalog of metadata
- To import existing metadata into the catalog
- To harvest metadata from remote GeoNetwork nodes
- To manage users' privileges on the metadata
- To perform a CSW query
- To search and visualize metadata

How to create an online catalog of metadata?

In this section, we will learn how to create online metadata through a widely used online metadata catalog called GeoNetwork. GeoNetwork is a catalog application to manage spatially referenced resources. It provides powerful metadata editing and search functions as well as an embedded interactive web map viewer. It has been developed to connect spatial information communities and their data using a modern architecture, which is at the same time powerful and low cost, based on the principles of Free and Open Source Software (FOSS) and International and Open Standards for services and protocols.

GeoNetwork can be accessed through a web browser.

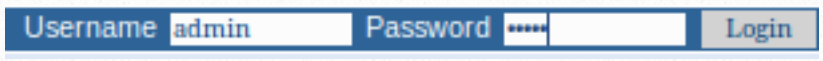
- **Open the web browser** e.g. by double-clicking the **start.html** file on the desktop and selecting **GeoNetwork** in the **Tools** section.

Note: you can also enter http://localhost:8080/geonetwork in your web browser.

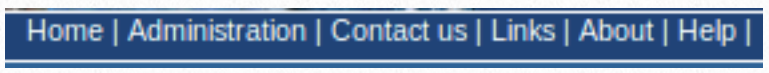
The GeoNetwork main page shows up.

As you can see on this GeoNetwork homepage, there is a possibility to search for data directly. But in order to enter new metadata records, you need to be logged in.

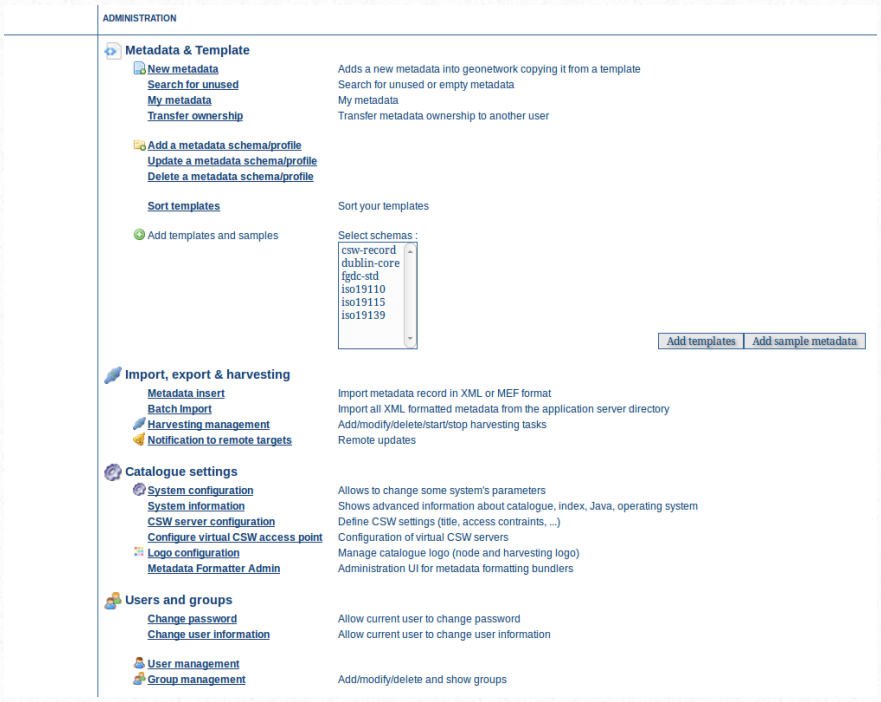
- **Log in** using the **admin** username and **admin** password.



- Click the **Administration** tab:



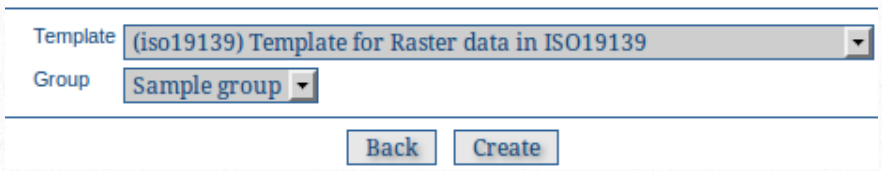
You reach then the `Administration` interface from which you will be able to perform all the operations of the coming sections.



The administration interface is made of several sections (`Metadata & Template`; `Import, export & harvesting`; `Catalogue settings`; `Users and groups`; `Thesauri and classification systems`; `Index settings`; `Localization`), each containing a certain number of functions.

For this exercise, we will create a new online metadata record of the vector layer of the cyclones events for the period 1969-2009 available on the preview website. As this metadata is already existing and for facilitating this process, we have created a file from which you only need to copy/paste the information into the appropriate metadata fields. This file is located under `/home/GEOSS/Desktop/data_exercises/` and is called `cyclones_metadata.txt`.

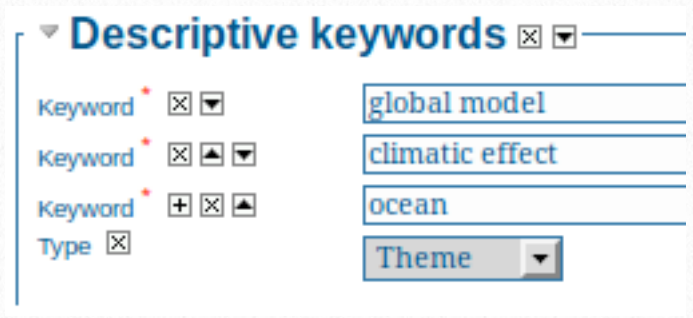
- Open this file with **gedit**.
- In GeoNetwork, click on the **New Metadata** link.
- Select **Template for Vector data in ISO19139 (preferred!)** and the group **Sample group** and click on **Create**



- **Copy/Paste** the various elements **from** the **cyclones_metadata.txt** file **to GeoNetwork**.

We provide values for the most relevant fields in the `metadata_cyclones.txt` file. The mandatory fields are indicated with a red asterisk (*) but you can feel free to add more fields for the exercise.

In some cases you will need to use the '+' button to add multiple entries. For example we ask you to enter 3 descriptive keywords so you can use the '+' button twice to add two keywords:

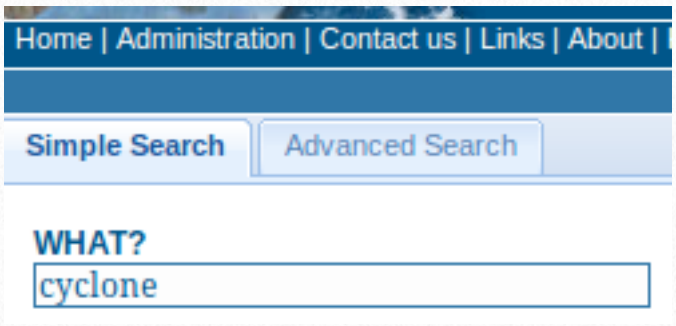


Also note that it is possible to delete fields that are not relevant and not mandatory. For example fields for spatial representation are not necessary for vector data, so they can be removed. To do this you can use the 'x' button in front of the corresponding metadata element.

- Once you have filled the required fields, you simply need to click on the **Save and Close** button.

- In GeoNetwork select the **Home** tab.

- In the `What?` section enter **cyclone**:



- Press **Enter** to validate your search.

A new entry called `Tropical Cyclones Wind-speed Buffers 1969-2009` has been created. If you click on this entry you can check that all the information that you have entered (by copy/paste) are now published on your GeoNetwork.

By default anyone can access this information. You will see in a next section how to configure this entry so that only certain groups can access it. This may be interesting if you publish data with access constraints or confidentiality.

How to import metadata from an XML?

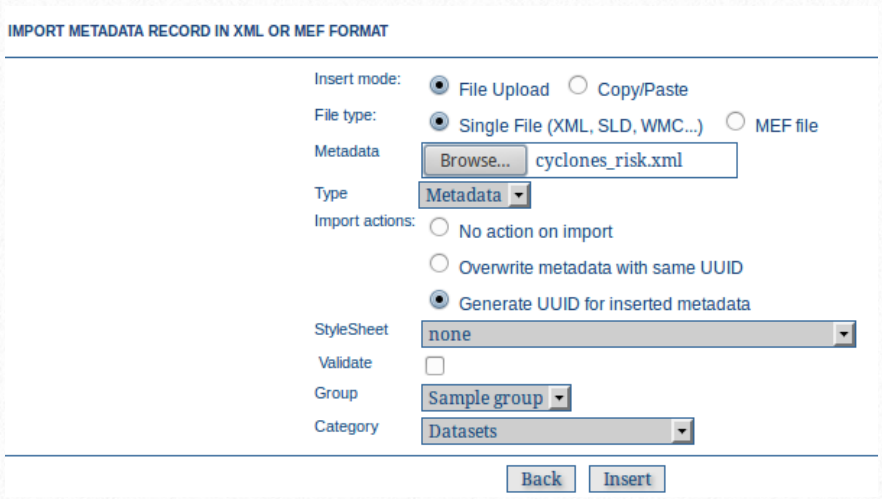
In the previous section we have seen how to create an online metadata record. But in many cases you will need to import existing metadata records into the GeoNetwork online metadata catalog. This will allow you to save a lot of time and avoid re-doing work already performed.

GeoNetwork can import `XML` files containing metadata and directly populate the corresponding fields but the `XML` must follow a metadata standard. Several metadata standards are accepted but for this exercise we will import an `XML` file following the ISO19115 standard. It is the metadata record of the raster layer showing the risk index linked to cyclones: `cyclones_risk.xml`. It is available in the Preview website and we have also downloaded it into the `/home/GEOSS/Desktop/data_exercises/` folder of your machine.

- From GeoNetwork select the **Administration** tab.

- Select **Metadata insert**:

- **Fill the window as** in the screenshot **below**:



Note: to select the `cyclones_risk.xml` file you need to use the `Browse` button.

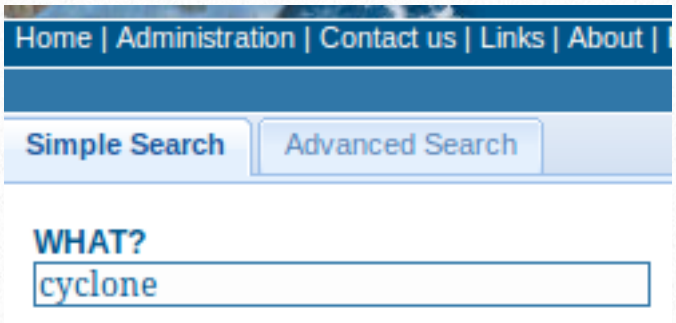
- Click **Insert**.

You should see then a message informing that the metadata has been added.

- Click on the **Edit** button to make sure all the fields have been filled properly.

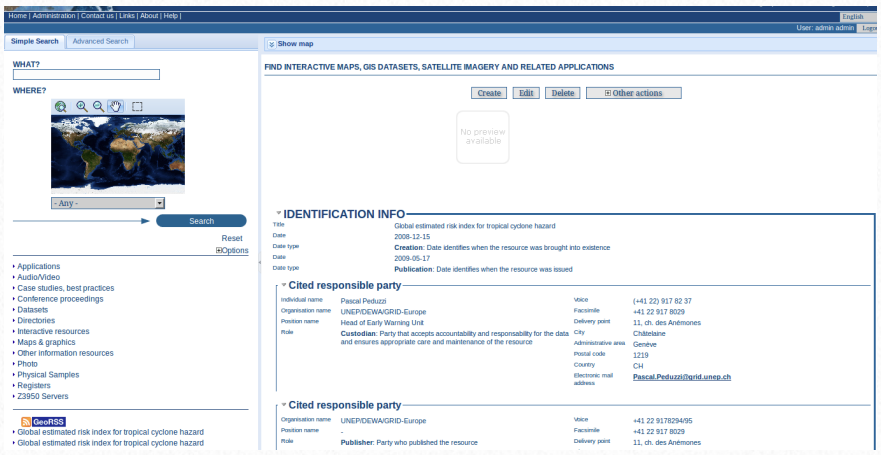
- Go then on the GeoNetwork homepage (**Home** tab).

- In the `What?` section enter **cyclone**:



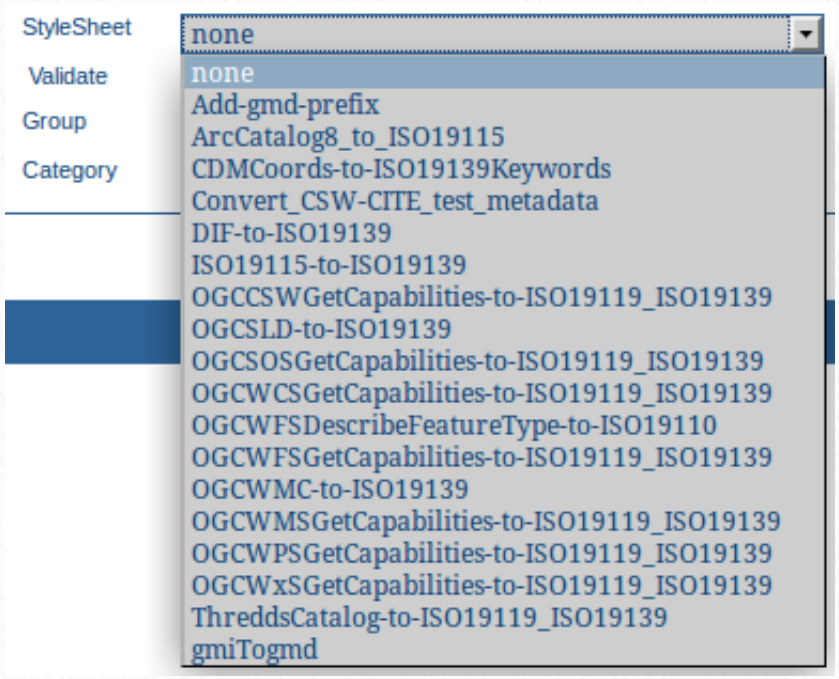
- Press **Enter** to validate your search. You will see that your record appears on the left pane.

- **Click on** this new record (called **Global Estimated Risk Index for Tropical Cyclone Hazard**) and you will see the details of the metadata record in the right pane:



The method used in this section allows to easily import a metadata record if it is properly documented and if it follows a standard. In case you have several such metadata records to import, you can follow exactly the same procedure but using the `Batch import` option from the `Administration` interface.

When importing an `XML` metadata file, you also have the possibility to choose a supported standard to translate it into. The supported standards list displays under the `stylesheet` field of the import.



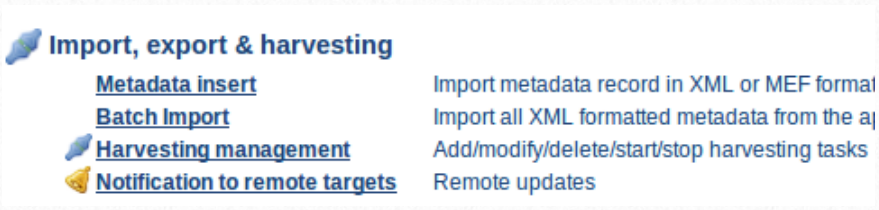
The main disadvantage of importing metadata through `XML` files is that they have to be re-imported regularly if they change with time. For example new cyclone events may be recorded by the original provider while your GeoNetwork will not reflect these edits. In next Section you will learn how to harvest metadata dynamically.

How to harvest metadata from remote geonetwork nodes?

In the two previous sections of this chapter, we have seen how to add or import manually metadata records into the GeoNetwork database. GeoNetwork also offers the possibility to automatically populate its metadata records from other online metadata sources. The GeoNetwork administrator simply needs to define what these other sources are and the frequency of this synchronization. This is called harvesting, which is "the process of collecting metadata from a remote source and storing it locally in GeoNetwork for fast searching via Lucene. This is a periodic process to do, for example, once a week. Harvesting is not a simple import: local and remote metadata are kept aligned" : see http://geonetwork-opensource.org/manuals/2.10.0/eng/users/managing_metadata/harvesting/index.html.

Here is the procedure to harvest other metadata sources. In this example, we will harvest the metadata catalog of the PEGASO project:

- From the GeoNetwork **Administration** interface, select the **Harvesting management** link:



- In the opening window, click on the **Add** button and choose to harvest a type **GeoNetwork v2.1 remote node**.

- Click **Add** once again.

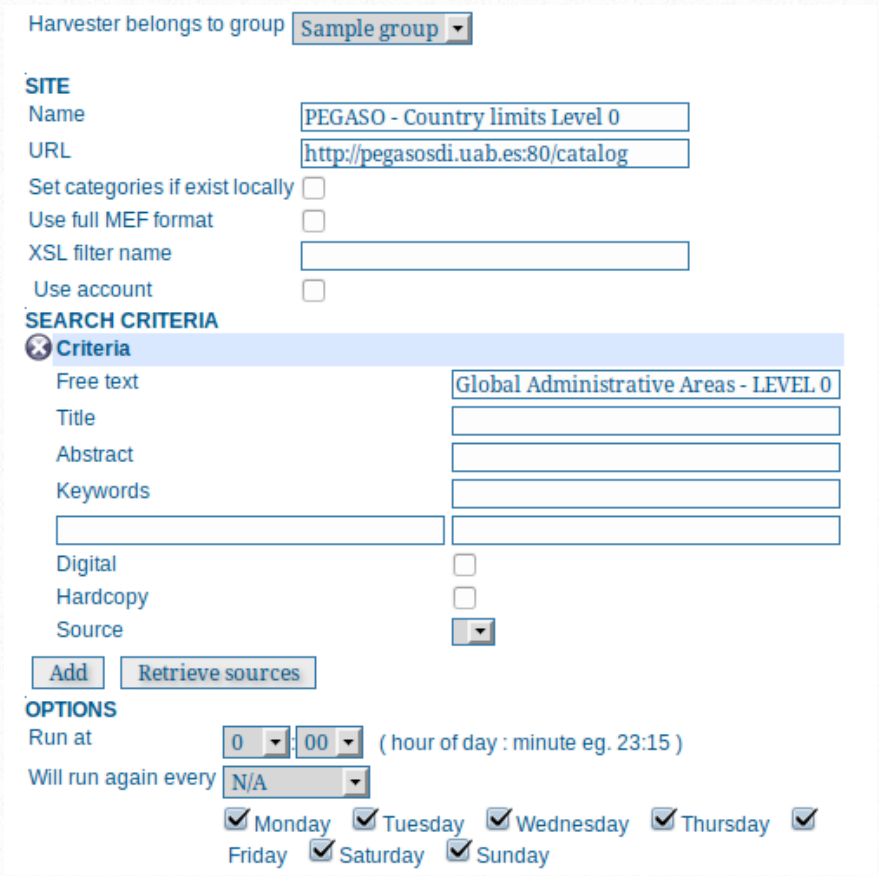
- **Fill the following information**:

Name: **PEGASO - Country limits Level 0**

URL: **http://pegasosdi.uab.es:80/catalog**

Use account: **the box should be unchecked**

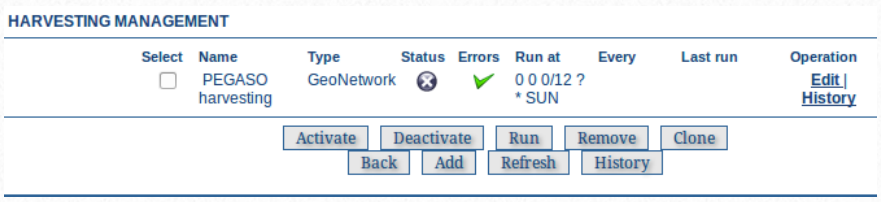
Criteria: click the **Add** button then for `Free text` enter: **Global Administrative Areas - LEVEL 0**



- Click **Save** to validate your choices.

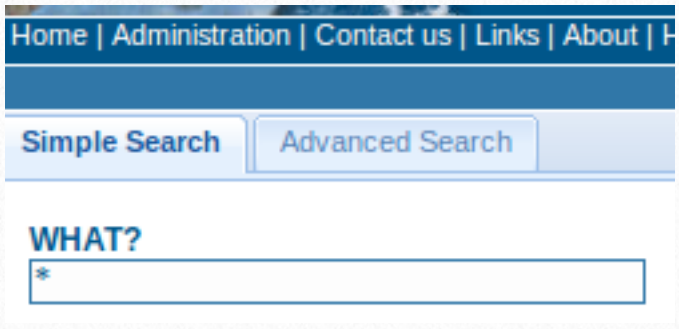
You just configured harvesting of a country limits layer published by PEGASO. You need to run the harvesting at least once to get the metadata on your GeoNetwork.

- **Check the box** next to `PEGASO harvesting` and click then on the **Run** button



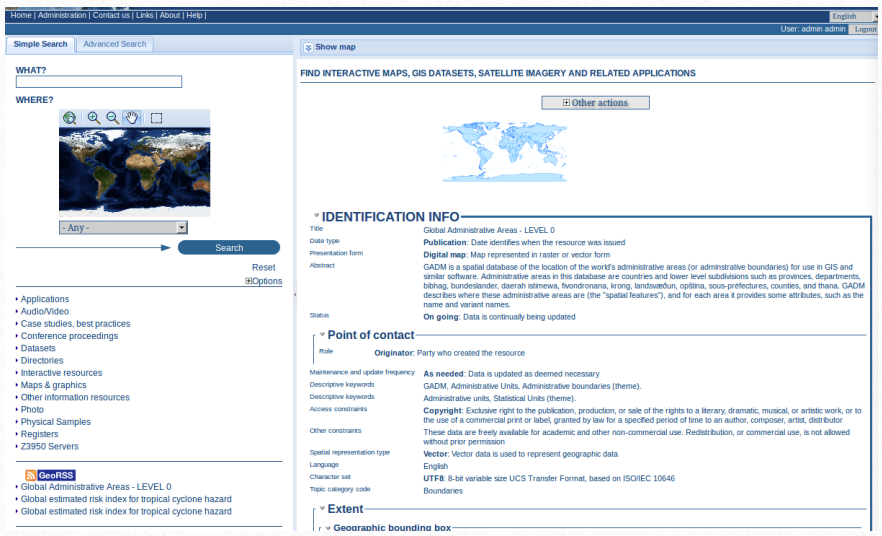
- **Go to the GeoNetwork homepage**.

- In the `What?` section enter a star (enter the * symbol):

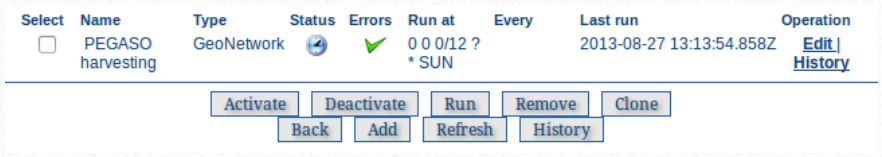


- Press **Enter** to validate your search.

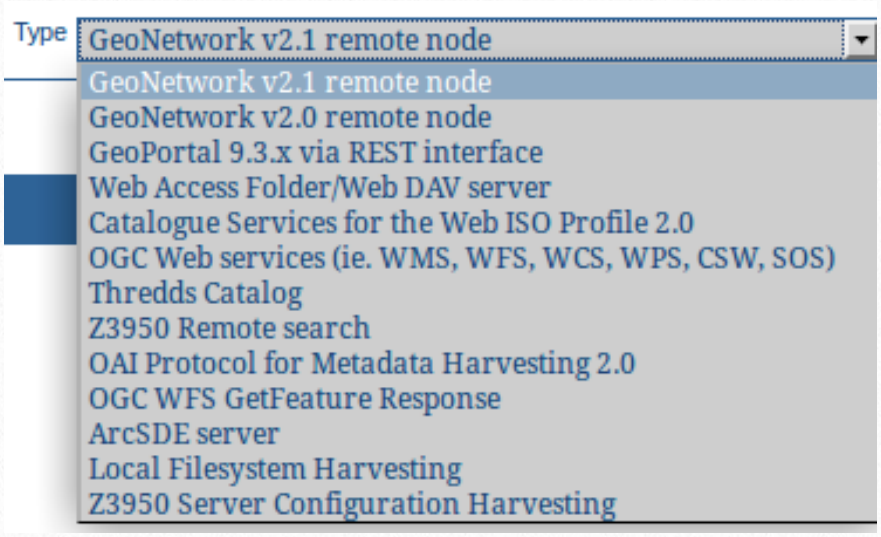
You will see a new layer called `Global Administrative Areas - LEVEL 0` appearing on the left pane. **Click on it** and you will see its metadata description, available in your GeoNetwork node.



Remember that a new harvest run will be performed regularly depending on the parameters you entered when you created the harvest. If you want to modify anything, simply click on the `Edit` link next to the harvest in question.



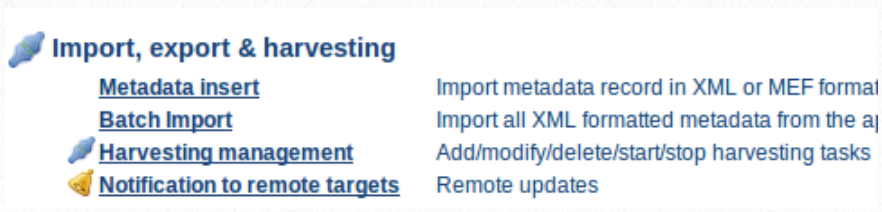
This exercice was done using the Geonetwork v2.1 remote node type of harvesting but several other types of harvesting exist:



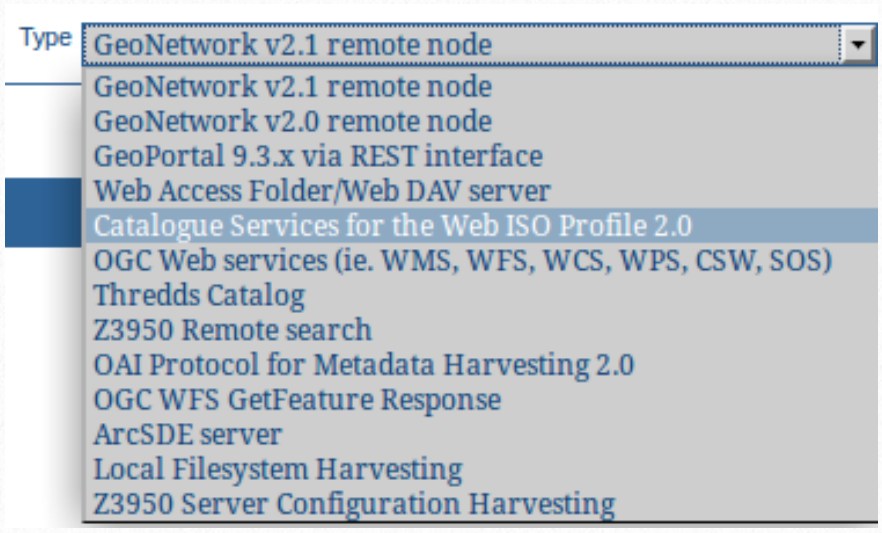
For example the one `Catalogue Services for the Web ISO Profile 2.0` also allows an harvesting of metadata catalog (not necessarily a geonetwork node) through CSW, and offers more filters options.

As you recall, we have used so far the `Free text` criteria (`Global Administrative Areas - LEVEL 0`) for filtering the results of the harvesting to only the record containing the text. If we use now the `Catalogue Services for the Web ISO Profile 2.0` type of harvesting, we will have access to more filters on the harvesting query.

- From the GeoNetwork *Administration* interface, **select** the **Harvesting management** link:



- In the opening window, click on the **Add** button and choose to harvest a type **Catalogue Services for the Web ISO Profile 2.0**.



- Click **Add** once again.

- Fill the following information and leave the rest by default:

  - Name: **GRID geonetwork**

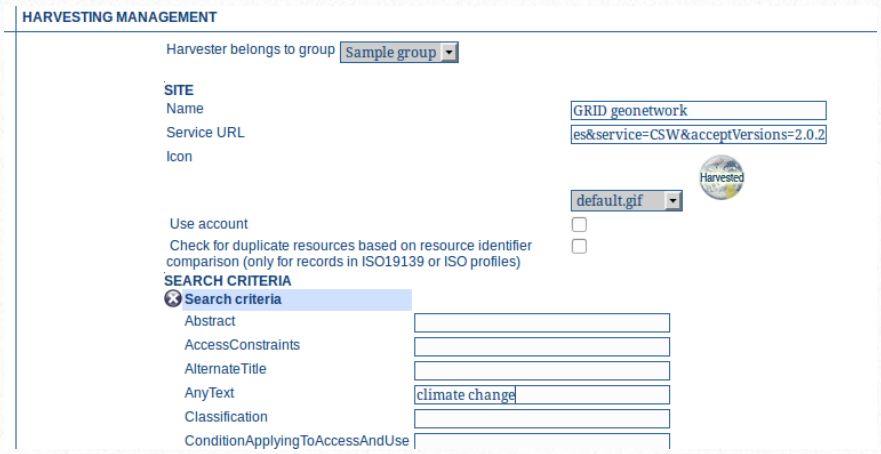
  - `Service Url:` **http://geonetwork.grid.unep.ch:8080/geonetwork/srv/en/csw?request=GetCapabilities&service=CSW&acceptVersions=2.0.2**

  - **Uncheck** the **Use account** box

  - Click on the **Add** button of the `Search Criteria`. This will open a series of fields

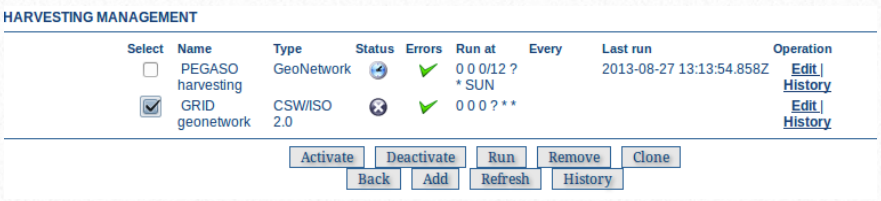
giving you the possibility to filter the results on the CSW query.

- In the **AnyText** search criteria, enter **climate change**:



- Click **Save.**

- In the harvesting management window that opens, **check** the box **GRID geonetwork** harvest and click on **Run**:



If you go now on the homepage and search for metadata records (in the What? section), you will see that new records appear in the bottom left pane, mostly linked to climate. When this tutorial was developed there were 26 new records.

In Section 1 of this Chapter you created new metadata. This was long and you had to copy/paste many information.

In Section 2 of this Chapter you imported existing metadata through `XML`. This was easy but metadata are not updated on your GeoNetwork even if they are updated by the original provider. To update them you have to import the `XML` file again.

In Section 3 you harvested metadata of a remote GeoNetwork node by creating a dynamic link between the original provider and your GeoNetwork; the harvesting is run regularly according to the frequency you decided. You also learnt how to harvest metadata from a CSW server (not necessarily a GeoNetwork node) and filter it thanks to the type of harvest `Catalogue Services for the Web ISO Profile 2.0`.

You might also want to obtain metadata from a CSW server but directly for your website and not for your GeoNetwork node. This is possible thanks to the CSW queries that will allow you to get an XML file that you will then be able to parse for example in php code for displaying it in your website. This goes beyond the scope of this course but you can find more references in the geonetwork developer manual.

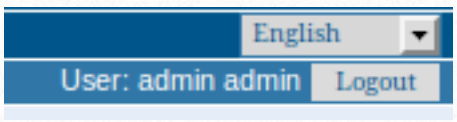
How to manage users' privileges on the meta-data?

So far we have focused on the ways to populate the GeoNetwork with metadata, be it by manual addition, `XML` import, harvesting or CSW queries. Two other important aspects of GeoNetwork need to be taken into consideration. The first one relates to the way it handles users, groups and privileges. Indeed, metadata records you enter are not always public. The second relates to ordering metadata items into specific categories. In this Section you will learn how to manage users, groups and categories.

Visualize public users' privileges

You want to know if public users can access metadata. At this stage of the exercise you should still be logged as administrator.

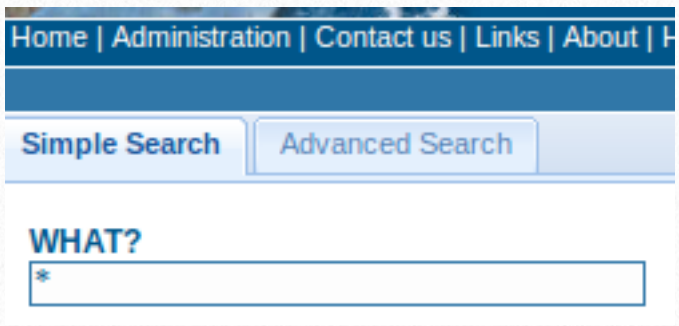
- **Log out**:



You are now connected as a public ('simple') user and not anymore as the GeoNetwork administrator.

- Click the **Home** link.

- In the `What` section enter a star (enter the * symbol):



- Press **Enter** to validate your search.

As you can see, the records you had entered previously are not visible to simple users. They are still in the system but not visible for the public, only for the group(s) that have this privilege.

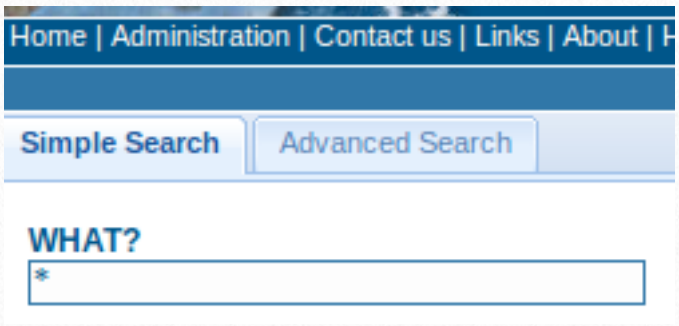
Note: simple users belong to the `Public group` (also called `All`).

Update privileges for public users

To update privileges you need to log in as administrator because only the administrator can edit/ update privileges.

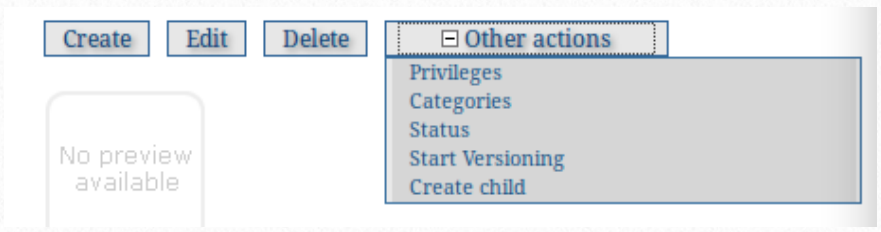
- **Log in** again (user name: **admin** and password: **admin**)

- By searching * in the **What?** section your records will appear again in the bottom left pane:



- **Click** the very bottom record **Global estimated risk index for tropical cyclone hazard** that you had manually entered.

- Click **Other actions > Privileges**.



This opens a window for managing users' and groups' privileges for this specific record. For each group it is possible to set up the following rights:

Publish: Users in the specified group/s are able to see the map, i.e. if searching with matching criteria.

Download: Users in the specified group/s are able to download the map.

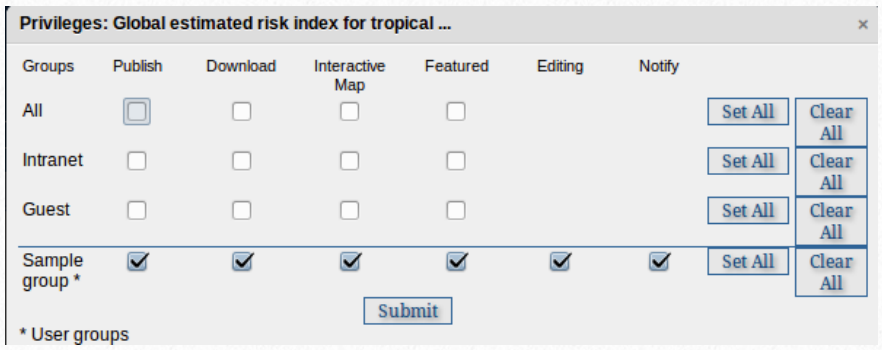
Interactive Map: Users in the specified group/s are able to get an interactive map. The interactive map has to be created separately using a Web Map Server, which is part of the GeoNetwork open source application.

Featured: When selected, the map is placed in the Features Maps of the home page and it appears there randomly.

Editing: When selected, the editors of the group(s) concerned can edit the respective metadata record.

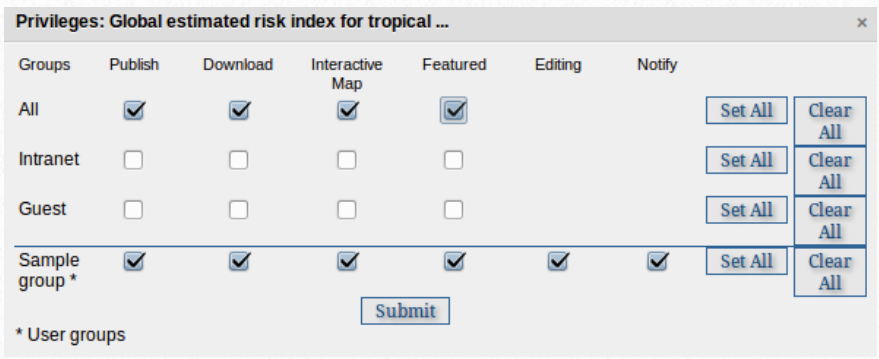
Notify: defines what Groups are notified when a file managed by GeoNetwork is downloaded

To assign specific privileges, click on the small box next to the privilege to place or remove a checkmark. `Set All` and `Clear All` buttons allow you to place and remove the checkmarks all at once.



We will make this record completely public:

- **Check all boxes for the Public group**:



- Click **Submit**.

- **Logout**.

You are now logged as a public user. You should be able to visualize the `Global Estimated Risk Index for Tropical Cyclone Hazard` metadata record.

Create a new user group

You will now learn how to create a new group of users. For this you need to log as administrator once again.

- **Log in** (user name: **admin** and password: **admin**).

- Click on **Administration** > **Group Management > Add a new Group**.

- **Create** a group called **GEOSS test group**. No need to enter a description or an email:

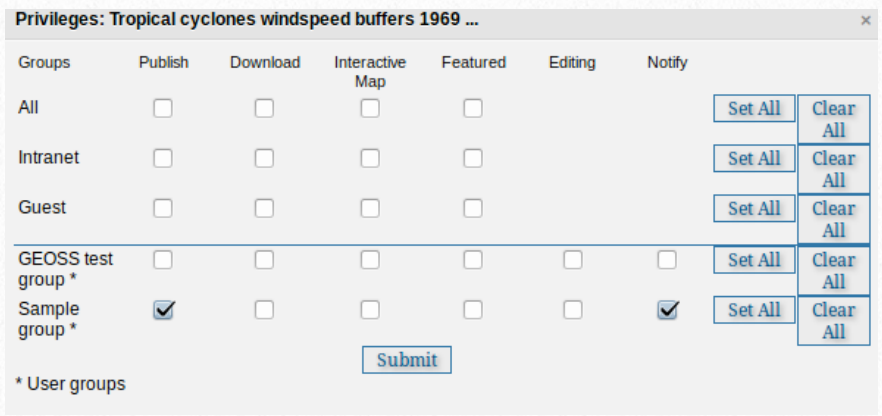
- Click the **Save** button.

- Click the **Back** button.

- Click again on **Home**.

- **Select any metadata record** and select **Other actions > Privileges**.

As you can see, the new group appears now and it is possible to assign privileges to it.



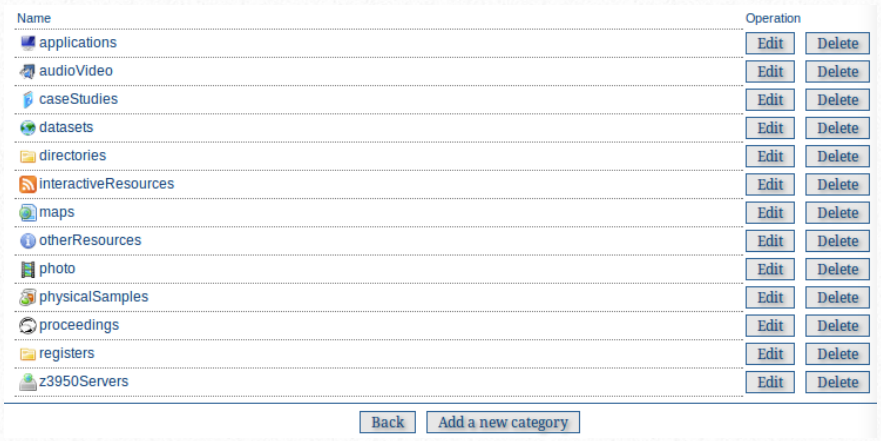
- You can now **close** the **Privileges** window.

Create a new category

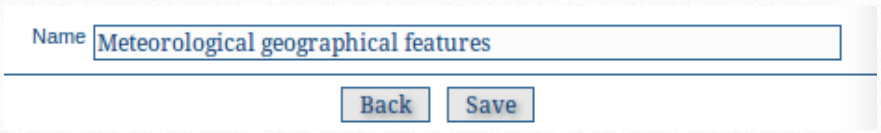
GeoNetwork gives the possibility to either assign existing categories to metadata records or to create new categories. It is strongly suggested to follow the INSPIRE categories to categorize your metadata records.

In the following exercise, we will create a new category to be INSPIRE compatible. You need to be logged as administrator, which is the case at this stage of the exercise.

- Click **Administration > Category Management > Add a new Category**:

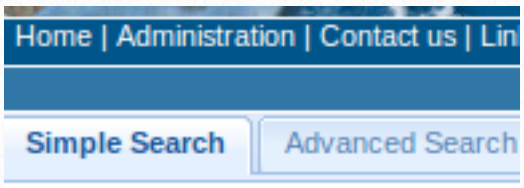


- Name the new category **Meteorological geographical features**. This is an INSPIRE Category.



- Click **Back**.

- Click the **Home** page.

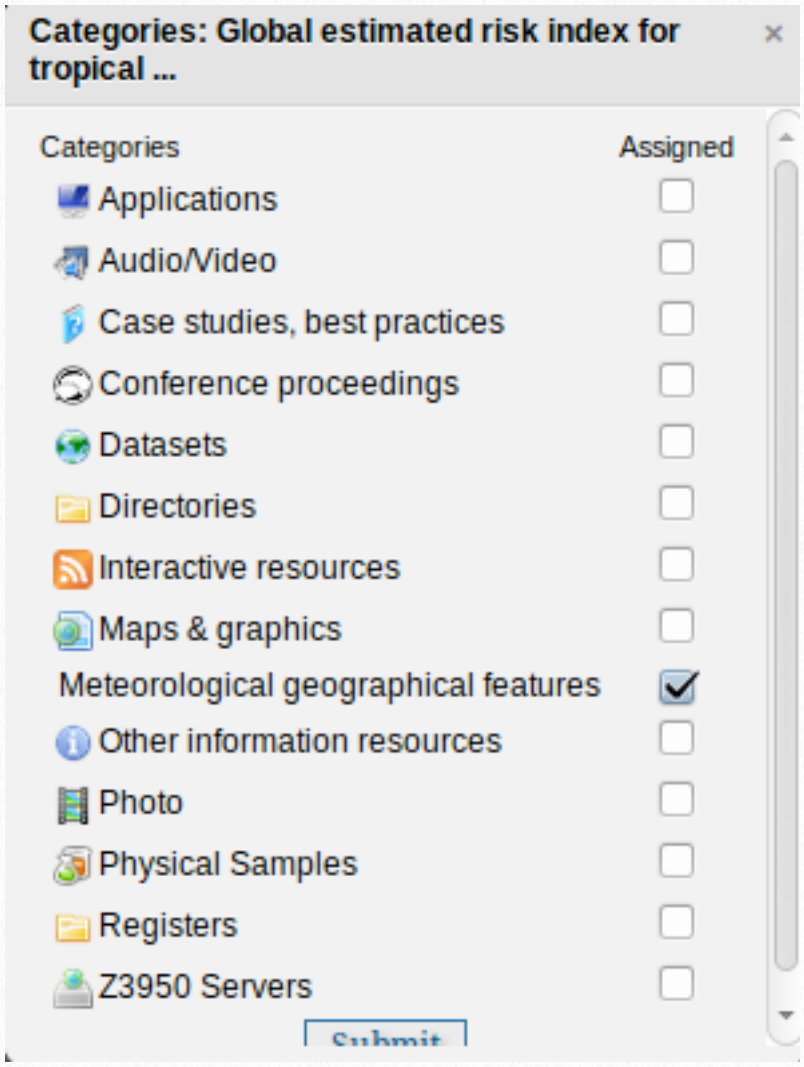


- Search and select **Global estimated risk index for tropical cyclone hazard**.

- Click **Other actions > Categories** in the right pane.

- **Uncheck `Datasets`** if checked.

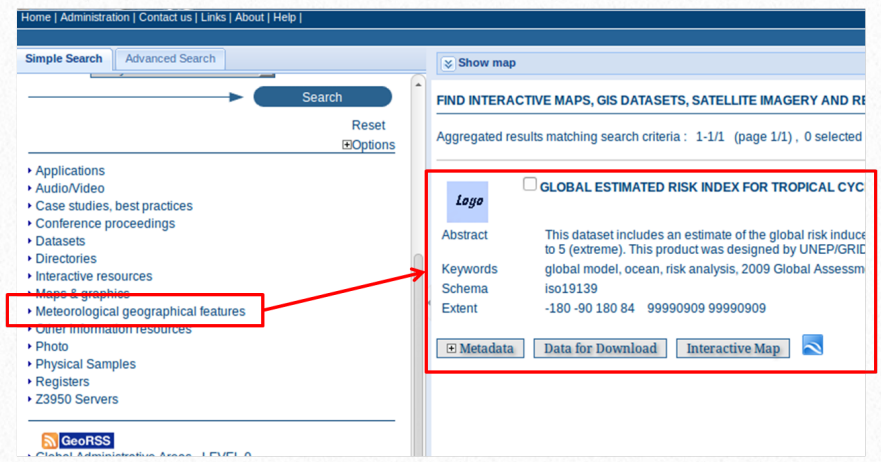
- Select the one you just created: **Meteorological geographical features**.



- Click **Submit**.

- From the **Home** page **click** the new **Meteorological geographical features** `category`.

One record appears: `Global estimated risk index for tropical cyclone hazard`:



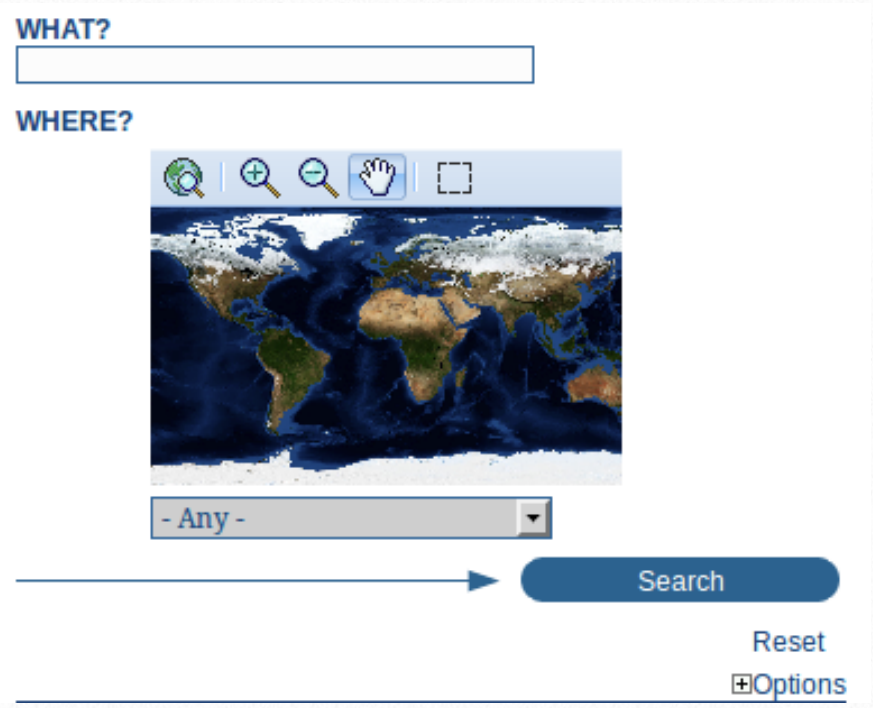
How to search and visualize metadata?

Default search

The default search allows you to search text within the entire record, such as keywords of the metadata and/or geographic location.

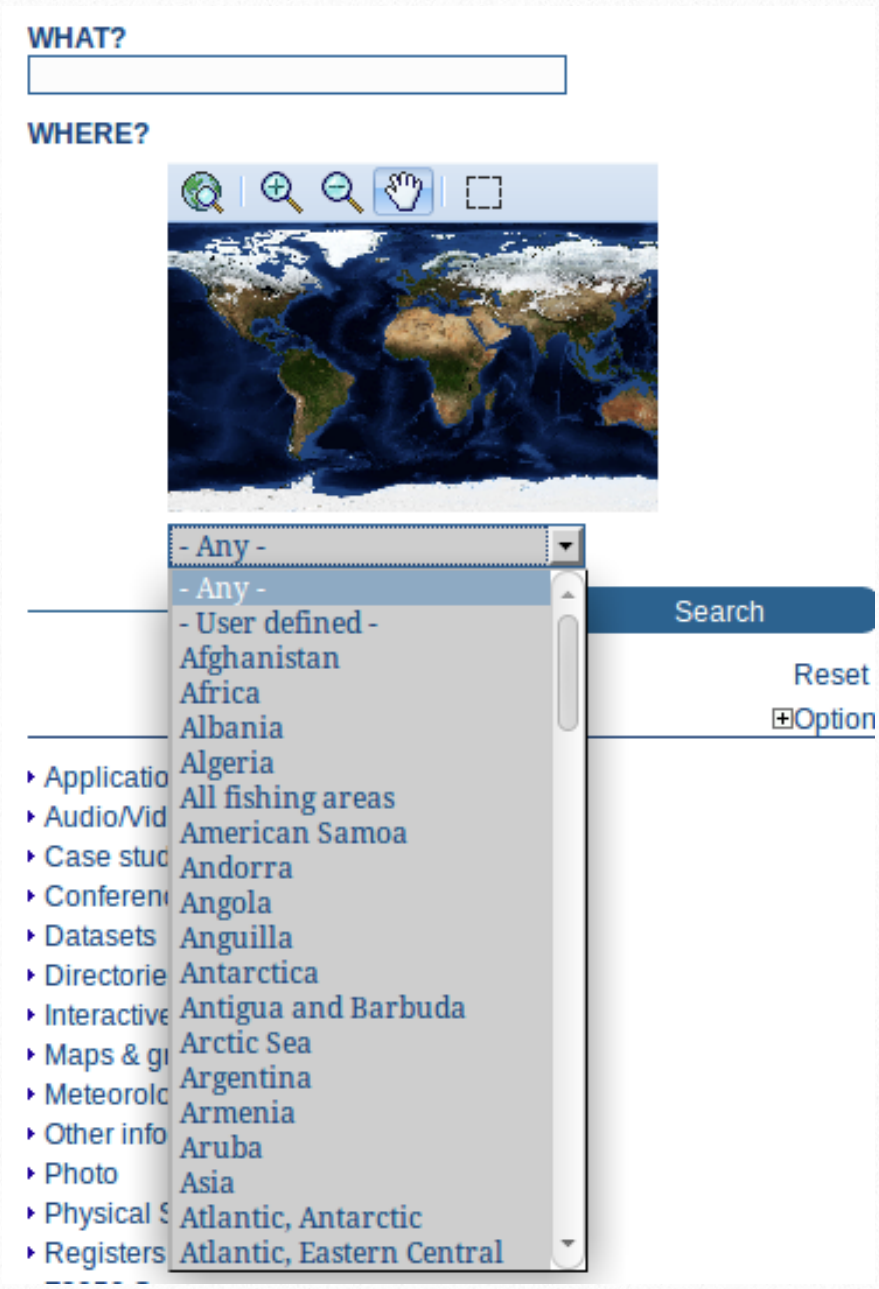
Free text search: Type a search term in the `What?` field. You can type anything here (free text). You can use quotes around text to find exact combinations of words.

Text and operators (and, or, not) are not case sensitive.



Geographic search: For the geographic search, two options are available for selecting a particular region to limit the search:

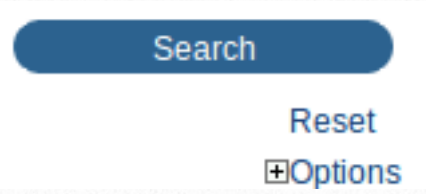
- You can select a region from a predefined list:



- You can select your own area of interest in a more interactive way. A small global map is shown on the screen from which you can drag and drop the frame of your location area. Just click on the rectangle on the upper right of the map screen.

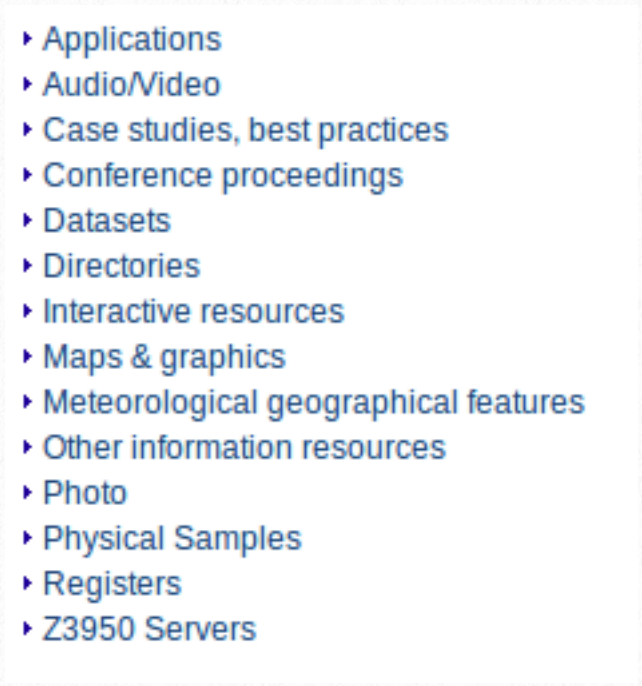
Perform search: Both types of search, free text search and geographic search can be combined to restrict the query further.

Click the Search button to proceed and show the results.



Searching by categories: An additional way to search data within the GeoNetwork database, from the home page, is searching by Category. A list of categories is provided to the user to identify data at a more generic level: `Applications`, `Audio/Video`, `Case studies and best practices`, `Conference proceedings`, `Datasets`, `Directories`, `Interactive resources`, `Maps and graphics`, `Other information resources`, `Photo`, `Physical Samples`, `Registers`, `Z3950 Servers`.

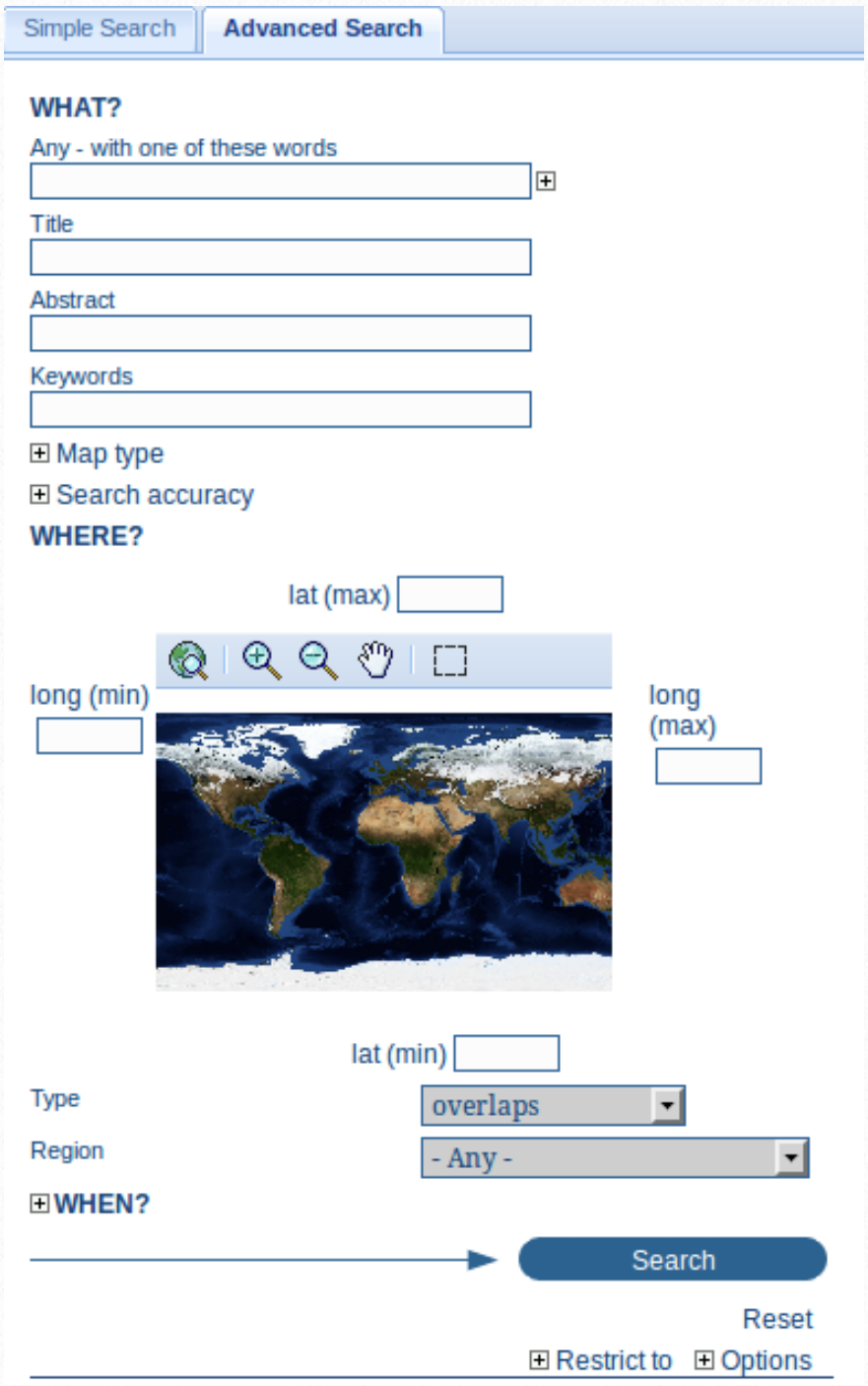
To search only for maps, click on Maps and Graphics. A list of maps will be displayed from which you may view details of every single map; just clicking on the `Metadata` button of the map you wish to review. As shown in the previous section, you can also add your own categories, in our case the INSPIRE category `Meteorological geographical features`.



Advanced search

The advanced search option works similarly to the default search. However, you can be more specific in your search criteria as it offers different elements to look for data, each of them focusing one of the following aspects: What?, Where?, When?

To perform an advanced search, from the home page click the `Advanced` tab, which displays more search options.



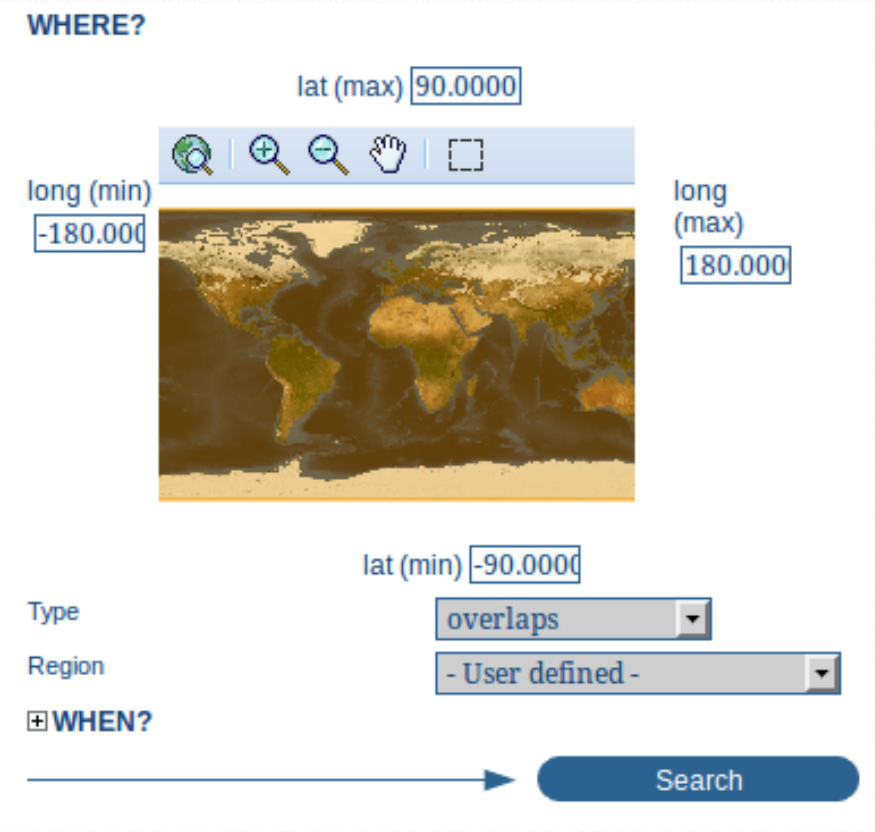
In the WHAT? section the elements are all related to the data content. Through them, in addition to searching only free keywords in the entire metadata content, you can also search directly in the title or abstract fields and add more keywords to customize your search further. You can also specify the level of accuracy you wish to reach in performing your search.

To search by Title, Abstract, Free Text, or Keyword(s) type any text into the respective field. You can enter information in one or multiple field(s). If you do not want to search by a given field, simply leave it blank.

You can choose the accuracy of your search, in terms of spelling words, from Precise = 1 to Imprecise = 0.2, through 3 more consecutive steps which are equal to 0.8, 0.6, 0.4.

The WHERE? parameters, which are related to the spatial extent, allow you, as in the default search, either to select your own area of interest or to select a predefined region from the drop-down list. In this section you can also type the geographic coordinates of a specific location that is not available from the above list.

- To select your own area of interest, drag and drop the frame of your area on the global map using the appropriate tool on the upper right of the map screen;
- To use free coordinates, type the lat-long geographic references in the appropriate fields around the map screen, without any limitation of decimal figures;
- To use the coordinates of a predefined region, select the region from the drop-down list.



Whatever type of geographic search you decide to perform, in the `Spatial search` type field, you can choose from different options: `is`, `overlaps`, `encloses`, `is fully outside of`. If you use this field, be cautious as this limits your output data as follows:

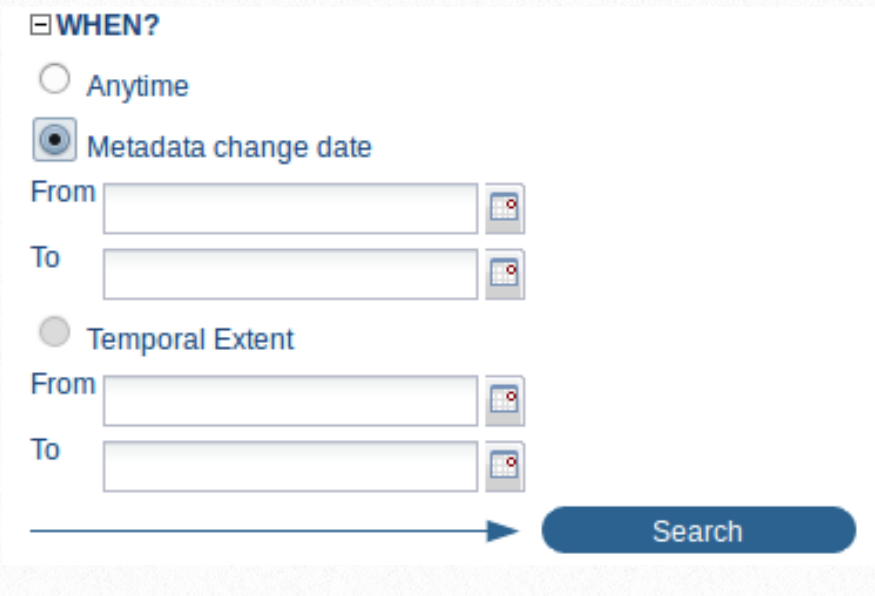
- If you choose Spatial search type `is + Region`, only maps for the selected region will be displayed. In other words, a city map within that region will not show in the output results.

- If you choose Spatial search type `overlaps + Region`, all maps with the bounding box overlapping that region will be displayed in the results, i.e. the neighboring regions, the continent of which that region is part of and the global maps.

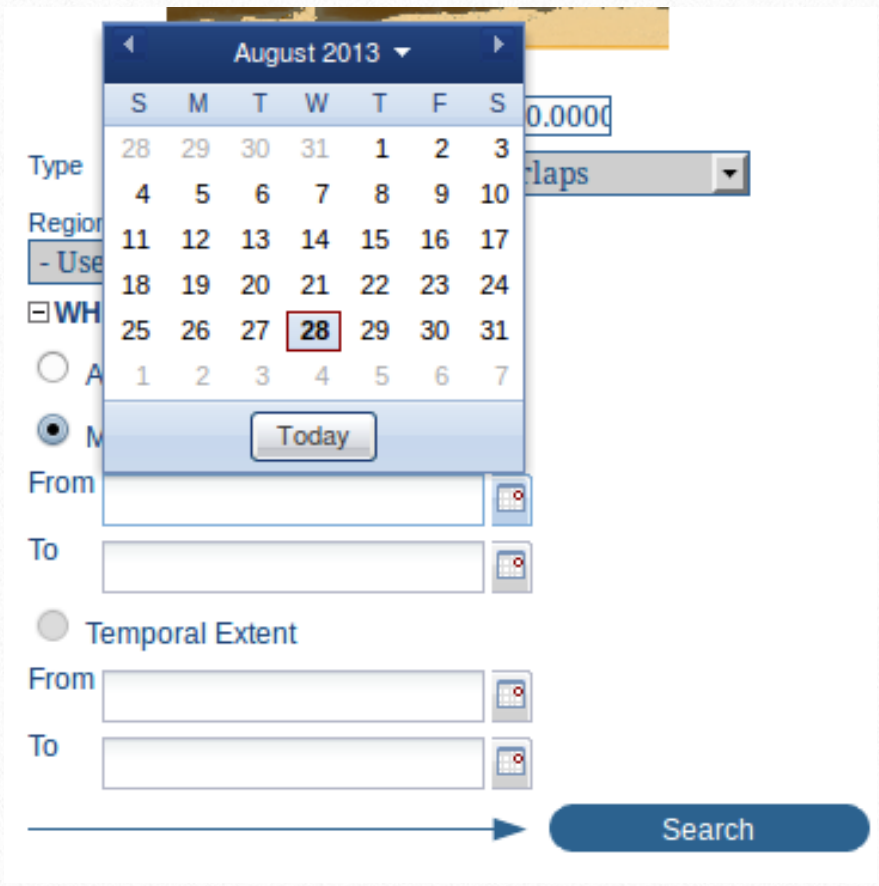
- If you choose `encloses + Region` you will get, in the output results, maps of that region first and then all maps within its bounding box.

- Similarly, if you choose `is fully outside of + Region`, only maps that follow that exact criteria will show in the output results.

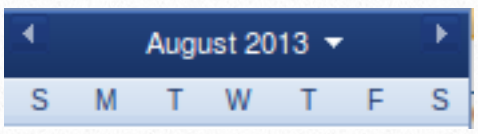
The `WHEN?` section gives you the possibility to restrict your search in terms of temporal extent, indicating a specific range of time referred to the metadata creation/change date or data temporal extent.

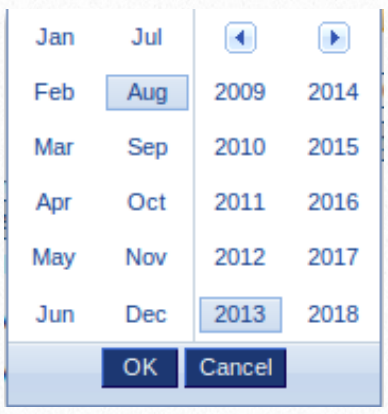


To specify a range of time for any of these, click on the date picker calendar next to the corresponding fields.



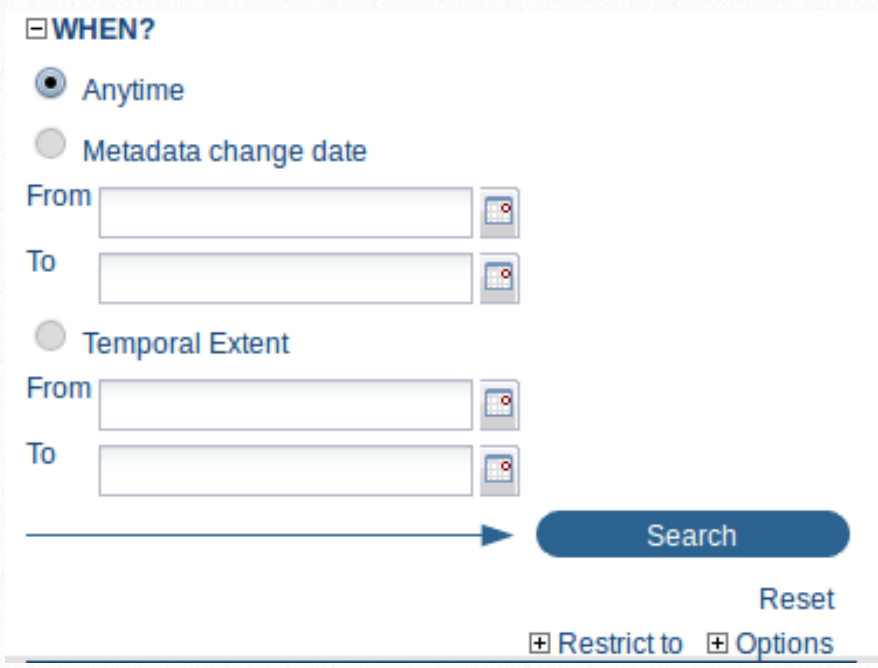
Make use of the symbol ▼ on top of the calendar to select the month and the year first





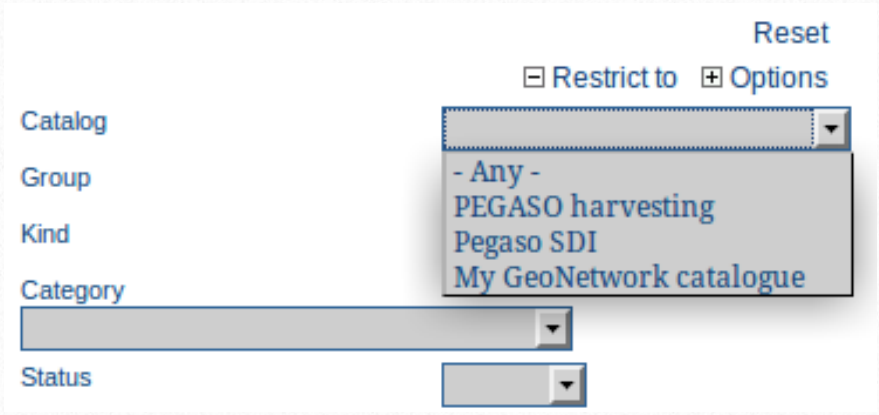
Click then on the exact day; a complete date will be filled in using the following standard order: YY-MM-DD.

To clean the time fields, simply click on the `any-time` radio button or the `Reset` link at the bottom; the search will be performed without any restriction on the time period.

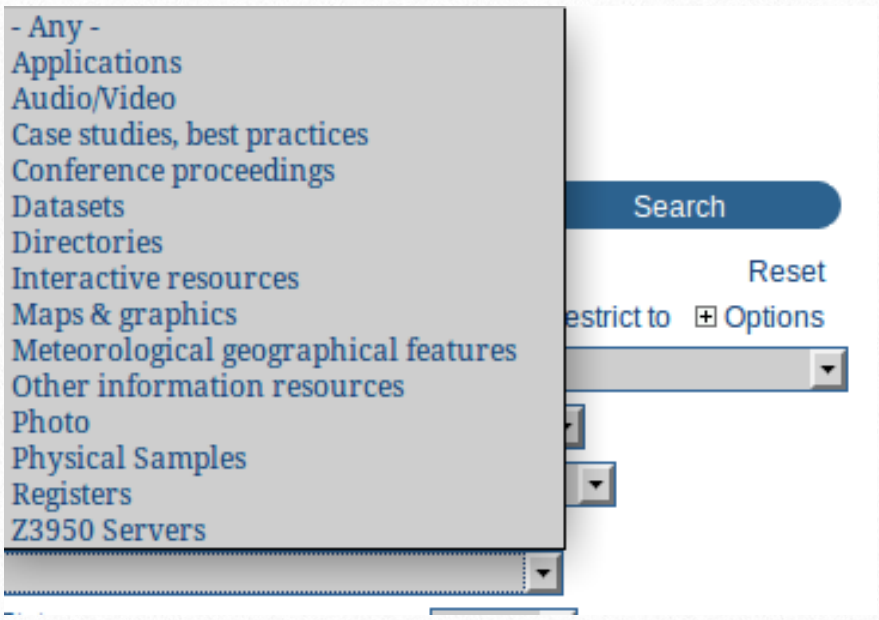


Finally, the advanced search allows you to apply further restrictions on the basis of additional parameters as data source, data categories and data format.

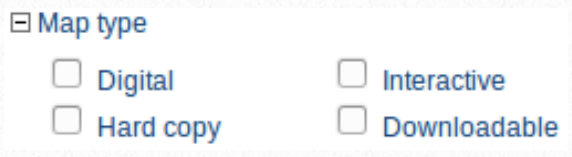
- To limit your queries to only one `Catalog` out of those made available by the installation through the harvesting process, highlight the catalogue of preference or just keep `Any` selected to search all sites.



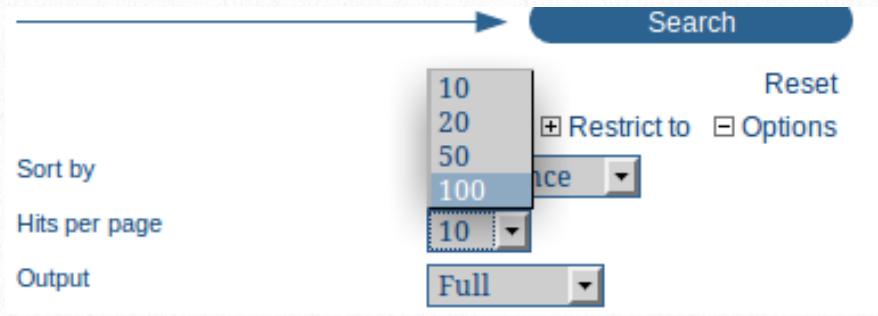
- To search for data organized by `Category`, such as `Applications`, `Datasets`, etc., simply highlight the category you wish to search in from the related drop-down list, otherwise we suggest to leave this field in `Any` category.



- You can search for `Digital`, `Interactive`, `Hard copy` or `Downloadable` maps. If no box is checked all content will be searched



Finally, you can customize the number of output results per page in the `Hits Per Page` field. Simply highlight the number of records to be displayed or leave the field set on the default number (10).



Once you are satisfied with the search parameters, you can perform the search by clicking on the `Search` button.

Viewing metadata

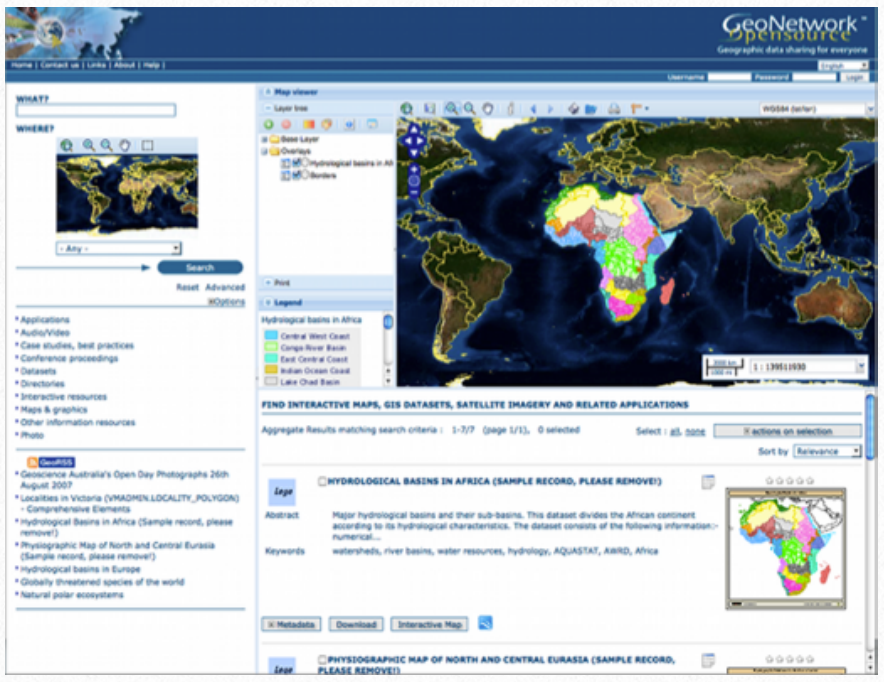
The output of a search provides you a list of the metadata records that should fit your request. For each record, the result page shows the title, an abstract and the keywords.

Metadata: The metadata section describes the dataset (e.g.: citation, data owner, temporal/ spatial/ methodological information) and could

contain links to other web sites that could provide further information about the dataset.

Download: Depending on the privileges that have been set for each record, when this button is present, the dataset is available and downloadable. The process for retrieving data is simple and quick by just clicking the download button or by using the proper link in the specific metadata section for distribution info in the full metadata view.

Interactive Map: The map service is also optional. When this button is shown, an interactive map for this layer is available and, by default, it will be displayed on the map screen of the simple search. To better visualize the map through the map viewer, click `Open Map Viewer` on the bottom left of the map screen.



- You can now **disconnect** from **GeoNetwork** and **close your web browser**.

In Chapters 3 and 4 you learnt how to publish data and metadata. In the next chapter you will process geospatial data.

5

How to process data?

Keywords: environmental data; geo-processing; Python script; PyWPS; Web Processing Service (WPS)

What you will learn:

- Analyze geospatial data through web services
- Publish a Python script as WPS
- Chain geoprocessing operations and store them in a Python script
- Generate data in GML, SHP and TIFF formats.

Definitions

This section will provide a few definitions of basic concepts underlying the publication and the consumption of web processing services. With this section, you will have all the necessary knowledge to go through Chapter 5.

The OpenGIS® Web Processing Service (WPS) Interface Standard provides rules for standardizing inputs and outputs (requests and responses) for geospatial processing services, such as polygon overlay. The standard also defines how a client can request the execution of a process, and how the output from the process is handled. It defines an interface that facilitates the publishing of geospatial processes and clients' discovery of and binding to those processes. The data required by the WPS can be delivered across a network or they can be available at the server.

Python is an open source and free programming language that is used in a wide variety of application domains, e.g. web and internet development, database access, desktop graphical user interface, scientific and numeric-based computing, education, and many others. Python includes dozens of functions for managing and analyzing geospatial data. Python is used by commercial as well as open source GIS solutions. In this chapter of the training you will use Python 2.7 which is the current version as of 2013. Python runs on Windows, Linux, Mac OS X and has been ported to the Java and Dot Net virtual machines.

Python Web Processing Service (PyWPS) is an implementation of the Web Processing Service standard from Open GeoSpatial Consortium. The project has been started in May 2006. It offers an environment for programming own processes (geofunctions or models) which can be accessed from the public. The main advantage of PyWPS is that it has been written with native support for GRASS GIS.

Quantum GIS, also known as QGIS, is a cross-platform free and open source desktop application that provides data viewing, editing and analysis capabilities. QGIS runs on Linux, Mac OS X, Microsoft Windows and Android. QGIS supports vector, raster and database formats, such as shapefiles, PostgreSQL/PostGIS, GRASS and GeoTiff. QGIS supports a number of plugins, in particular a WPS client for consuming web processing services.

Objectives of the chapter

The main objective of this chapter is to learn how to analyze geospatial data through web processing services. This chapter builds on a practical exercise.

The scenario is the following: one of your partners working in the field of environmental risk has made measurements of pollution (e.g. by heavy metal) in a specific area. This partner needs an overview of the pollution in the area of study in order to report to local authorities and to agree with these authorities on which action to take.

A requirement from your partner is that the overview should be produced quarterly. Another requirement is to have a visual map of contamination by heavy metal. Your partner also needs for further analysis: (1) vector buffers around the pollution points, and (2) the boundaries of the contaminated zone.

To address these requirements, you will register a series of geoprocessing operations in a Python script.

This script will generate different outputs: (1) polygonal buffers around the input contamination data, (2) a convex hull representing the whole contaminated area, and (3) a contamination map based on interpolation from the original contamination data.

To allow reusing the script regularly, with updated data and under various conditions, you will declare parameters, in particular the input point layer and the buffer distance.

The script will be published on your local server, ready to be consumed (executed) from a desktop client. Consumption will be done in Chapter 8 through the QGIS WPS client.

This exercise will also give you the opportunity to generate data in three distinct output formats (GML, SHP and GeoTIFF) and in two distinct graphic types (vector and raster).

In the first step of this exercise, you will load the data in the QGIS desktop client to get familiar with them. In particular you will visualize the number of points, the coordinate system, the attributes and the extent of the contamination layer.

Note: all data provided for this exercise are fictive and do not reflect the reality in the field. The area of study was selected arbitrarily and is not meant to be actually polluted by heavy metal or any other contaminant.

Exercise: Processing data

1. Navigate through input data

In what follows you will add the input data to a new QGIS project to get familiar with them. You will be learn how to get the information about the number of points, the geographical extent, the coordinate system and the attributes.

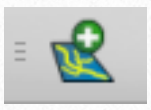
- Click on **Applications Menu**. This is the blue button on the upper left of your virtual machine.

- Select **Education > Quantum GIS Desktop**.

A new empty project opens.

By default you get a QGIS Tips! Windows each time you start the application. Even if it is a good source of info you can switch it off by checking the I've had enough tips,… checkbox so that it does not appear every time you launch the application.

- Click on the **Add Vector Layer** tool.



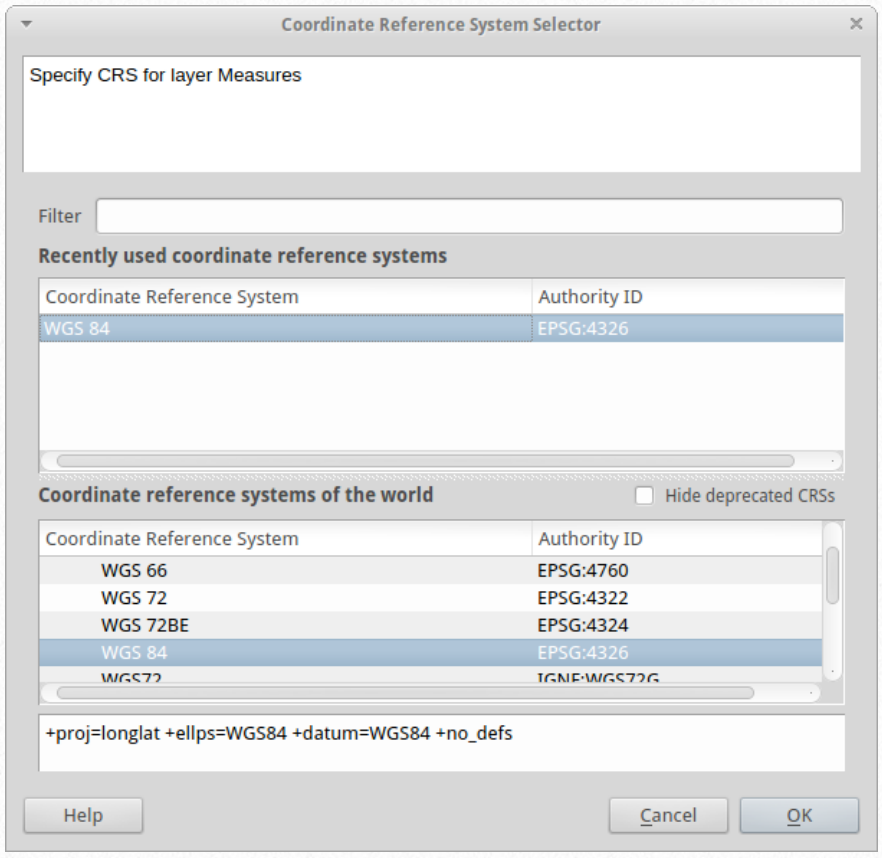
- Browse to **Geoss > Desktop > data_exercises**. Select **Measures.gml**.

This Geography Markup Language (GML) file contains the input measures. GML is an eXtensible Markup Language (XML) grammar for expressing geographical features. GML serves as a modeling language for geographic systems as well as an open interchange format for geographic transactions on the Internet.

XML is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. The XML syntax is used by many standards, protocols and technological solutions.

- Click the **Open** button to load the file into QGIS. Click **Open** once again.

- Select the **EPSG:4326** coordinate system. This system corresponds to the World Geodetic System 1984 (also called WGS 84).

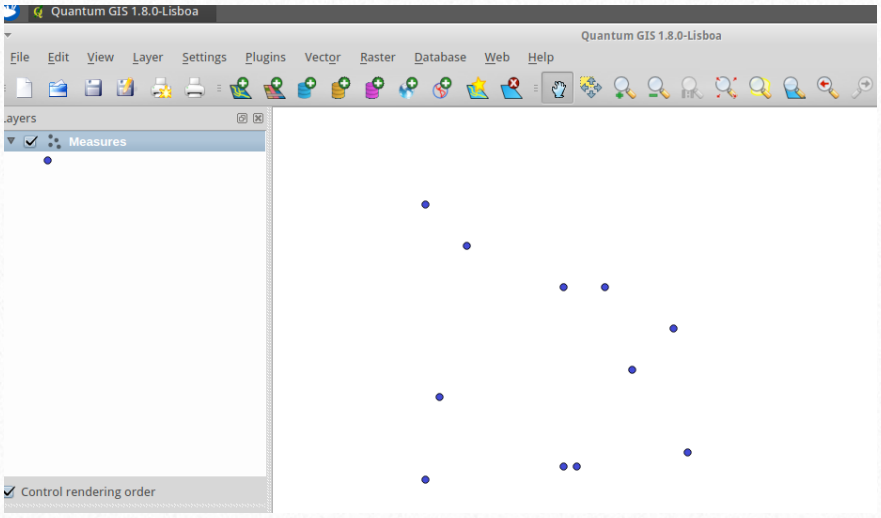


WGS 84 is a standard for use in cartography, geodesy and navigation. It is based on a coordinate grid for the Earth (latitude / longitude), a reference spheroidal surface (also

called “datum” or “reference ellipsoid”) for altitude data, and a gravitational equipotential surface (“geoid”) that defines the nominal sea level.

- Click **OK**.

The input point layer is displayed in QGIS. It is composed of 11 points.



- Click the **Attribute Table** button to open the attributes of the input points.

Each input point has an X and a Y coordinate stored in WGS 84. The extent of the layer are: XMin = 8 Decimal Degrees (DD);

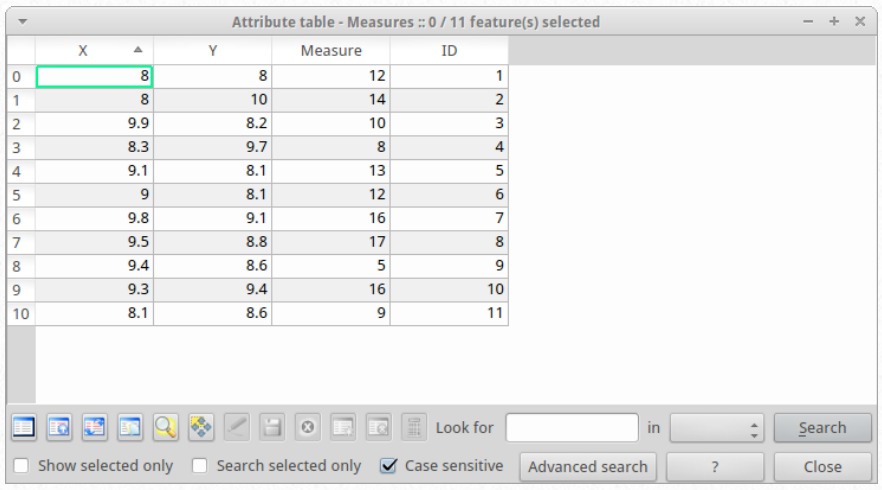
Xmax = 9.9 DD;

YMin = 8 DD;

YMax = 10 DD.

Note that this extent is arbitrary and does not reflect the reality in the field.

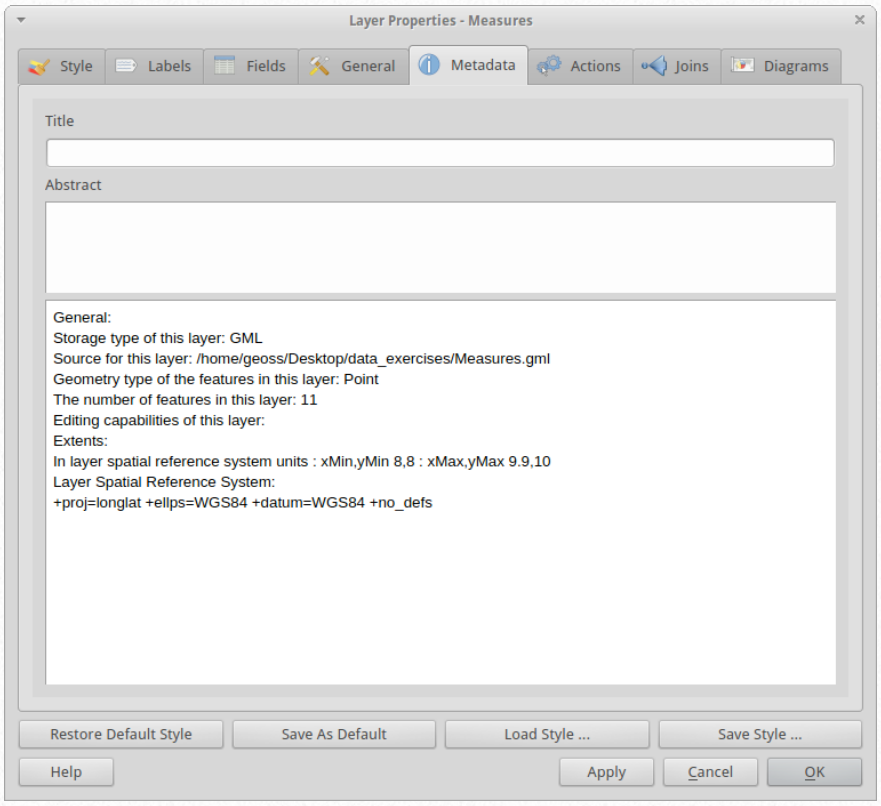
The layer also has a unique identifier (ID field) and a field called `Measure` that stores the pollution measures. Measures do not have a particular unit, since data are fictive. The data used in this exercise represent contamination by heavy metal, but this exercise could be applied to atmospheric pollution, radioactivity or any type of environmental pollution. Measures range from 5 to 17.



	X	Y	Measure	ID
0	8	8	12	1
1	8	10	14	2
2	9.9	8.2	10	3
3	8.3	9.7	8	4
4	9.1	8.1	13	5
5	9	8.1	12	6
6	9.8	9.1	16	7
7	9.5	8.8	17	8
8	9.4	8.6	5	9
9	9.3	9.4	16	10
10	8.1	8.6	9	11

- You can now **close the attribute table** of the points layer.

- From the table of contents of QGIS, **right-click** the **Measures** layer and select **Properties**. Select the **Metadata tab**: information about the extent, the coordinate system, and the number of points is summarized in the opening window. Other information can be accessed from this window: format (GML), path and name of the source file, type of geometry (points) and editing capabilities.



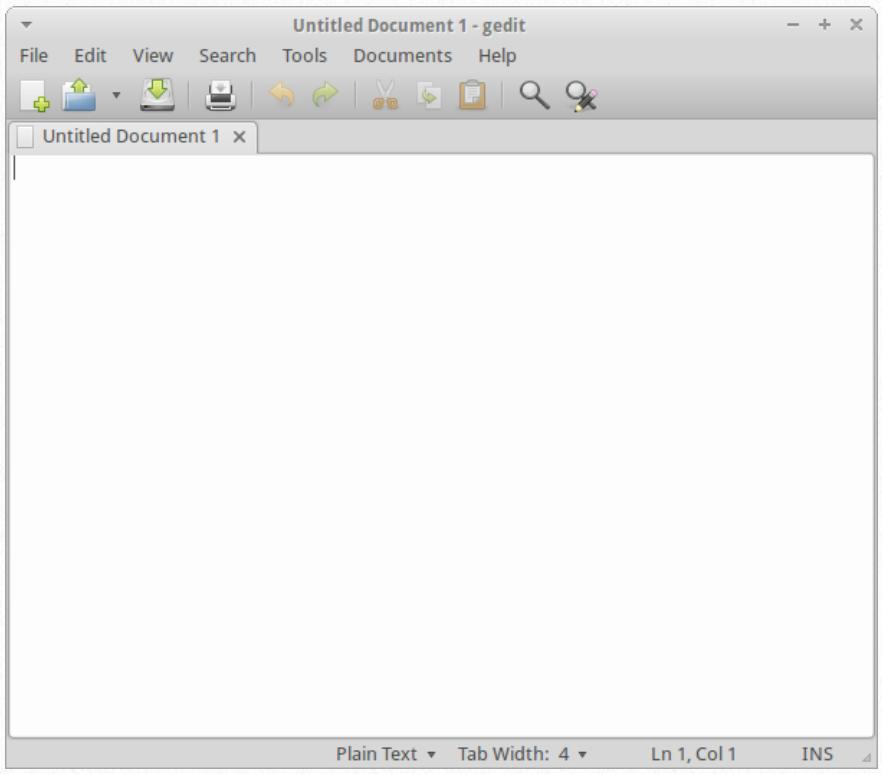
- Click OK or **Cancel** to close the layer properties.
- Close Quantum GIS, e.g by selecting the **File menu > Exit** or with CTRL+Q. When prompted, choose **Close Without Saving**.

2. *Write the Python script*

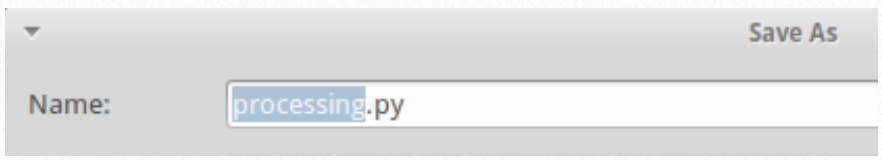
In this paragraph, you will write the Python code for processing the data. You will declare input and output parameters and write instructions for generating a buffer, a convex hull and a contamination map. The source code will be written in a text editor. Examples of well-known text editors are: `NotePad++`, `IDLE`, `LeafPad` and `gedit`. You will use the fourth one which has been installed on your virtual machine. One advantage of using `gedit` is that the source code is written in different colors, which provides more readability.

- Click on **Applications Menu**.

- Select **Accessories > gedit**. A new empty file called `Untitled Document 1` opens.



- To save this file, click on **File > Save as...**
- Browse to **Desktop/data_exercises**
- In the Name field, enter: **processing.py**



- Click on **Save** to validate.

The script will be composed of three main parts: (1) initialization of the process and declaration of useful libraries, (2) declaration of the input and output parameters, and (3) coding of the geoprocessing operations through specific commands.

- In what follows you will write the first part of the script. At the top of the *processing.*py document, enter the following code:

```
from pywps.Process import WPSProcess

class Process(WPSProcess):
```

```
def __init__(self):
        WPSProcess.__init__(self,
                identifier = "processing",
                title = "Data geoprocessing",
                version = "1.0",
                storeSupported = "true",
                statusSupported = "true",
                grassLocation = True,
                abstract = "Processing data")
```

This first section of code deals with the initialization of the process. The `__init__` function (with two “underscores") will be discussed later in this chapter.

The process is inherited from the `pywps.Process.WPSProcess` class. In Python, you need to initialize this parent class from the start.

For that you must define the process identifier and a human-readable title. Other elements can be added, such as:

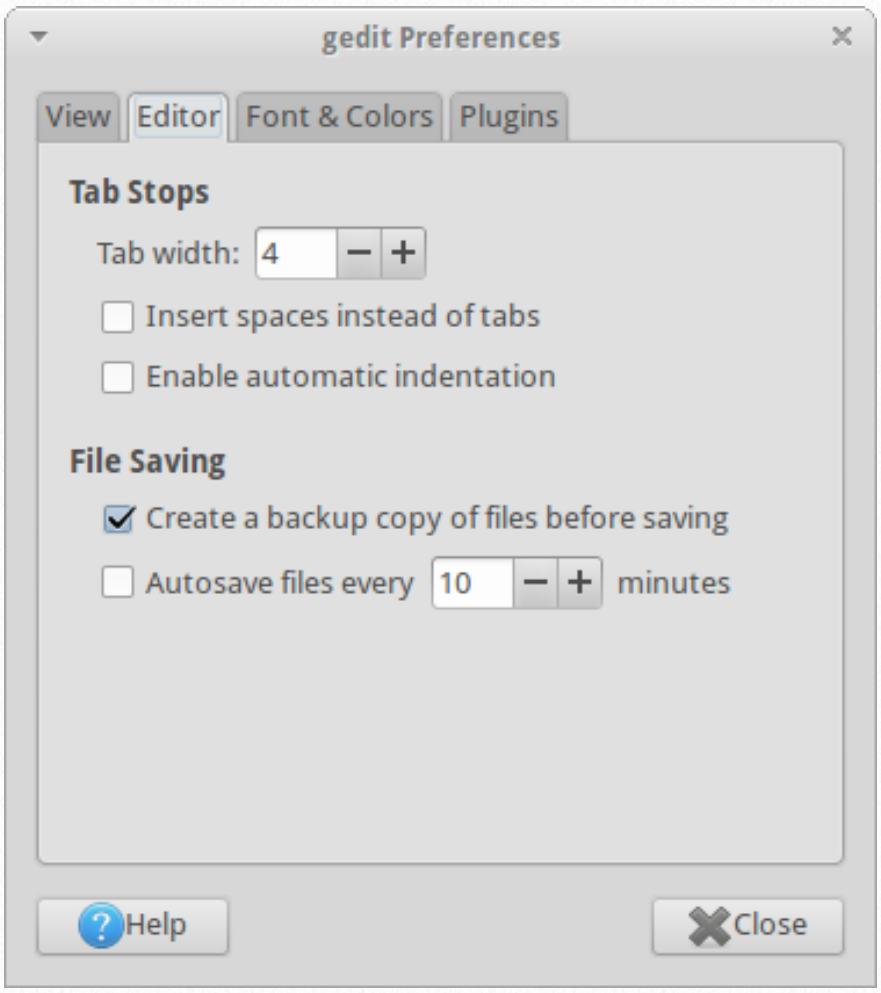
- An `abstract` that will appear when you will consume the process in a client. This abstract is for describing what the script does;
- The process `version`;
- Additional `metadata` (keywords);
- The `statusSupported` parameter that indicates whether the process supports asynchronous calls;
- The `storeSupported` parameter that indicates whether the results of the process can be stored on the server for later download;
- The `grassLocation` parameter. If set to `True`, this parameter indicates that you will use the GRASS commands in your process. These commands will be called later with `self.cmd` in the third part of the script.

Note that the code is indented. For example, the third line of code (starting with `def`) is not left justified but is shifted compared to the first two lines of the script.

If you do not indent the code, its execution will generate errors. If you want to indent a line of code, you can use the Tab button located on your keyboard.

Similarly, the fourth line of code (starting with `WPSProcess...`) is indented twice and the following instructions are indented three times.

In the `gedit` application, you can define tabs width as well as other settings from `Edit > Preferences`.



Also note that the list of parameters described above is surrounded by parentheses. These parentheses are mandatory.

In what follows you will write the second part of the script, which deals with the declaration of input and output parameters. These parameters will be visible when executing the script (e.g. from the QGIS WPS client: see Chapter 8), in the form of drop-down lists or text boxes.

From the `gedit` window, you need to enter the four following instructions:

```
self.dataIn =
self.addComplexInput(identifier="data",title="Input
data",formats = [{'mimeType':'text/xml'}])

self.widthIn =
self.addLiteralInput(identifier="width",title="Width",
type="Float")

self.bufferOut =
self.addComplexOutput(identifier="buffer",title="Out
put buffer file",formats = [{'mimeType':'text/xml'}])
```

```
self.outputInterpo=self.addComplexOutput(identifier
="output",title="Output interpolation
surface",formats=[{'mimeType':'image/tiff'}])
```

These instructions should be indented twice, i.e. at the same level of indentation than `WPSProcess.__init__(self`.

The first instruction is for declaring the first parameter of the script. This parameter corresponds to the input layer with contamination measures. `ComplexInput` means that this layer contains geospatial objects (either vector or raster). The layer is stored in XML, hence the type `text/xml`. The identifier (`data`) is declared at this stage and will be reused later, in the third part of the script. The title (`Input data`) will be displayed in the QGIS WPS client processing window when executing the script (See Chapter 8).

The second instruction is for declaring the buffer distance. This parameter does not contain geometry (vector/raster), therefore the command `LiteralInput` is required. The type is set to `float`, which means that it can store decimal values, e.g. 0.1 decimal degrees (DD) or 100.5 meters.

The third parameter corresponds to the output GML file with buffers, encoded in XML as is the input layer. In our case, the buffer will represent circles around the input points. These circles are not yet created, hence the use of the `addComplexOutput` command rather than `addComplexInput`.

The fourth parameter corresponds to the output interpolation surface that will be created from the input measures. The type is set to `image/tiff`. Other possible type values are: `image/png` and `image/geotiff`.

Note that the four above mentioned parameters are declared in a specific order. This order will be preserved in the QGIS WPS client processing window (See Chapter 8).

- In what follows you will write the third part of the script. Start by entering the following instruction:

def execute(self):

This instruction initializes the function that will contain the geoprocessing commands. This line of code must be indented once, i.e. it needs to have the same level of indentation than `def __init__(self)`.

Enter now the 13 following lines of code. Make sure that they are indented twice.

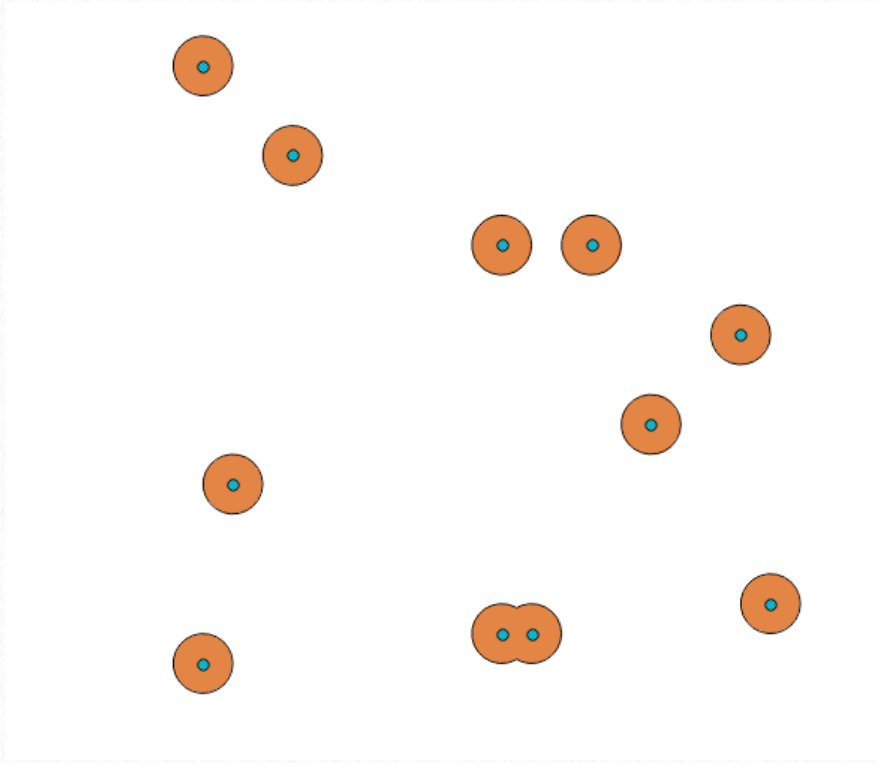
```
self.cmd("v.in.ogr dsn=%s output=data" %
(self.getInputValue('data')))

self.cmd("v.buffer --o input=data output=data_buff
buffer=%s scale=1.0 tolerance=0.01" %
(self.getInputValue('width')))

self.cmd("v.out.ogr type=area format=GML
input=data_buff dsn=outGML.xml
olayer=outGML.xml")

self.bufferOut.setValue("outGML.xml")

self.cmd("v.out.ogr type=area format=GML
input=data_buff dsn=/tmp/outputGML.gml")

self.cmd("v.out.ogr type=area
format=ESRI_Shapefile input=data_buff dsn=/tmp
olayer=outputSHP")

self.cmd("v.hull --o input=data output=outputHull")

self.cmd("v.out.ogr type=area
format=ESRI_Shapefile input=outputHull dsn=/tmp
olayer=outputHull")

self.cmd("g.region vect=data_buff rows=50 cols=50")

self.cmd("v.surf.idw --o input=data
output=idwRaster column=Measure")

self.cmd("r.out.gdal input=idwRaster output=./
tmp.tif")

self.outputInterpo.setValue("./tmp.tif")

return
```

Most of these instructions are embedded into the GRASS `self.cmd` command.

`v.in.ogr` converts a vector layer to a GRASS vector map.

The `self.getInputValue` command is for reading the path and name of the layer from the client processing window.

`v.buffer` creates a security perimeter (also called «buffer») around each input point.



The value of the buffer distance (e.g. 0.1 DD or 100.5 meters) is read from the width parameter through the self.getInputValue command.

The `--o` flag allows overwriting a previous result, e.g. if the script has already been executed in a previous run.

`data_buff` is the name of the output variable.

`scale` is a scaling factor for attribute column values.
The `tolerance` parameter corresponds to the maximum distance between theoretical arc and polygon segments as multiple of buffer.

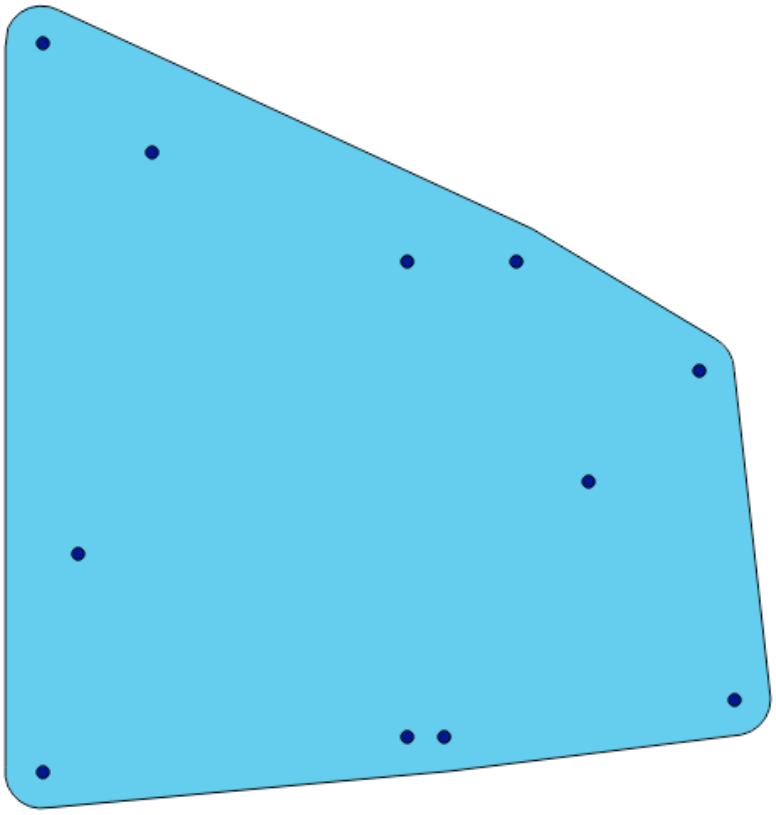
`v.out.ogr` converts the vector map containing the input points to an output format. In our case, there are two different output formats:

- Geography Markup Language (GML) encoded in XML
- Shapefile (SHP).

Parameter `dsn` allows specifying where the output files will be created. In your case, they will be saved in the `/tmp` folder.

`self.bufferOut.setValue` sets the name of the output GML file containing buffers, and `outputSHP` is the name of the output shapefile.

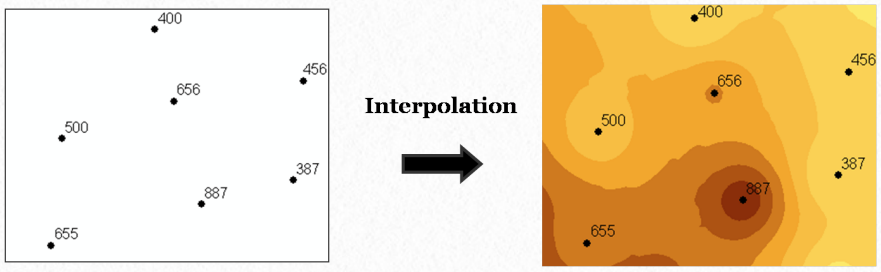
`v.hull` computes the convex hull of a vector map and outputs the convex hull polygon as a vector area map. The convex hull, or convex envelope, for an object or a set of objects is the minimal convex set containing the given objects. Convex hulls find useful applications in image processing, statistics, pattern recognition and mapping. In our case, the convex hull is generated from the buffered objects and represents the study area in the form of a polygon. A new shapefile called `outputHull` will be generated when consuming the process (this will be done in Chapter 8).



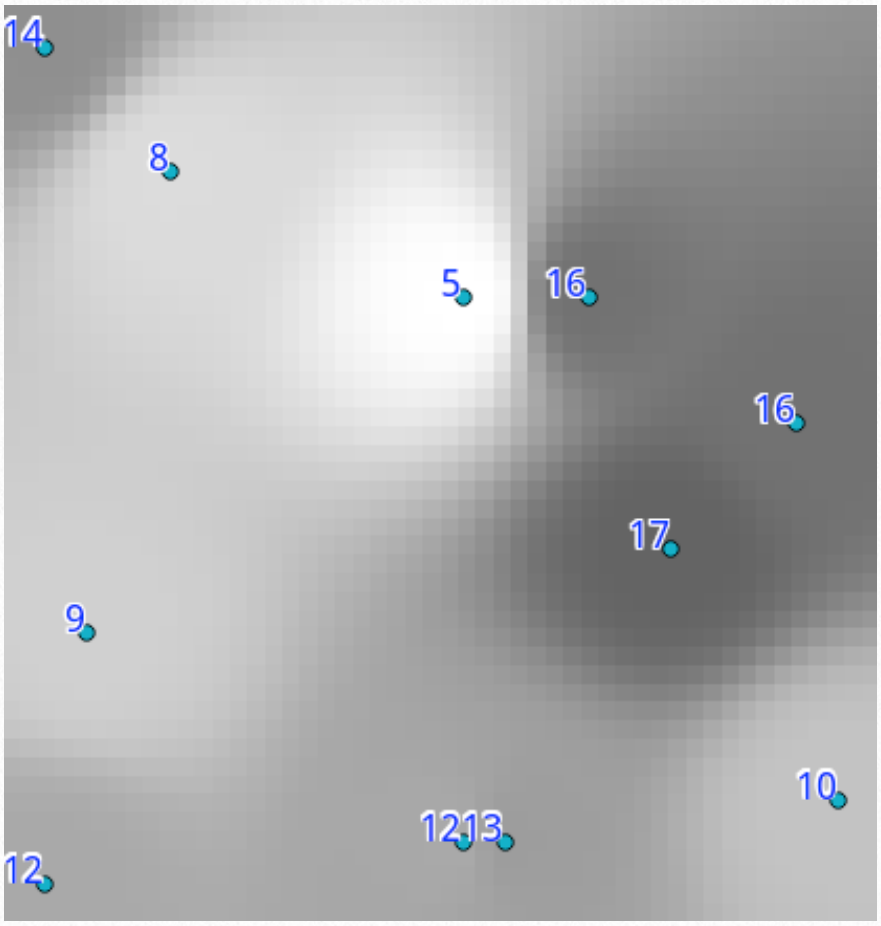
`g.region` manages the boundary definitions for the geographic region. In this exercise, we want to interpolate the input points to a raster map on the extent of the buffered objects, i.e. on an extent slightly larger than the input point layer. The `rows` and `cols` parameters specify how many rows and columns will be created in the output raster map. The higher the number of rows and columns, the higher the resolution of the raster.

`v.surf.idw` creates a surface interpolation from vector point data by Inverse Distance squared Weighting (IDW). This function fills a raster matrix (in our case a 50x50 cells matrix) with interpolated values generated from a set of irregularly spaced data points using numerical approximation (weighted averaging) techniques. The interpolated value of a cell is determined by values of nearby data points and the distance of the cell from those input points. In comparison

with other methods, numerical approximation allows representation of more complex surfaces (particularly those with anomalous features), restricts the spatial influence of any errors, and generates the interpolated surface from the data points.



In this exercise, the IDW interpolation surface will be computed on the extent defined previously by `g.region`. In the illustration below, the interpolation surface has been applied a greyscale color ramp and the input points are labelled with the contamination value stored in the `Measure` field.



The last two lines of code specify in which folder and under which name the output raster will be created: `/tmp` and `GeoTIFF` format.

You are done with writing the source code.

- You can now **close** the **gedit** application. **Save** when prompted.

The script will be consumed in Chapter 8. If it contains errors (e.g. syntax errors, spelling mistakes, bad indentation) you will not be able to make it work properly. A file with the correct syntax has been prepared for you. The file is entitled `Exercise5.py`. It is located in `/home/geoss/Desktop/data_exercises`. You may want to open this file with *gedit*, to copy the source code and to paste it in `processing.py`.

3. Publish the script on your local server

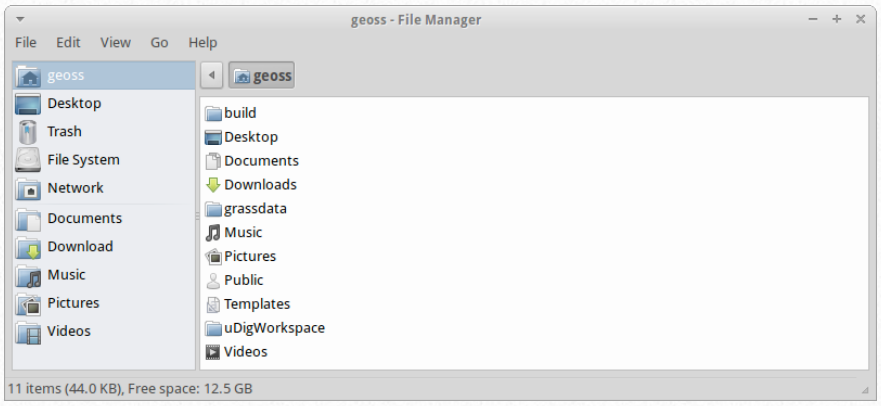
Publishing a PyWPS process requires storing specific information in the `processes` directory. This directory is namely `/usr/local/lib/python2.7/dist-packages/pywps-3.2.1-py2.7egg/pywps/processes`.

The `processes` directory must contain:

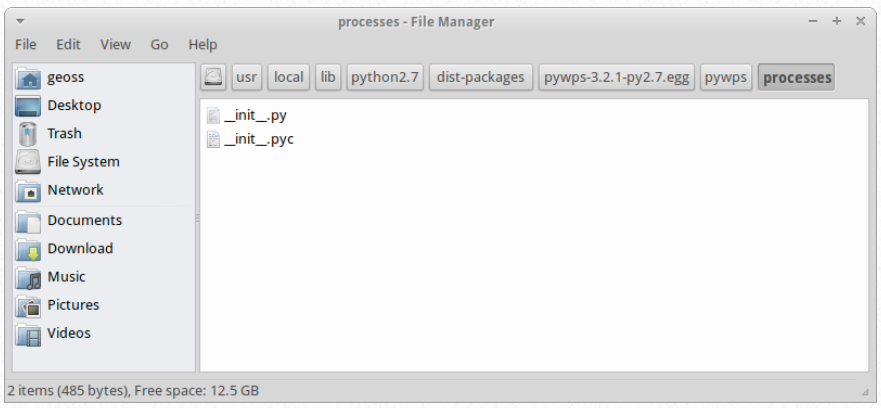
- An `__init__.py` file which sets the `__all__` array to contain the list of available variables. Specifically, `processing.py` must be declared in the `__init__.py` file.
- All Python scripts that you want to publish as web processing services, for example `processing.py`.

In what follows you will make sure that these two requirements are met.

- Click on **Applications Menu > Accessories > File Manager** to open the file manager:



- From the file manager, **navigate to /usr/local/lib/python2.7/dist-packages/pywps-3.2.1-py2.7.egg/pywps/processes**.



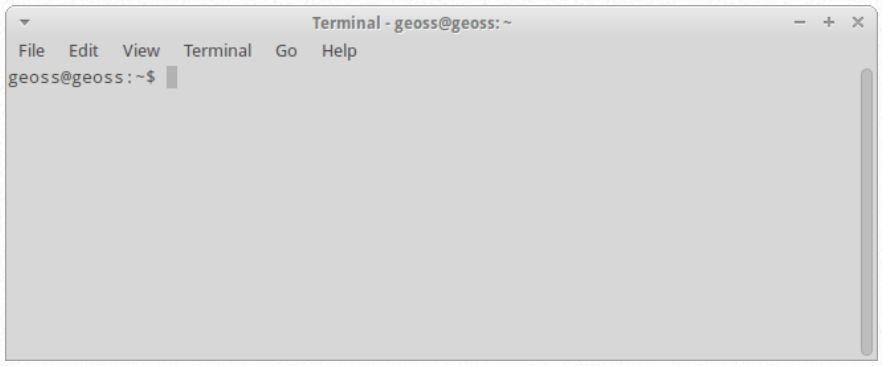
- **Right-click __init__.py > Open With > Open with gedit**. The file should open in a text editor. Ensure that "processing" is already present in the list:

```
ne","tests","processing","TI
```

It has been declared by the authors if the exercise.

- **Close** the text editor.

- Click on **Applications Menu > Accessories > Terminal Emulator**. You are in currently in the `geoss` folder.



- Enter the following command:

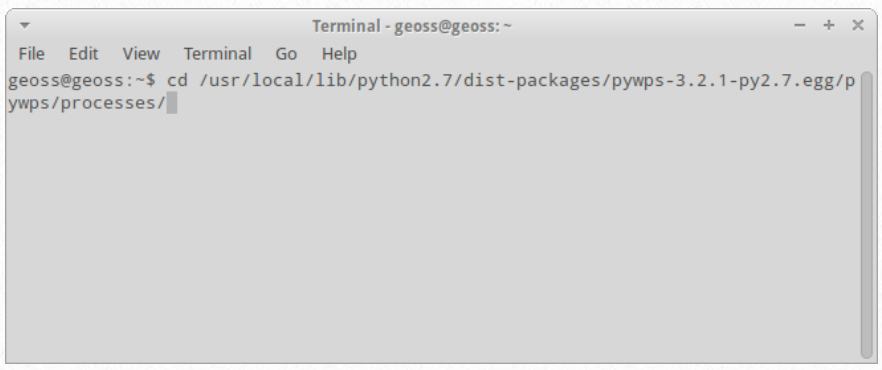
cd /usr/local/lib/python2.7/dist-packages/pywps-3.2.1-py2.7.egg/pywps/processes

Note that you can also perform this operation in several stages, e.g.:

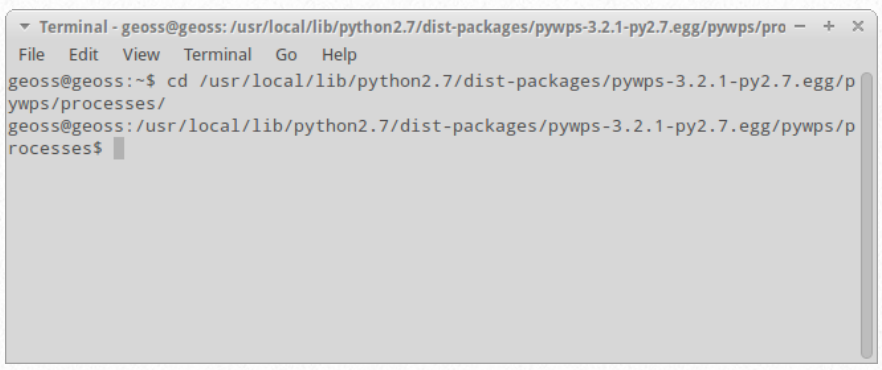
`cd /usr/local/lib/`

Then: `cd python2.7/dist-packages/pywps-3.2.1-py2.7.egg/`

Then: `cd pywps/processes`

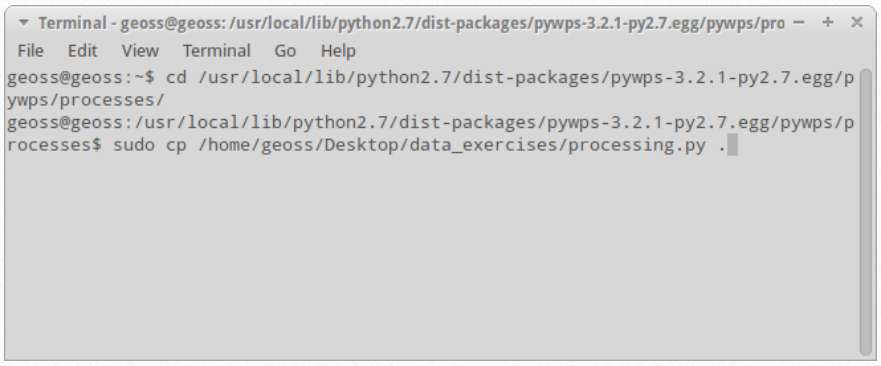


- Click **Enter** to validate. You are currently located in the `processes` directory.



- Enter the following command: **sudo cp /home/geoss/Desktop/data_exercises/processing.py .**

Do not forget the point at the end of the command.

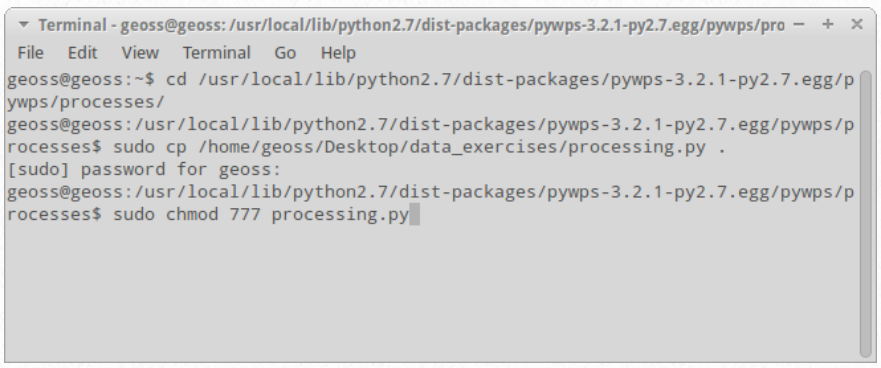


If prompted, enter the password **geoss**

Using the `sudo` command ensures that you are copying the file as a superuser. The `cp` command actually copies the file.

The `processes` folder now contains the Python file with the source code as well as a reference to this file in the `__init__.py` file.

- You need to assign read, write and execute privileges on your Python script. For that, enter the following command: **sudo chmod 777 processing.py**



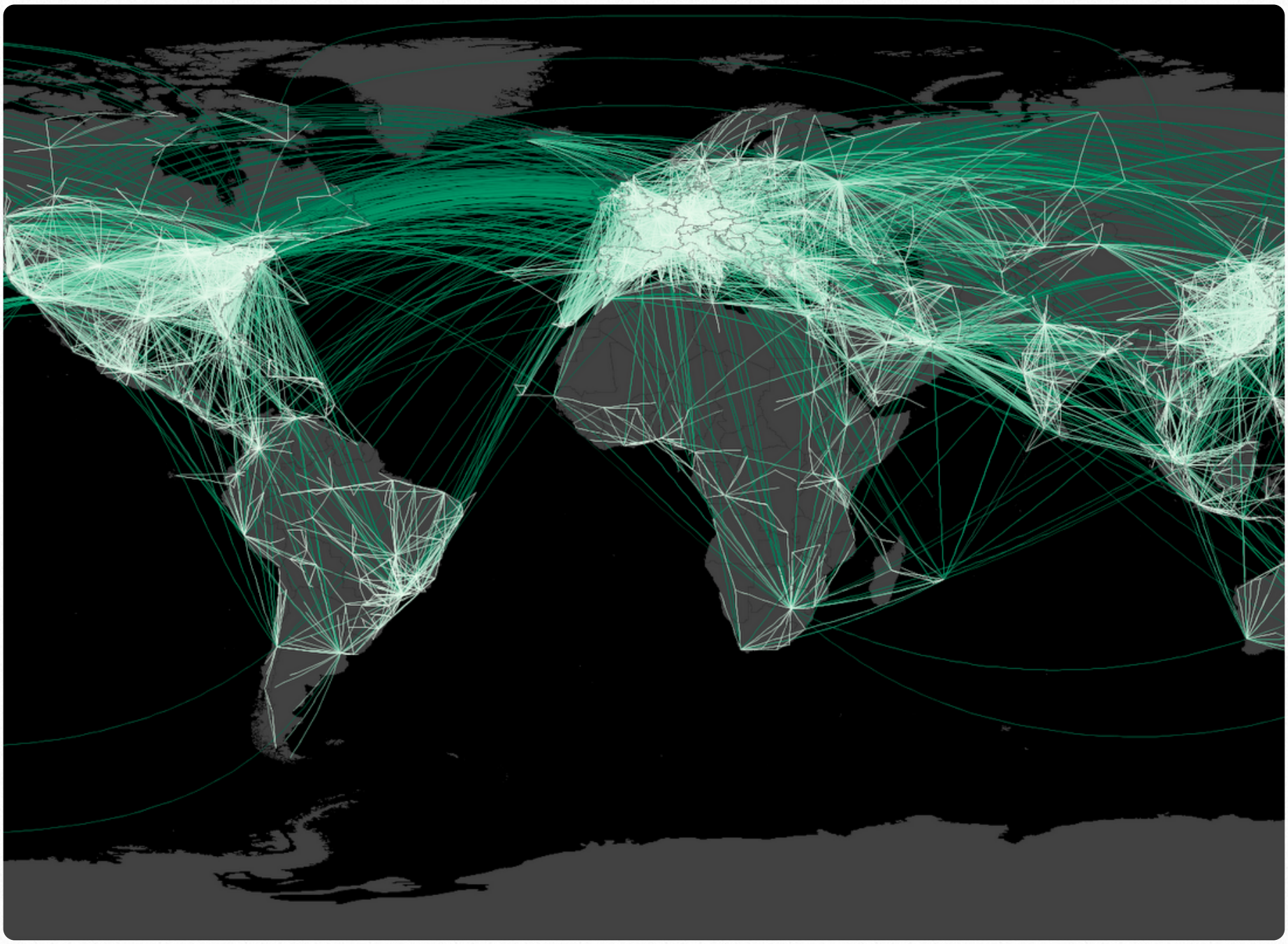
You just published a Web Processing Service on your local server. This service is ready to be consumed (executed) from a client. You will do this later in Chapter 8, through the QGIS WPS plugin client.

You can now **close** the **Terminal Emulator** as well as the **File Manager**.

6

How to visualize data?

Keywords: Google Earth; OpenLayers; QGIS; WMS



What you will learn:

- What a Web Map Service (WMS) is
- To use a WMS in OpenLayers
- To visualize a WMS in a client GIS (QGIS)
- To visualize a WMS in Google Earth

Definition of a WMS

The Web Map Service defines an interface that allows a client to retrieve maps of georeferenced data. In WMS context, a map means a graphical representation (jpeg ,gif or png files) of a geospatial data meaning that a WMS service does not give access to the data itself. It is used for mapping purposes and can be combined with other WMS services.

A traditional WMS interface, invoked by a URL, consists of the following operations:

- `GetCapabilites`: answer to a client telling him what kind of layers are available and which ones are queryable.
- `GetMap`: produce a map as a picture showing selected layers,
- `GetFeatureInfo`: answer simple queries about the content of the map

As seen on the following examples, invoking a WMS service need to specify different parameters (mandatory or optional) in the URL. For the purpose of this guideline we will focus our attention on the basic operations of the service that provides map layers in predefined styles (made available by the data provider) thus we will not discuss the Styled Layer Descriptor (SLD) capabilities.

Example of a WMS URL with a `GetCapabilities` request:

http://preview.grid.unep.ch:8080/geoserver/wms?request=GetCapabilities

The GetCapabilities operation returns to the user an XML document describing the service and the data sets available from which either desktop and/or web clients may request maps. This operation is common for all the OWS and is presented in details in the OpenGIS Web Service Common Implementation Specification (OGC, 2007). To invoke the operation, the user has only to define the service and the request parameters.


```
- <Layer queryable="1">
    <Name>preview:Cities_esri</Name>
    <Title>Cities_esri</Title>
    <Abstract/>
  - <KeywordList>
      <Keyword>features</Keyword>
      <Keyword>esri2003_cities</Keyword>
    </KeywordList>
    <CRS>EPSG:4326</CRS>
    <CRS>CRS:84</CRS>
  - <EX_GeographicBoundingBox>
      <westBoundLongitude>-176.151565551758</westBoundLongitude>
      <eastBoundLongitude>179.221893310547</eastBoundLongitude>
      <southBoundLatitude>-54.7919998168945</southBoundLatitude>
      <northBoundLatitude>78.1999969482422</northBoundLatitude>
    </EX_GeographicBoundingBox>
    <BoundingBox CRS="CRS:84" minx="-176.151565551758" miny="-54.7919998168945"
    maxx="179.221893310547" maxy="78.1999969482422"/>
    <BoundingBox CRS="EPSG:4326" minx="-54.7919998168945" miny="-176.151565551758"
    maxx="78.1999969482422" maxy="179.221893310547"/>
  - <Style>
      <Name>point</Name>
      <Title>Default point</Title>
    - <Abstract>
        A sample style that just prints out a 6px wide red square
      </Abstract>
    - <LegendURL width="20" height="20">
        <Format>image/png</Format>
        <OnlineResource xlink:type="simple" xlink:href="http://preview.grid.unep.ch:8080
        /geoserver/ows?service=WMS&request=GetLegendGraphic&format=image%2Fpng&
        width=20&height=20&layer=Cities_esri"/>
      </LegendURL>
    </Style>
  </Layer>
```


Screenshot of a WMS GetCapabilities

The WMS GetCapabilities screenshot above shows for each dataset a few elements sur-

rounded by tags. These elements are described below:

- `<Name>`: the layer's system name
- `<Title>`: the layer's display name
- `<Abstract>`: the layer's description
- `<KeywordList>`: the various keywords contained, each between the `<Keyword>` tag
- `<CRS>`: the Coordinate Reference System used for the layer in question, expressed both as EPSG number and CRS number
- `<EX_GeographicBoundingBox>`: the layer's limits of the enclosing rectangle in longitude and latitude decimal degrees. Each rectangle coordinate is encompassed in the corresponding tag: `<westBoundLongitude>`, `<eastBoundLongitude>`, `<southBoundLatitude>` and `<northBoundLatitude>`.
- `<BoundingBox>`: the layer's limits of the bounding box in units of the specified coordinate reference system. In the same tag, the following attributes are indicated: `CRS`, `minx`, `miny`, `maxx` and `maxy`.
- `<Style>`: The defined style to be used. It is itself defined by several tags:
  - `<Name>`: the style's system name
  - `<Title>`: the style's display name
  - `<Abstract>`: the style's description
- `<LegendURL>`: the information for the legend's display:
  - `<Format>`: the Internet Media Type (MIME type) of the legend. E.g. image.
  - `<OnlineResource>`: The URL of the legend's style.

Example of a WMS URL with a **GetMap** request:

http://preview.grid.unep.ch:8080/geoserver/wms?bbox=-178.21655,18.92548,-66,71.35144&styles=&Format=image/png&request=GetMap&version=1.1.1&layers=preview:cy_buffers,preview:Admin1&width=640&height=309&srs=EPSG:4326

The GetMap operation is the most important of the three basic operations of the WMS interface as it returns to a client request a map of selected geospatial layers.

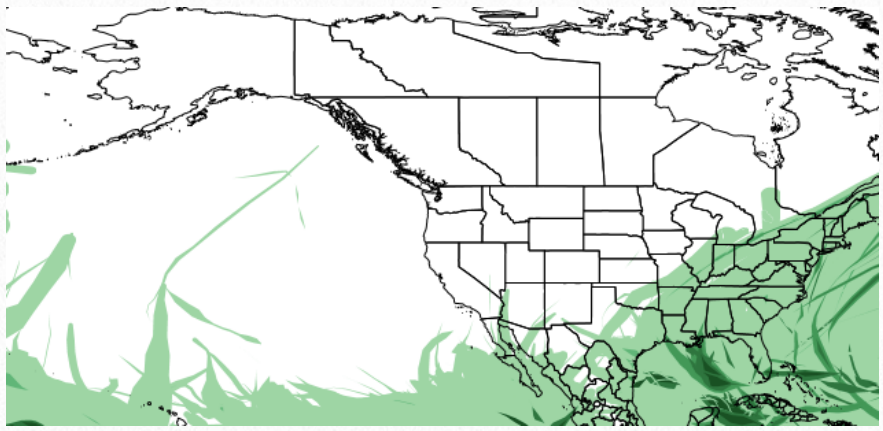
In comparison of the GetCapabilities request that needs only two parameters, we can see on the above example that the GetMap operation needs several parameters (also mandatory or optional) that we describe hereafter:

Mandatory parameters for the GetMap operation:

- BBOX: coordinates of the bounding box following minx,miny,maxx,maxy,
- STYLES: list of style names separated by a comma. It's necessary to have an exact corre-

spondence between the number of layers and the number of styles. If this parameter has an empty value, the default style provided by the data custodian will be applied.

- FORMAT: graphical format of the returned map (eg: image/png, image/gif, image/jpeg).
- REQUEST: value “GetMap”, this the request itself to invoke the specific operation.
- VERSION: the version of the specification.
- LAYERS: list of selected layers separated by a comma.
- WIDTH: specify the width of the returned map (in pixels)
- HEIGHT: specify the height of the returned map (in pixels)
- SRS: identifier of the Spatial Reference System.



Returned image after a GetMap request

Example of a WMS URL with a `GetFeatureInfo` request:

http://ogi.state.ok.us/geoserver/wms?SERVICE=WMS&VERSION=1.1.1&REQUEST=GetFeatureInfo&SRS=EPSG:4326&BBOX=-104.5005,32.7501,-94.01,37.20&WIDTH=800&HEIGHT=300&LAYERS=ogi:okcounties&QUERY_LAYERS=ogi:okcounties&STYLES=&X=550&Y=105

The GetFeatureInfo operation is used to query the attribute table of a selected layer and get information on a specific feature. For example, a user can click on point of a map (retrieved by a GetMap request) and he obtains more information.

Mandatory parameters for the GetFeatureInfo operation:

- VERSION: the version of the specification.
- REQUEST: value “GetFeatureInfo”, this the request itself to invoke the specific operation.
- LAYERS: list of selected layers separated by a comma.

- STYLES: list of style names separated by a comma. It's necessary to have an exact correspondence between the number of layers and the number of styles. If this parameter has a empty value, the default style provided by the data custodian will be applied.

- SRS: identifier of the Spatial Reference System.

- BBOX: coordinates of the bounding box following minx,miny,maxx,maxy.

- FORMAT: the format of the returned information (text/xml, text/html, text/plain)

- X,Y: coordinates of the clicked point on the map (in pixels). The origin is at the upper left corner. The X,Y coordinate must be inside the BBOX extent.

- QUERY_LAYERS: list of selected layers to query separated by a comma.

```
Results for FeatureType 'http://ogi.state.ok.us:okcounties':
--------------------------------------------
statefp = null
countyfp = null
countyns = null
cntyidfp = null
name = OKLAHOMA
namelsad = null
lsad = null
classfp = null
mtfcc = null
ur = null
funcstat = null
the_geom = [GEOMETRY (MultiPolygon) with 569 points]
state = 40
county = 109
stateplane = N
--------------------------------------------
```

Result of a GetFeatureInfo query

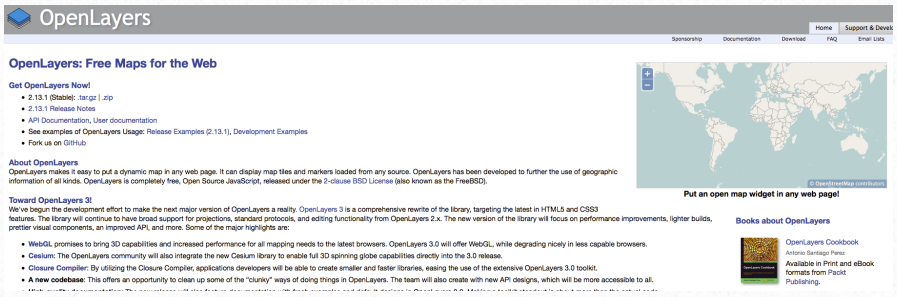
How to use a WMS in OpenLayers?

OpenLayers is an open source (provided under a modified Berkeley Software Distribution (BSD) license) JavaScript library for displaying map data in web browsers. It provides an API for building rich web-based geographic applications similar to Google Maps and Bing Maps. (Source: Wikipedia)

The following simple steps allow to display WMS layers in OpenLayers:

1. Download OpenLayers

On the OpenLayers homepage there is a link to download the latest stable release:



2. Install OpenLayers

Once you have downloaded the zip file of the latest stable release, you normally need to unzip it and extract the `OpenLayers.js` file and `img` directory to your local disk. For this exercise we have already done it for you and they are located under `/home/GEOSS/Desktop/data_exercises/OpenLayers` of your virtual machine.

3. Build a simple dynamic map web page

3.1 Creating your first map

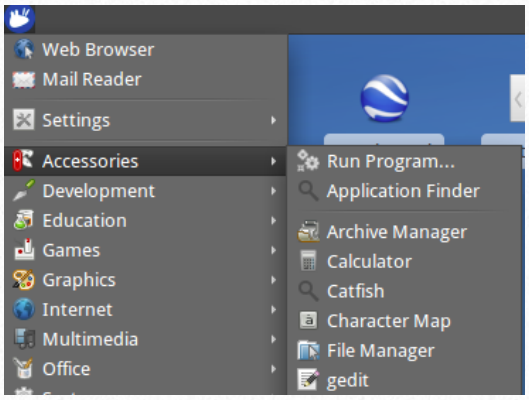
The OpenLayers API has two concepts which are important to understand in order to build your first map: 'Map', and 'Layer'. An OpenLayers Map stores information about the default projection, extents, units, and so on of the map. Inside the map, data is displayed via Layer's. A Layer is a data source – information about how OpenLayers should request data and display it.

3.2 Crafting HTML

Building an OpenLayers viewer requires crafting HTML in which your viewer will be seen. OpenLayers supports putting a map inside of any block level element – this means that it can be used to put a map in almost any HTML element on your page.

In addition to a single block level element, it is also required to include a script tag which includes the OpenLayers library to the page.

- Open a text editor (e.g. `gedit`, that is accessible by clicking **Accessories > gedit** of your virtual machine)



- **Copy the block of code below**:

```
<html>
   <head>
      <title>OpenLayers Example</title>
      <script
        src="http://openlayers.org/api/OpenLayers.js">
      </script>
   </head>
   <body>
      <div style="width:100%; height:100%" id="map">
      </div>
   <script defer="defer" type="text/javascript">

  //Here you will write later the rest of the code

     </script>
   </body>
</html>
```

It is important to respect the case (lower/upper case). For example the OpenLayers.js file has a capital ‘O’ and a capital ‘L’. Also ensure that quotes and quotation marks are straight and not oblique. For example:

“ is not correct

" is correct

- **Save** the document as **map.html** in the same directory as the files you unzipped.

3.3 Creating the map viewer

In order to create the viewer, you must first create a map. The OpenLayers.Map constructor requires one argument: This argument must either be an HTML Element, or the ID of an HTML element. This is the element in which the map will be placed.

```
var map = new OpenLayers.Map('map');
```

Here again and through all the script it is important to respect lower/upper case. For example OpenLayers has a capital ‘O‘ and a capital ‘L’.

The next step to creating a viewer is to add a layer to the Map. OpenLayers supports many different data sources, from WMS to Yahoo! Maps to WorldWind. In this example, the WMS layer is used. The WMS layer is an example provided by MetaCarta.

```
var Open_Layers = new OpenLayers.Layer.WMS(
   "OpenLayers WMS",
   "http://vmap0.tiles.osgeo.org/wms/vmap0",
   {layers:'basic'},
   {displayInLayerSwitcher:'true'});
map.addLayer(Open_Layers);
map.zoomToMaxExtent();
```

The first parameter in this constructor is the URL of the WMS server. The second argument is an object containing the parameters to be appended to the WMS request.

Finally, in order to display the map, you must set a center and zoom level. In order to zoom to fit the map into the window, you can use the `zoomToMaxExtent` function, which will zoom as close as possible while still fitting the full extents within the window. Note that zoomToMaxExtent has a capital 'T', a capital 'M' and a capital 'E' while 'z' is in lower case.

3.4 Putting it all together

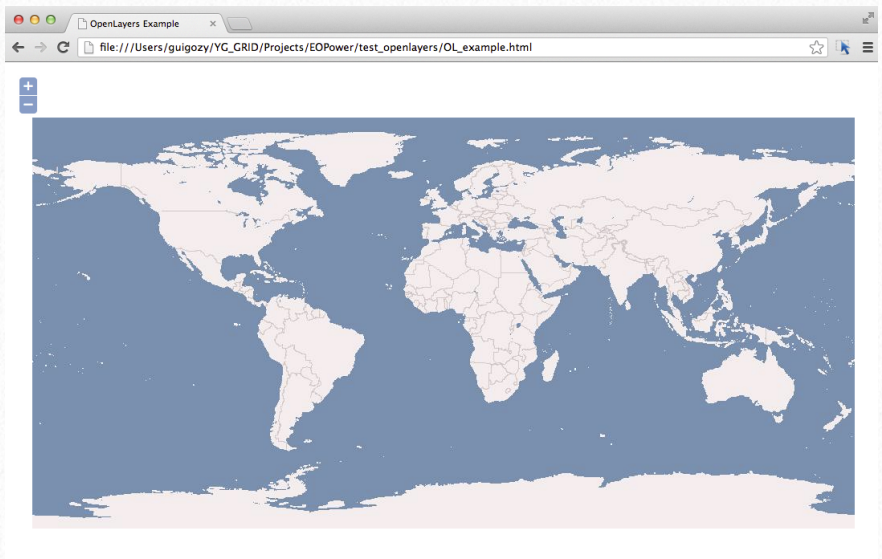
The following code block puts all the pieces together to create an OpenLayers viewer:

```
<html>
   <head>
      <title>OpenLayers Example</title>
      <script
        src="http://openlayers.org/api/OpenLayers.js">
      </script>
   </head>
   <body>
      <div style="width:100%; height:100%" id="map">
      </div>
   <script defer="defer" type="text/javascript">
     var map = new OpenLayers.Map('map');
     var Open_Layers = new OpenLayers.Layer.WMS(
         "OpenLayers WMS",
         "http://vmap0.tiles.osgeo.org/wms/vmap0",
         {layers:'basic'},
         {displayInLayerSwitcher:'true'});
     map.addLayer(Open_Layers);
     map.zoomToMaxExtent();
   </script>
   </body>
</html>
```

- Save your changes in `gedit` (**File > Save**).

- From the **File manager**, **double-click map.html**. A map should open in your browser.

If it does not ask your trainer or refer to the `OpenLayers_example.html` file in your data folder. This file contains the complete source code of the exercise.



3.5 Adding an overlay WMS

WMS layers have the capability to be overlaid on top of other WMS layers in the same projection. There are several ways to mark a layer as an overlay, rather than a base layer. With WMS, the best way to do this is by setting the `transparent` parameter to `true`. The example here uses two WMS layers (one vector and one raster) to demonstrate overlaying a transparent WMS.

```
var CY_risk = new OpenLayers.Layer.WMS(
"Cyclones risk",
"http://preview.grid.unep.ch:8080/geoserver/wms?layers
=cy_risk",
{transparent:'true'});

var CY_buffers = new OpenLayers.Layer.WMS(
"Cyclones buffers",
```

```
"http://localhost:8080/geoserver/wms?layers=cy_buffers
", {transparent:'true'});

map.addLayers([CY_buffers, CY_risk]);
```

Using the `transparent: 'true'` parameter sets the WMS layer to `image/png` if the browser supports transparent PNG images (all browsers except Internet Explorer 6). This also allows the layer to be displayed at the same time as other layers.

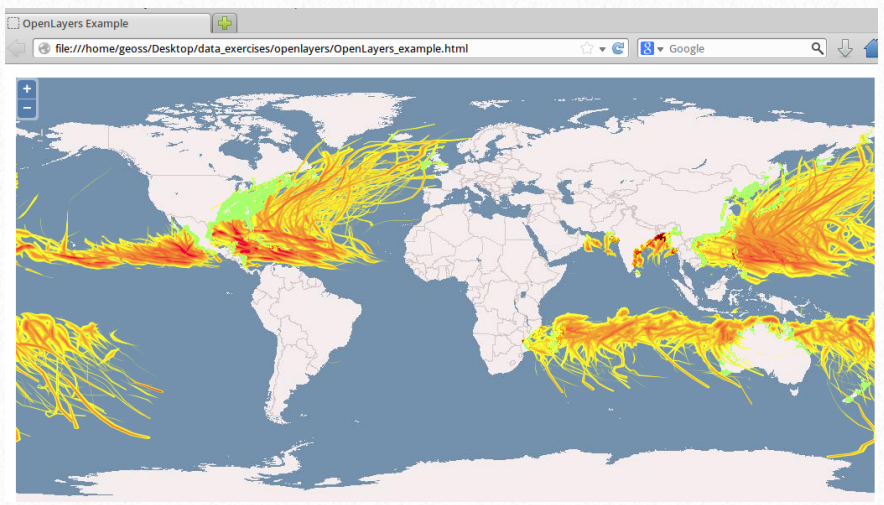
In the last line of code you have used `addLayers` on the map object to add several layers at the same time. This allows you to save one line of code in this case, and may be useful in other cases when you need to add multiple layers to the map at the same time.

Putting this code together with our earlier example, you will get the following:

```
<html>
   <head>
      <title>OpenLayers Example</title>
      <script
        src="http://openlayers.org/api/OpenLayers.js">
      </script>
   </head>
   <body>
      <div style="width:100%; height:100%" id="map">
      </div>
   <script defer="defer" type="text/javascript">
      var map = new OpenLayers.Map('map');
      var Open_Layers = new OpenLayers.Layer.WMS(
         "OpenLayers WMS",
         "http://vmap0.tiles.osgeo.org/wms/vmap0",
         {layers:'basic'},
         {displayInLayerSwitcher:'true'});

//Here you will write some code in Section 4.4

     var CY_risk = new OpenLayers.Layer.WMS(
      "Cyclones risk",
      "http://preview.grid.unep.ch:8080/geoserver/wms?
       layers=cy_risk",
         {transparent:'true'});
      var CY_buffers = new OpenLayers.Layer.WMS(
          "Cyclones buffers",
          "http://localhost:8080/geoserver/wms?
           layers=cy_buffers",
         {transparent:'true'});
      map.addLayers([Open_Layers,CY_risk,CY_buffers]);
      map.zoomToMaxExtent();

//Here you will write some code in 4.1, 4.2 and 4.3

   </script>
   </body>
</html>
```

Note that the `addLayers` function is now used with three layers: Open_Layers, CY_risk and CY_buffers. This will give the code more visibility than using `addLayer` three times.

- Save your changes (**File > Save**).
- **Double-click** the **map.html** file (or refresh the map in your browser). You should obtain:



4. Working with controls

4.1 Create an overview map

Online maps often contain a smaller overview map that displays the extent of the larger map. In OpenLayers, this is possible using the `OpenLayers.Control.OverviewMap` control.

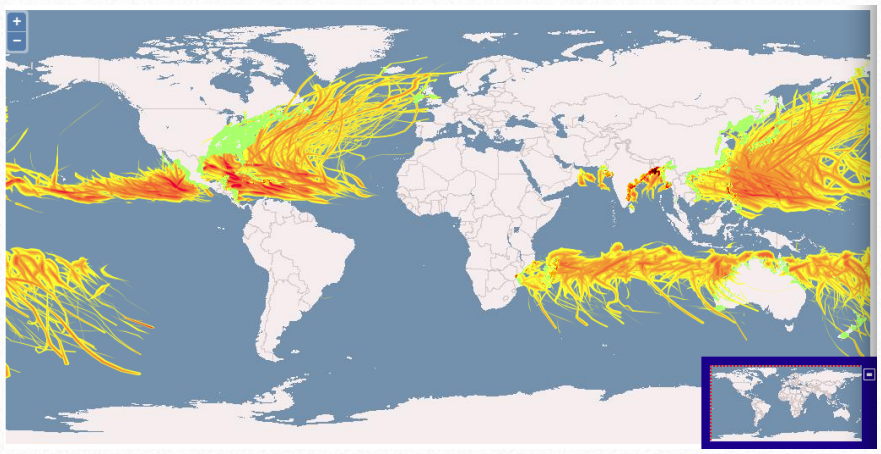
You are now going to add an overview map with default options.

- Add the following **after `map.zoomToMaxExtent()`**:

```
var overview = new OpenLayers.Control.OverviewMap();
map.addControl(overview);
```

- **Save your changes** and **refresh** your map in your browser.

- To see the overview map in action, open the **plus sign** at right bottom of the map viewport.



4.2 Create a scale line

- Just after the code for the map overview, **add the following code** to create a new scale line control for your map:

```
var scaleline = new OpenLayers.Control.ScaleLine();
map.addControl(scaleline);
```

- **Save your changes** and **refresh** the map in your browser.

A scale bar is now displayed at the left bottom of your window.

4.3 Add other controls

In this exercise, we will add a few other useful controls: a layer switcher, a mouse position, a pan panel and a zoom panel.

- Below the code you had added for the scaleline, **add the following code**:

```
var layer_switcher = new
OpenLayers.Control.LayerSwitcher();
map.addControl(layer_switcher);

var mouse_position = new
OpenLayers.Control.MousePosition();
map.addControl(mouse_position);

var pan_panel = new OpenLayers.Control.PanPanel();
map.addControl(pan_panel);

var zoom_panel = new OpenLayers.Control.ZoomPanel();
map.addControl(zoom_panel);
```

- **Save your changes** and **refresh** the map in your browser to see the new functions displayed in the map.

4.4 Add other base maps

In this section of the exercise, you will add a few other useful base maps, as some background maps are sometimes more appropriate than others:

- Below the code you had added for the Metacarta base map in section 3.3 , **add the following code**:

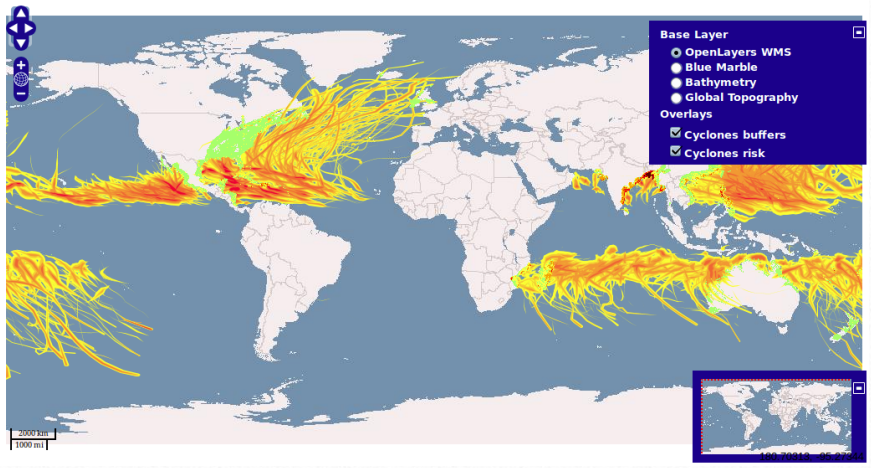
```
var Blue_Marble = new OpenLayers.Layer.WMS(
    "Blue Marble",
    "http://neowms.sci.gsfc.nasa.gov/wms/wms?
    layers=BlueMarbleNG",
    {layers:'basic'},{displayInLayerSwitcher:'true'});

var Bathymetry = new OpenLayers.Layer.WMS(
   "Bathymetry",
   "http://www.gebco.net/data_and_products/
    gebco_web_services/web_map_service/mapserv?
    layers=gebco_08_grid",
    {layers:'basic'},{displayInLayerSwitcher:'true'});

var Global_Topography = new OpenLayers.Layer.WMS(
    "Global Topography",
```

```
"http://neowms.sci.gsfc.nasa.gov/wms/wms?
layers=SRTM_RAMP2_TOPO",
{layers:'basic'},{displayInLayerSwitcher:'true'});
```

Note that all these layers are defined as basic and will appear in the Layer Switcher. You will also need to add these new variables into the map.addLayers command. This command will become:

```
map.addLayers([Open_Layers,Blue_Marble,Bathymetry,Glob
al_Topography,CY_buffers,CY_risk]);
```

- **Save** your HTML file and **refresh** the map in your browser. You should obtain the final map:



If you had some problems to do this exercise, a correction has been prepared for you in

`/home/GEOSS/Desktop/data_exercises/OpenLayers/OpenLayers_example.html`.

- You can now **close** your **web browser**.

How to visualize a WMS in a client GIS (QGIS)?

Most GIS client softwares allow you to visualize WMS layers. In this exercice, you will be using the Quantum GIS software (cf. Chapter 5). Here is the procedure to follow to display WMS layers in QGIS:

- Click on **Applications Menu**. This is the blue button on the upper left of your virtual machine:

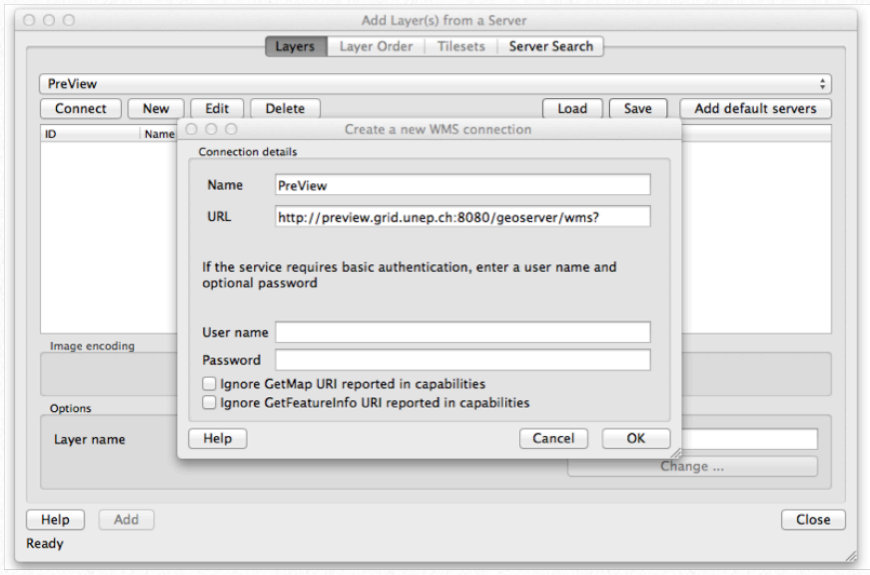
- Select **Education > Quantum GIS Desktop**.

A new empty project opens.

- Click on the **Add WMS Layer** tool:

• Click **New** and name this connection: **Preview - WMS**.

• **Paste** the URL: `http://preview.grid.unep.ch:8080/geoserver/wms?`

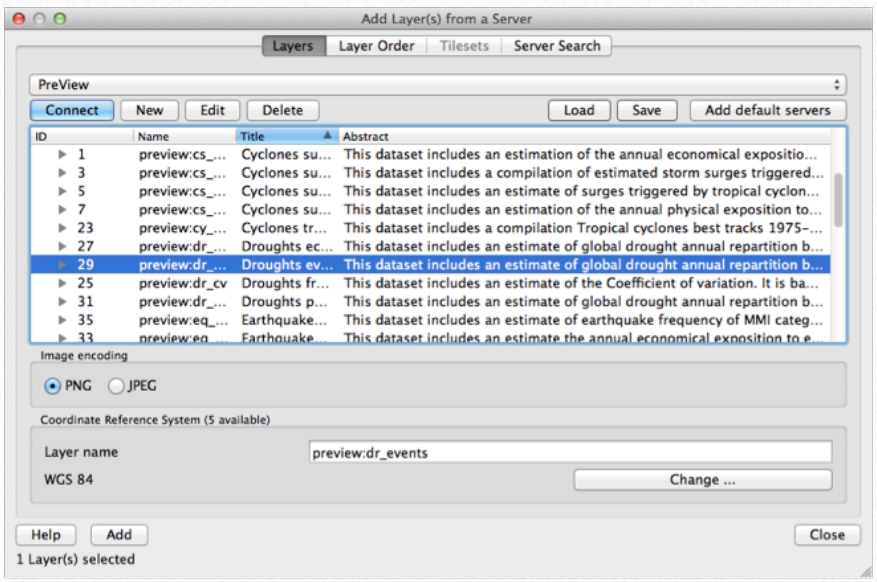


• Click **OK**.

• Click **Connect** (it might take a moment for the data layers list to load).

• Click **Title** to sort the layers by title.

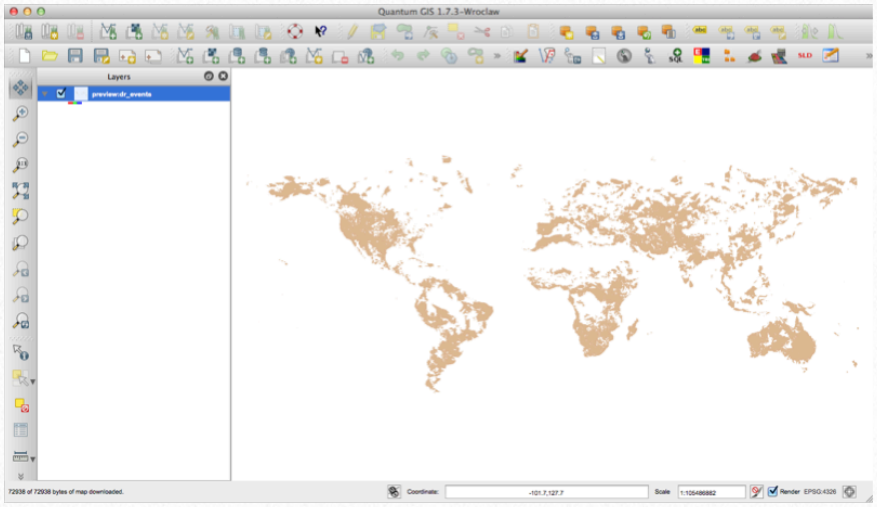
• Click the layer entitled "**Droughts events**":



• Click **Add** (this will add the layer to the map document)

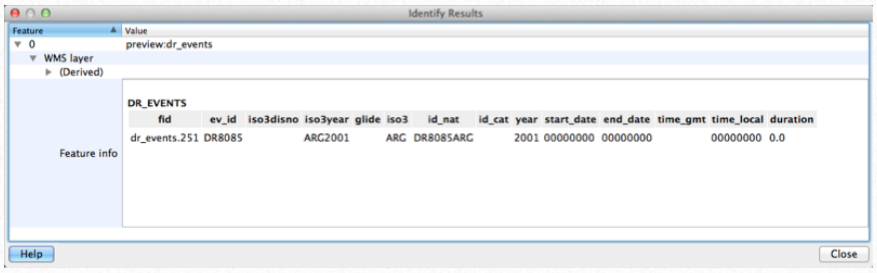
• **Close** the dialog box

• **Navigate** through your map (Zoom IN/OUT, Pan, etc..):



• Click the "**Identify features**" button:

• **Click** on a **drought event** and **have a look** at the query result:



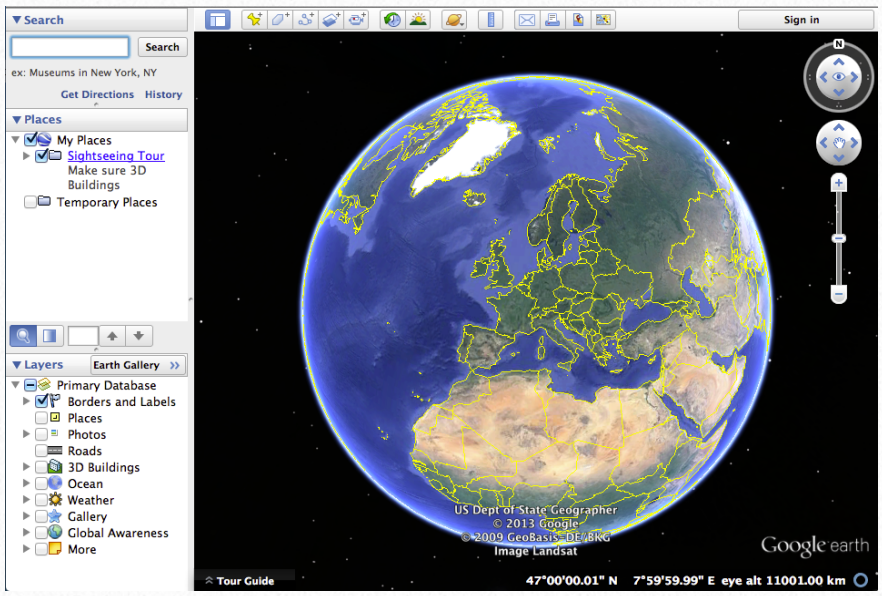
With the WMS, users receive only images of the data and not the data itself.

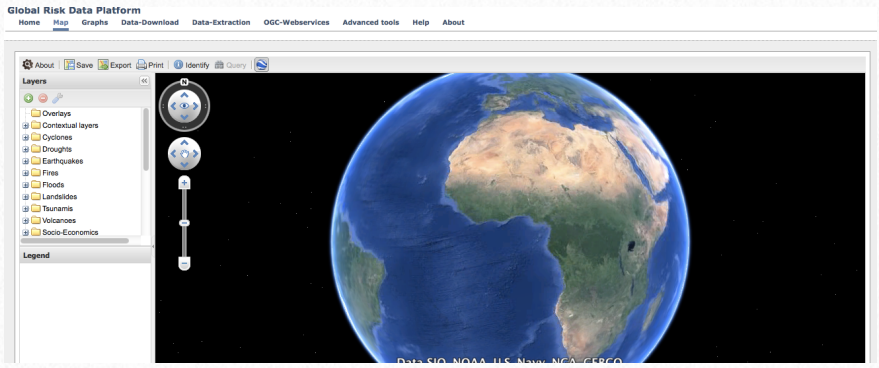
Data can be queried but for example attribute table cannot be viewed.

- **Close** the Quantum GIS application. When prompted, select **Close without Saving**.

How to visualize a WMS in Google Earth?

As just seen in the previous sections, several possibilities exist for visualizing WMS layers. One of the most popular viewers is Google Earth, which is a virtual globe program. It maps the earth by the superposition of images obtained from satellite imagery, aerial photography and GIS 3D globe. It can be downloaded from here as a software but very often it is directly embedded in a website, which is for example the case in the Preview website where you have a specific button allowing you to directly activate Google Earth inside the application.





In this exercise, we will use the Google Earth Software that can be launched from the desktop of your virtual machine.

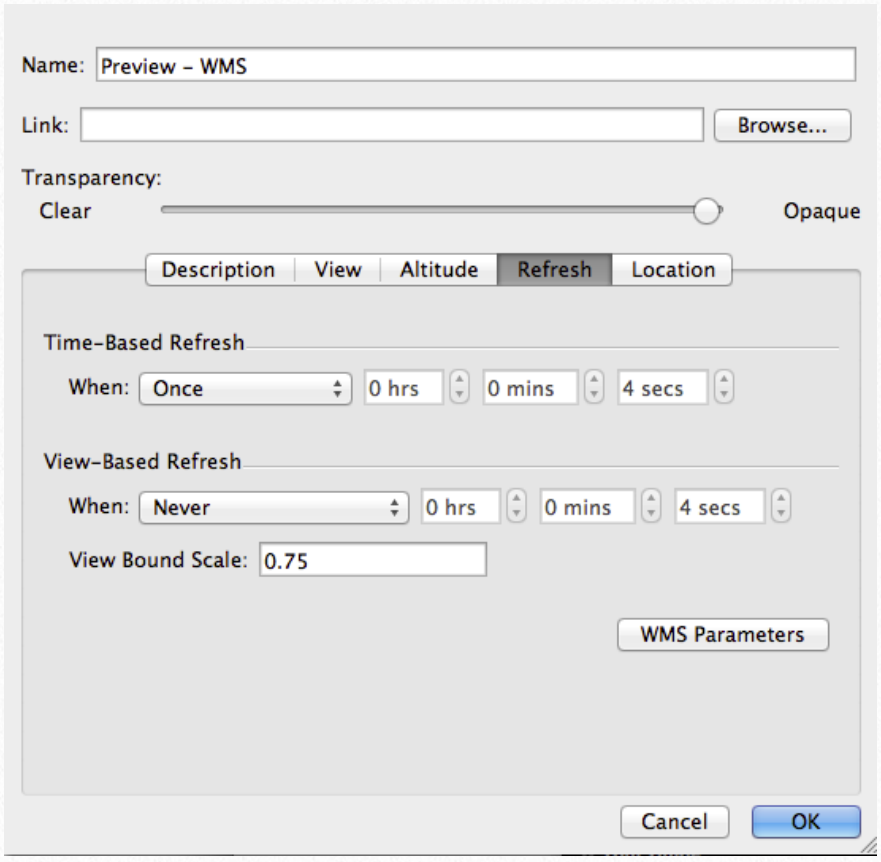
- **Double-click** on the **Google Earth** icon:

Follow then the procedure below:

- **Click** on the menu item **Add** and choose **Image Overlay** or use the corresponding icon:

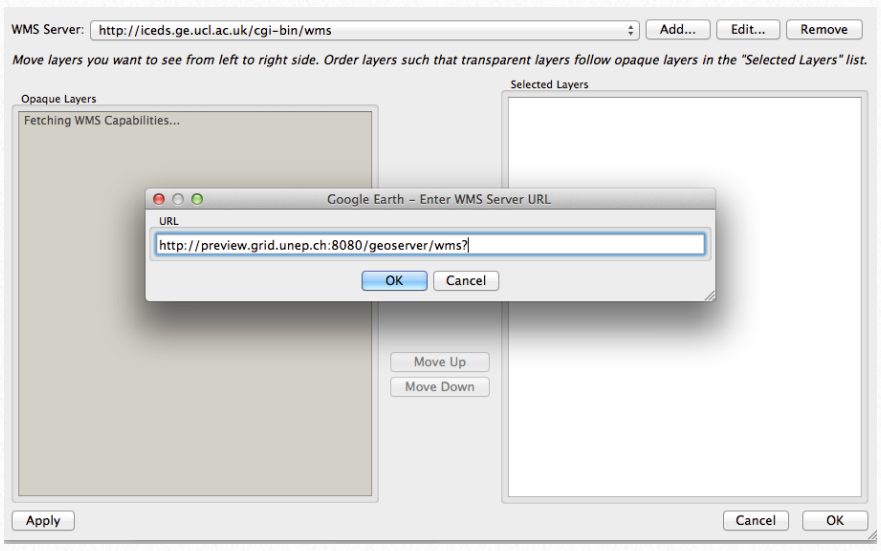
- Enter a **name** for the layer, e.g. **Preview - WMS**.

- Click on the **Refresh** tab of the window and click on the **WMS Parameters** button at the bottom right of this window.



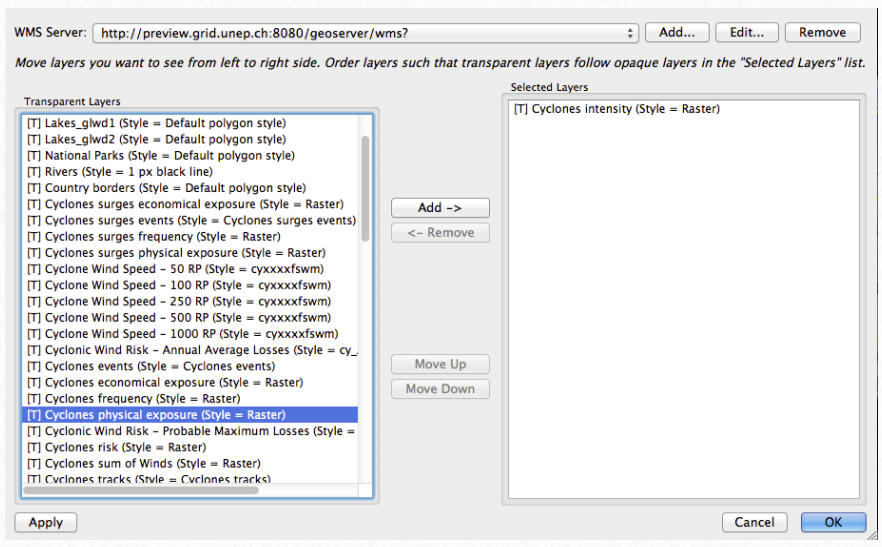
- This will open a new window. Click on **Add** and **insert the Preview WMS server url** as follows: `http://preview.grid.unep.ch:8080/geoserver/wms?`

- Click **OK**.



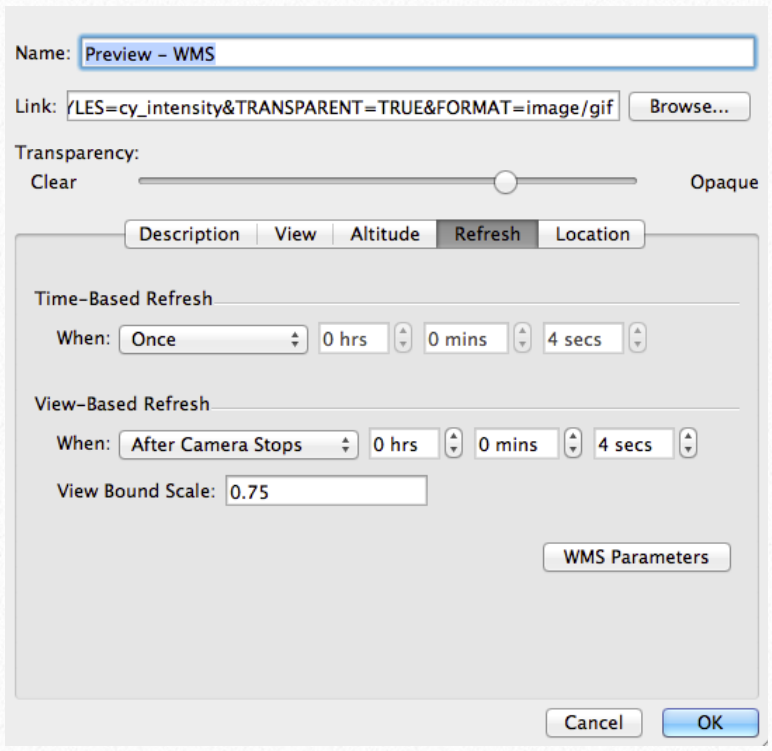
You will see in the left pane all WMS layers available from the Preview server.

- **Select** for example the one **`Cyclone intensity`**

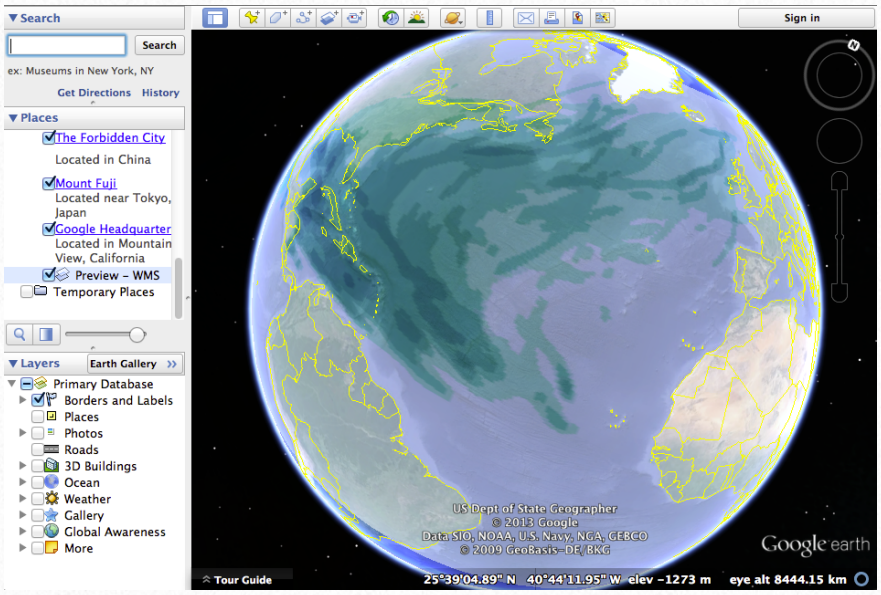


- Click on the **Add** button, which will add this layer to the right pane and click **OK**.

You will see the layer in question appearing in the `My Places` menu and overlaying the globe. You can play on the layer's transparency with the **transparency slider** to better see the background.



- When you are satisfied with the transparency, click **OK**:



If you want to modify again the transparency of this layer, right-click on its name and choose **Properties**. This will display again the window where you can control the WMS parameters.

If you want to remove this layer and add other ones, right-click on the layer's name, choose **delete** and confirm with **OK.** Start then again the WMS layer selection process as just described.

- **Close** Google Earth without saving.

7

How to download data?

Keywords: download; plugins; WCS, WFS

What you will learn:

- Download WFS data in a GIS client
- Edit vector layer using WFS-T

Recommended readings:

- http://www.postgis.us
- PostGIS documentation

Downloading and editing data in a desktop GIS client

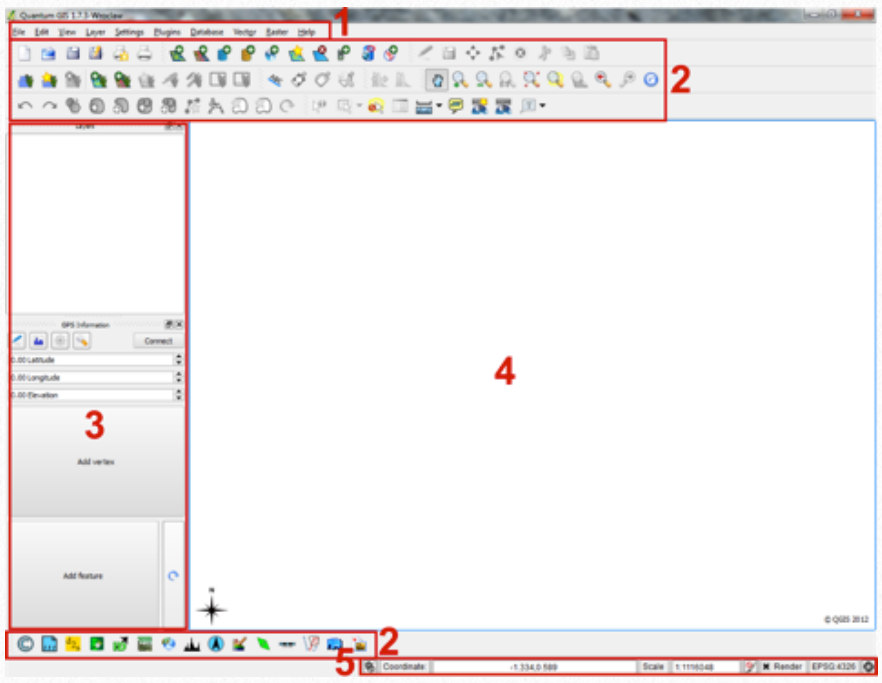
QGIS interface and customization

In this Section you will get familiar with the Quantum GIS graphical interface and customize it to add/remove new functions.

- Start QGIS by selecting **Applications menu > Education > Quantum GIS Desktop**.

QGIS graphical interface is divided in 5 zones:

1. Menu bar,
2. Toolbars to facilitate the access to the different functions (also available from the menu bar),
3. Table of Content (ToC) to manage the layers and access specific functions,
4. View, where you can interact with the map,
5. Status bar where you can see information such as coordinates of the pointer, extent, scale, coordinate system; start/stop rendering or define how the view is displayed.



Toolbars are divided by category (greyed icons means they are inactive because the appropriate conditions to use them are not fulfilled). Some of them are included by default in QGIS, some others are plugins that can be added/removed from the interface:

1. File

2. Manage Layers

3. Map Navigation

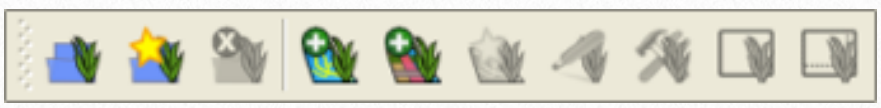
4. Attributes

5. Label

6. Raster

7. Digitizing

8. GRASS plugin

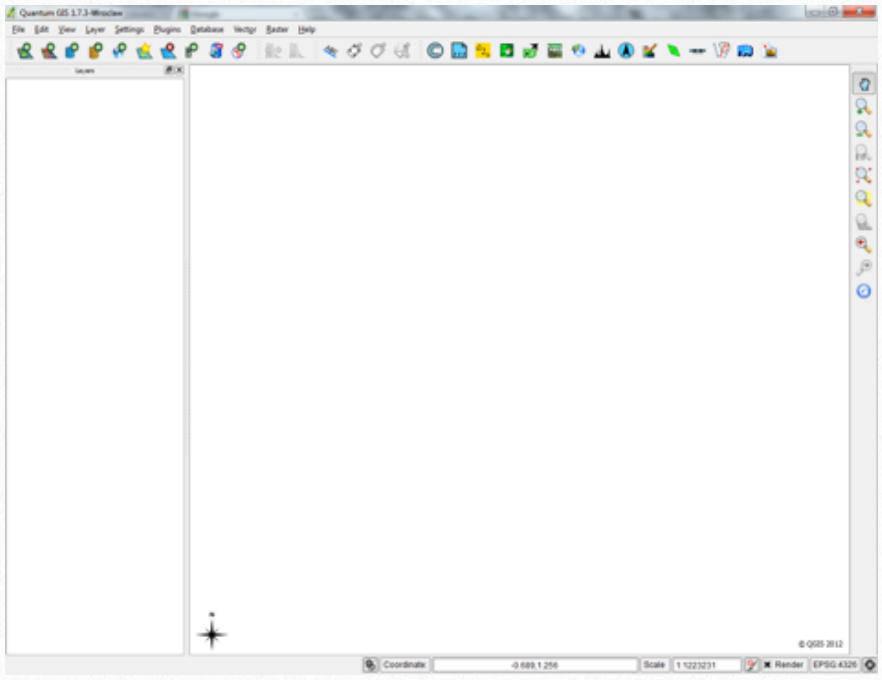


9. Advanced Digitization

10.Plugins

We will not describe here the functionalities of each function, some of them will be used during this training, you will discover the remaining ones on your own as long as you practice.

The toolbars can be added/removed from the interface by right clicking in any empty area of any toolbar or menu and enable/disable them. They can also be moved using drag and drop (point your mouse on the vertical separation line between 2 categories until it becomes a white cross; you can then drag and drop). The Figure below shows an example of organization of the interface.



More information can be found on the QGIS User Manual at: http://www.qgis.org/en/documentation/manuals.html

Once you get something satisfying, close QGIS and Start it again, as you can see the interface remains as you arranged it.

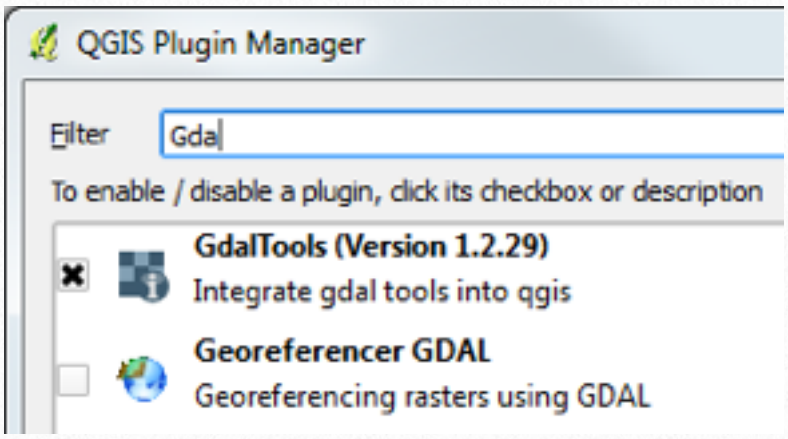
You can see the plugin toolbar remains quite overcrowded, and it is more efficient to only keep in the toolbar the tools you usually use. We suggest to remove the plugins that we will not need often:

- `Plugins > Manage Plugins...`,
- Uncheck the unnecessary plugins (see the list below with the plugins I generally keep): `Add Delimited Text Layer`, `CopyrightLabel`, `GdalTools`, `NorthArrow`, `Plugin Installer`, `ScaleBar`, `fTools`.

The disabled plugins will remain installed in QGIS, where they can be activated when needed (do not hesitate to use the `Filter` func-

tion to find more easily the plugin you are looking for).

The enabled plugins will be available through the Plugins menu or specific menu some other through the Plugin toolbar.



- Take some time to **discover the various functions** available through the interface.

We just saw how to enable or disable the plugins installed by default in QGIS, lets see now how we can add more functionalities:

- Select **Plugins > Fetch Python Plugins...**
- Have a quick look at the official plugins available and installed (8 at the time of writing this document).
- In case you want to add an other repository than the official one, select the **Repositories** tab, click the **Add** button, and **enter the URL of repository in question**.
- Move to the **Options tab**, and **check** the **Show all plugins**, **except those marked as experimental radio button**.
- Move back to the **Plugins tab** and notice you have now more than 75 plugins available.
- Have a quick look at the plugins available and try to **install the XY Tools** plugin using the **Filter**.
- **Close** the window.

Each plugin displays differently in QGIS interface, then sometime you will have to search for them in the documentation associate with the plugin!

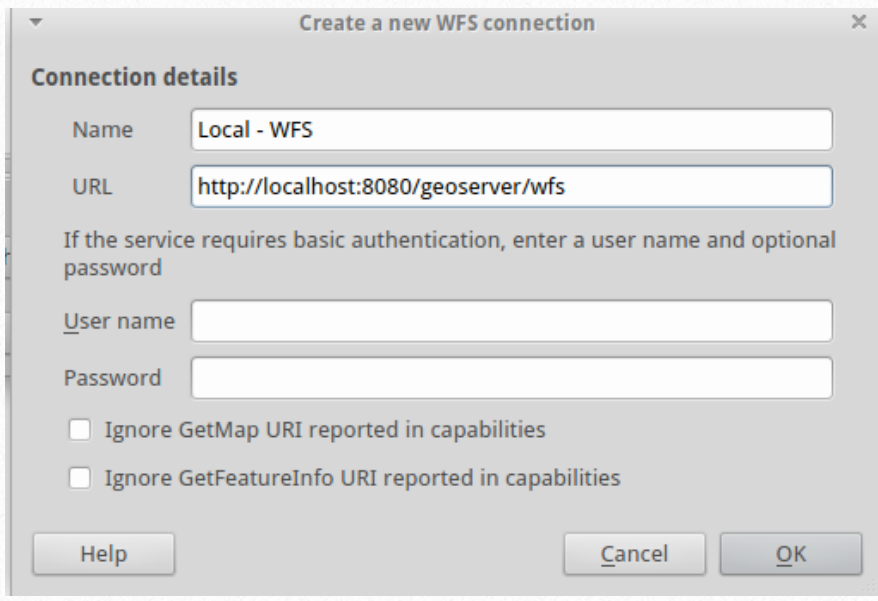
Adding WFS data

In this Section you will add WFS data in QGIS. This data was published on your local server and will be edited later in this exercise.

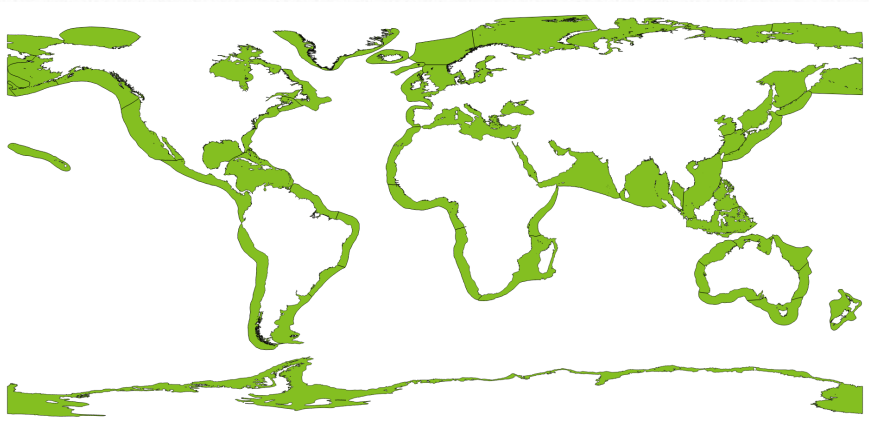
- Click **Add WFS Layer**:

- Click **New** and give a **name** to the connection, for example **Local - WFS**.
- **Paste** the URL of the WFS endpoint:

```
http://localhost:8080/geoserver/wfs
```

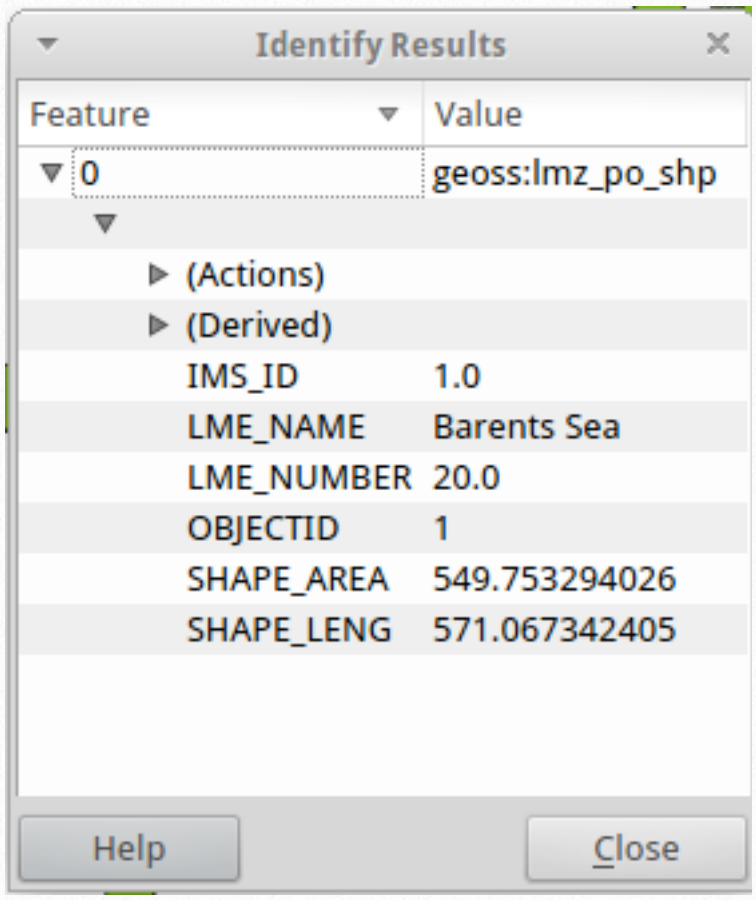


- Click **OK**.
- Click **Connect** (It may take a moment for the data layers list to load).
- Click **Title** to sort the layers by title.
- Select **lmz_po_shp**. This layer was published earlier in GeoServer.
- Click **Apply**. This will add the layer to the map document.
- **Close** the dialog box.
- **Navigate through your map**: Zoom IN/OUT, Pan, etc...



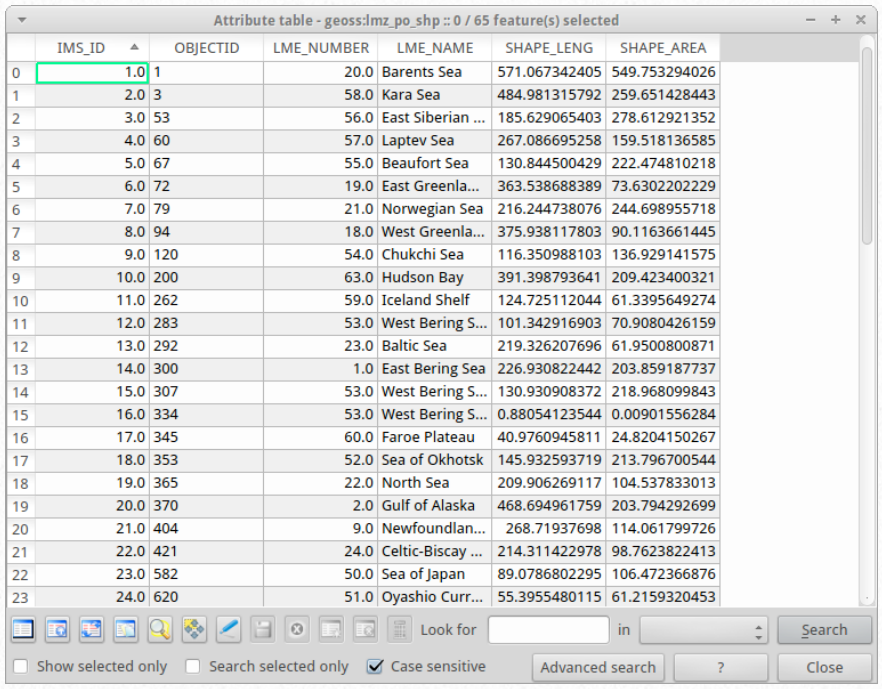
- Click the **Identify features** button

- **Click** on **a marine area** and have look to the query result.



The opening window shows a list of all attributes of the marine area that you identified.

You can also access the attribute table by right clicking on the layer name and selecting `Open attribute table`:



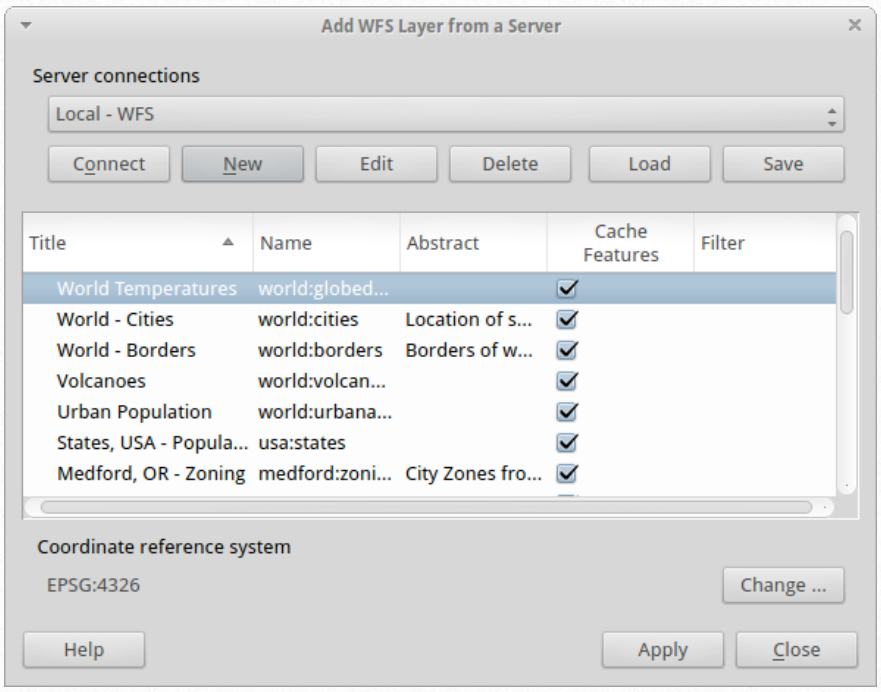
With the WFS you have access to the full vector data set (geometry encoded in GML & attrib-

utes). You can for example save the data to your computer (save as). Then if the transactional feature of a WFS service is enabled, users can edit online the data using the `Edit` toolbar. In what follows you will edit geometry and attributes of another layer stored on your local server.

- You can now **close** the **Identify Results** window.
- You can **turn off** the visibility of the **cyclone** intensity **layer**.

Edit WFS data (geometry)

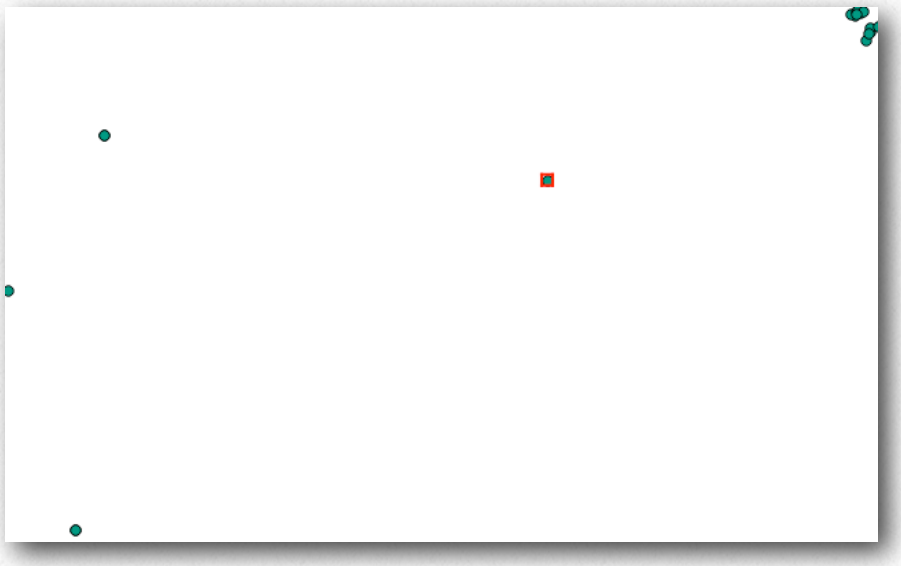
- From QGIS click the **Add WFS Layer** button.
- Select **Local - WFS** and click **Connect** then **OK**.
- Add the **World Temperatures** layer.
- Click **Apply**.



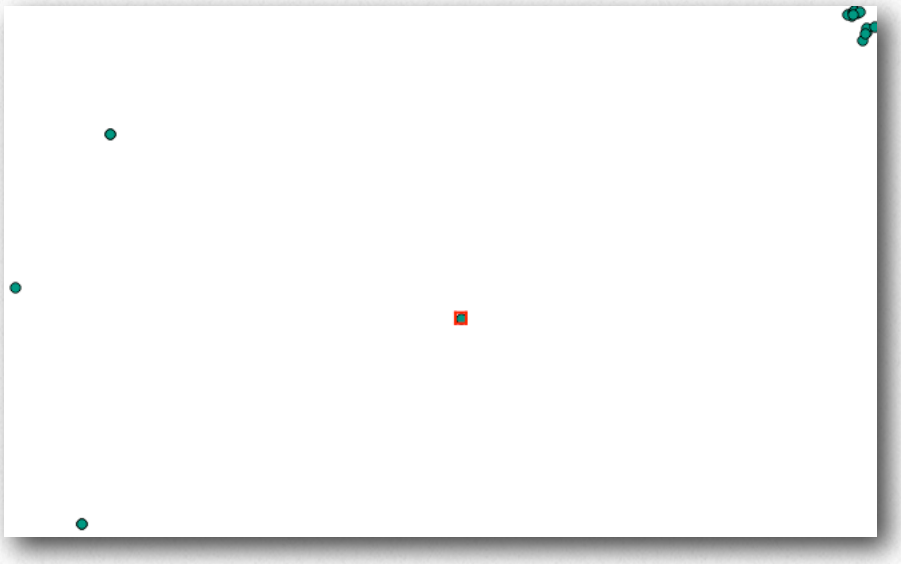
- **Right-click** the **world:globedata_temp** layer and select **Properties**. As you can see below this layer is stored on your local server, has the WFS capability and is in WGS 84 coordinate system. It is possible from QGIS to add features, delete features, change attribute values and change geometries extents:

- Click **OK** to close the Layer Properties window.
- **Right-click `world:globedata_temp`** again and select **Toggle Editing**.
- Select the **Node Tool** (the third one in the screenshot below):

- **Click a point** feature in the map. This point will become as a red square:



- **Move this point** to the South, as shown below:

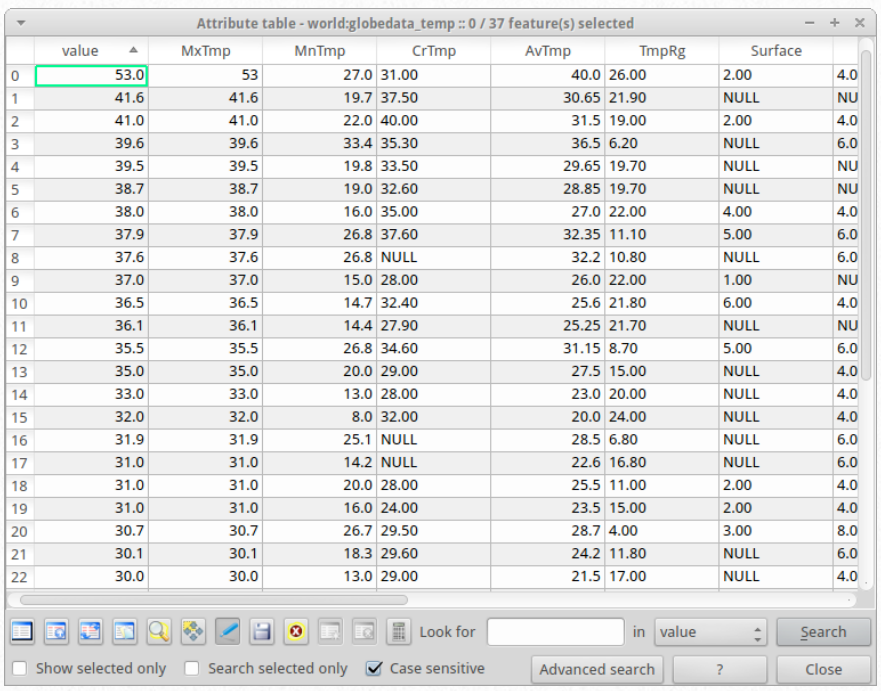


- From the **Layer** menu select **Save Edits**.

Your edits have been saved on your local server.

Edit WFS data (attributes)

- Open the **attribute table of** the **world:globedata_temp** layer.



	value	MxTmp	MnTmp	CrTmp	AvTmp	TmpRg	Surface	
0	53.0	53	27.0	31.00	40.0	26.00	2.00	4.0
1	41.6	41.6	19.7	37.50	30.65	21.90	NULL	NU
2	41.0	41.0	22.0	40.00	31.5	19.00	2.00	4.0
3	39.6	39.6	33.4	35.30	36.5	6.20	NULL	6.0
4	39.5	39.5	19.8	33.50	29.65	19.70	NULL	NU
5	38.7	38.7	19.0	32.60	28.85	19.70	NULL	NU
6	38.0	38.0	16.0	35.00	27.0	22.00	4.00	4.0
7	37.9	37.9	26.8	37.60	32.35	11.10	5.00	6.0
8	37.6	37.6	26.8	NULL	32.2	10.80	NULL	6.0
9	37.0	37.0	15.0	28.00	26.0	22.00	1.00	NU
10	36.5	36.5	14.7	32.40	25.6	21.80	6.00	4.0
11	36.1	36.1	14.4	27.90	25.25	21.70	NULL	NU
12	35.5	35.5	26.8	34.60	31.15	8.70	5.00	6.0
13	35.0	35.0	20.0	29.00	27.5	15.00	NULL	4.0
14	33.0	33.0	13.0	28.00	23.0	20.00	NULL	4.0
15	32.0	32.0	8.0	32.00	20.0	24.00	NULL	4.0
16	31.9	31.9	25.1	NULL	28.5	6.80	NULL	6.0
17	31.0	31.0	14.2	NULL	22.6	16.80	NULL	6.0
18	31.0	31.0	20.0	28.00	25.5	11.00	2.00	4.0
19	31.0	31.0	16.0	24.00	23.5	15.00	2.00	4.0
20	30.7	30.7	26.7	29.50	28.7	4.00	3.00	8.0
21	30.1	30.1	18.3	29.60	24.2	11.80	NULL	6.0
22	30.0	30.0	13.0	29.00	21.5	17.00	NULL	4.0

- **Double-click** the value **27** (first row / column `MnTmp`).

- **Replace** the value **27** with **27.5**

value	MxTmp	MnTmp	CrTm
53	53	27.5	31.00
41.6	41.6	19.7	37.50

- From the **Layer** menu select **Save Edits**.

- From the **Layer** menu select **Toggle Editing**. This will close your editing session.

Congratulations! You have successfully edited both the geometry and the attribute of points stored on your local server. These updates will be visible to anyone who has access (e.g. through WMS) to your server.

- You can now **close QGIS without saving** the project.

Note: In the future QGIS 2.0 release (autumn 2013), there will be a WCS plugin allowing users to download raster data published using OGC WCS standard.

8

How to analyze data?

Keywords: environmental data; geoprocessing; local server; PyWPS; remote server; Web Processing Service (WPS)

What you will learn:

- How to consume a web processing service hosted on your local server.
- How to consume a WPS hosted on a remote server.
- How to use the QGIS WPS client.
- How to visualize the results of a geoprocessing.

Objectives of the chapter

The main objective of this chapter is to learn how to analyze geospatial data through web processing services.

Chapter 5 is a pre-requisite to Chapter 8. It is therefore highly recommended to follow Chapter 5 before going through Chapter 8.

Chapter 8 builds on two practical exercises.

1. In the first exercise, you will consume a geo-processing service that has been published on your local WPS server in Chapter 5.

If you did not achieve Exercise 5, here is a brief reminder of the scenario and the workflow.

A partner working in the field of environmental risk has made measurements of pollution (e.g. by heavy metal) in a specific area. This partner needs an overview of the pollution in the area of study in order to report to local authorities. He/she asked you to register chained geoprocessing operations in a script to allow for automation and reusability. This script has been written in Python language. It generates different outputs: (1) polygonal buffers around the input contamination data, (2) a convex hull representing the whole contaminated area, and (3) a contamination map based on interpolation from the original contamination data. The script will be executed regularly (e.g. quarterly) with updated data and under various conditions. Therefore, the script declares several parameters, in particular the input point layer and the buffer distance. The script has been published as a web processing service on your local server and is ready to be consumed (executed) from a desktop client.

In the first exercise of Chapter 8, you will open the Python script to view input and output parameters. You will then consume the script as a web processing service in your QGIS WPS client through a graphical user window. The final stage is to view geoprocessing results in QGIS.

2. In the second exercise, you will navigate through existing remote WPS servers and consume a web processing service from your local QGIS WPS client. More specifically, you will generate the centroids of a set of polygons. Pros and cons of local and remote services will be briefly discussed.

Note: all data provided for the two exercises of Chapter 8 are fictive and do not reflect the reality in the field. In particular, the study areas were selected arbitrarily and are not meant to be actually polluted by heavy metal or any other contaminant.

Exercise: Analyzing data

1. Exercise 1: Consume a WPS hosted on your local server

Quantum GIS, also known as QGIS, is a cross-platform free and open source desktop application that provides data viewing, editing and analysis capabilities. QGIS runs on Linux, Mac OS X, Microsoft Windows and Android. QGIS supports vector, raster and database formats, such as shapefiles, PostgreSQL/PostGIS, GRASS and GeoTiff. QGIS supports a number of plugins, in particular a WPS client for consuming web processing services.

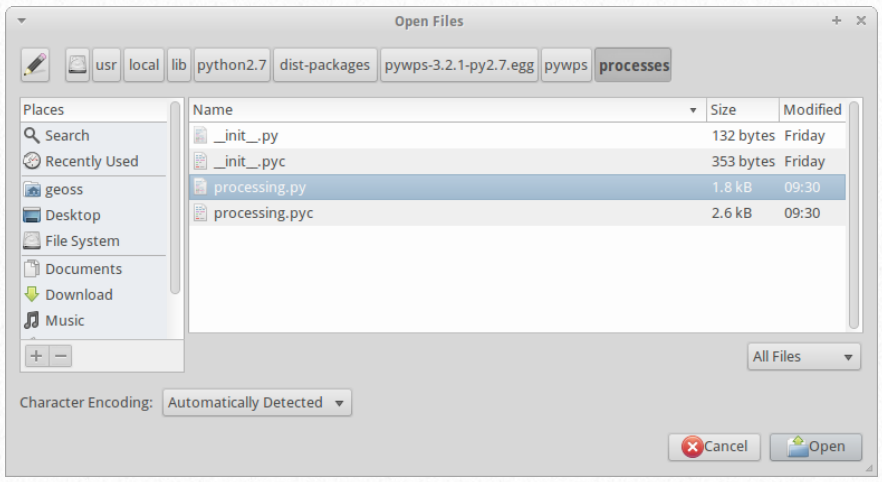
The aim of the first exercise of Chapter 8 is to execute an existing web processing service from the QGIS WPS client. This service has been published from a Python script in Chapter 5.

The first step of the exercise consists of opening the Python script in a text editor to view how input and output parameters have been declared. In a second step you will open QGIS and load vector data. In a third step you will connect to your local WPS server. In a fourth stage you will run the script as a web processing service. You will also check out that the information displayed in the geoprocessing window is in accordance with the Python script (input/output parameters). Finally, you will visualize the geoprocessing results in QGIS and verify that they are consistent with the input data, in particular the coordinate system, the extent, and the contamination values.

1.1. Open an existing script

In this section you will open `processing.py`, a Python script that was published in Chapter 5. It is important to understand how input and output parameters have been declared.

- Click on **Applications Menu > Accessories > gedit**.
- From the **File** menu, select **Open...**.
- Browse to **/usr/local/lib/python2.7/dist-packages/pywps-3.2.1-py2.7.egg/pywps/processes**
- Select **processing.py**



If you did not perform Exercise 5, you cannot select *processing.py* because this file does not exist. To be able to continue with Exercise 8 **you must follow the instructions below**.

- Open the **File Manager**.
- From the File Manager navigate to **Home > Geoss > Desktop > data_exercises**, and **rename** *Exercise5.py* to *processing.py*

Exercise5.py has been prepared for you. This file contains the source code that will be published as a WPS.

- Open the **Terminal Emulator**.
- From the Terminal Emulator enter: **cd /usr/local/lib/python2.7/dist-packages/pywps-3.2.1-py2.7.egg/pywps/processes**
- From the Terminal Emulator enter: **sudo cp /home/geoss/Desktop/data_exercises/processing.py .**
- If prompted enter the password: **geoss**
- From the Terminal Emulator enter: **sudo chmod 777 processing.py**

You are now ready to continue with Exercice 5.

Input and output parameters of the script are declared in four lines of code in the upper part of the script:

```
self.dataIn =
self.addComplexInput(identifier="data",title="Input
data",formats = [{'mimeType':'text/xml'}])

self.widthIn =
self.addLiteralInput(identifier="width",title="Width",
type="Float")

self.bufferOut =
self.addComplexOutput(identifier="buffer",title="Out
put buffer file",formats = [{'mimeType':'text/xml'}])

self.outputInterpo=self.addComplexOutput(identifier
="output",title="Output interpolation
surface",formats=[{'mimeType':'image/tiff'}])
```

The first parameter corresponds to the input layer with contamination measures. `ComplexInput` means that this layer contains geospatial objects (either vector or raster). The layer is a GML file encoded in XML, hence the type `text/xml`. The title (`Input data`) as well as the identifier (`data`) will be displayed in the QGIS WPS client processing window when executing the script.

The second instruction is for declaring the buffer distance. This parameter does not contain geometry (vector/raster), therefore the command `LiteralInput` is required. The type is set to `float`, which means that it can store decimal values, e.g. 0.1 DD or 100.5 meters. The identifier is `width` and the title is `Width`. Both the identifier and the title will appear in the QGIS WPS client processing window when executing the script.

The third parameter corresponds to the output GML file with buffers. This data is encoded in XML as are the input points. In our case, the buffer will represent circles around the input points. These circles are not yet created, hence the use of the `addComplexOutput` command and not `addComplexInput`. Output buffers will be saved in the `/tmp` directory:

```
self.cmd("v.out.ogr type=area format=GML input=d-
ata_buff dsn=/tmp/outputGML.gml")
```

The fourth parameter corresponds to the interpolation surface that will be created by the process from input contamination measures. The type is set to `image/tiff` and the output path is `/tmp`.

Note that the four above mentioned parameters are declared in a specific order. This order will

be preserved in the QGIS WPS client processing window. You will check this out later.

The process will also produce a shapefile named `outputHull` in the `/tmp` directory:

self.cmd("v.out.ogr type=area format=ESRI_Shapefile input=outputHull dsn=/tmp olayer=outputHull")

This data has not been declared as a script parameter. Therefore it will not show up in the QGIS WPS client processing window.

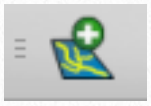
1.2. Add input data to QGIS

In this section you will open QGIS and load the data to process.

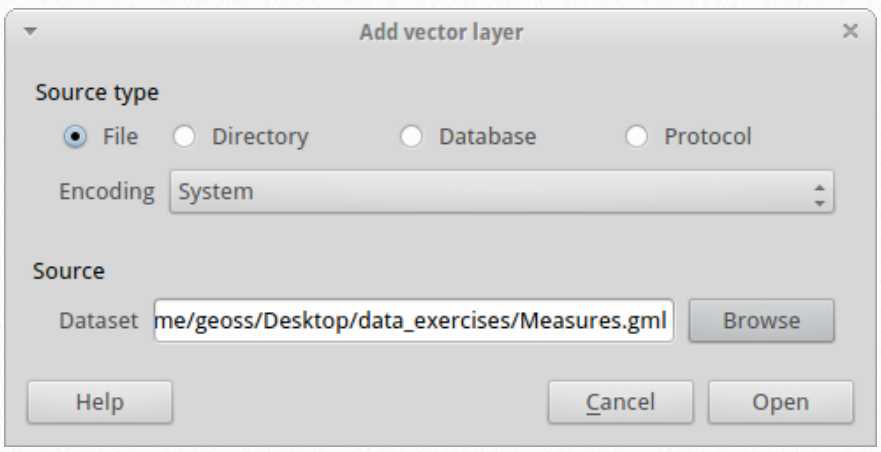
- Select **Applications Menu > Education > Quantum GIS Desktop** to open QGIS.

A new empty project opens.

- Click on the **Add Vector Layer** tool.

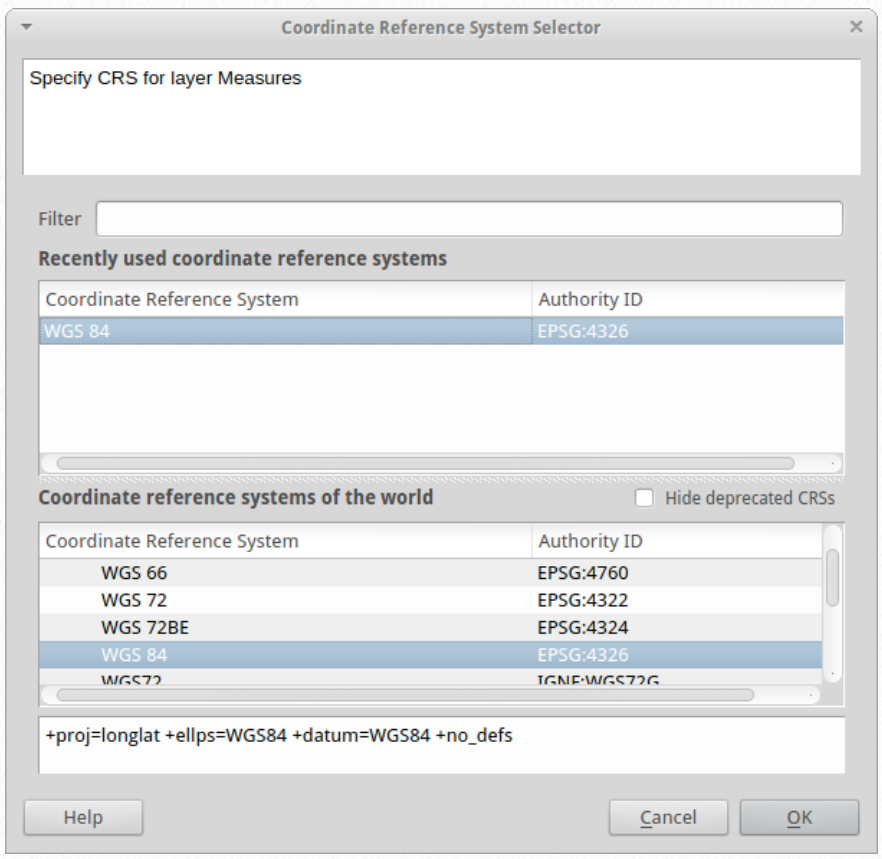


- Browse to **Home > Geoss > Desktop > data_exercises**. Select **Measures.gml**.



- Click **Open**.

- From the opening `Coordinate Reference System Selector` window, select **EPSG: 4326** (WGS 84)

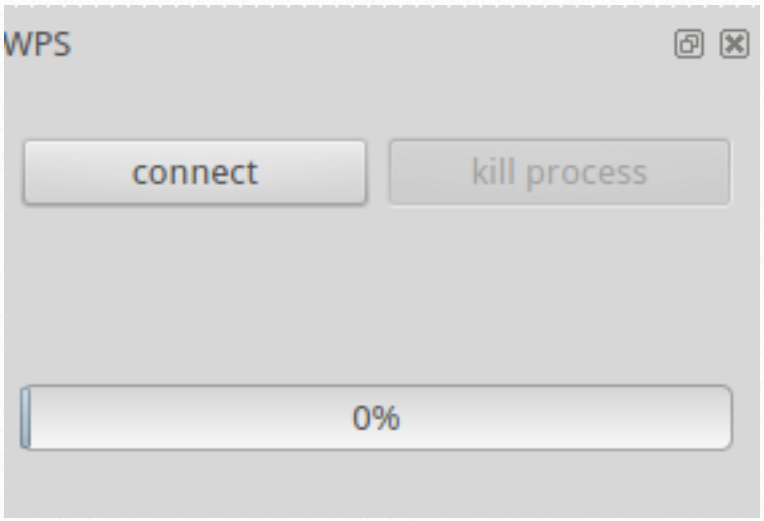


- Click **OK** to validate your choice.

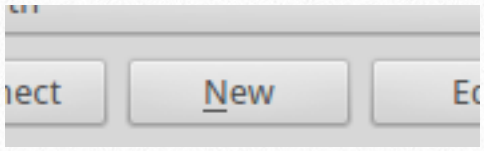
1.3. Connect to your local WPS server

In this section you will connect to your local WPS server through the QGIS WPS client and verify that the information displayed in the processing window is in accordance with the Python source code (input/output parameters).

The QGIS WPS client window should already be visible. It contains a title (*WPS*), a `connect` button, a `kill process` button and a progress bar.



- In the upper part of the QGIS WPS client window click **New** to create a new connection.



- In the opening window, enter the following information:

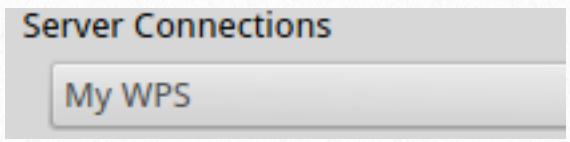
Name: **My WPS**

URL: **http://localhost/cgi-bin/pywps.cgi?service=WPS&request=getcapabilities**

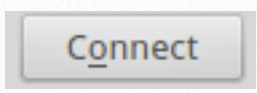
`localhost` stands for your local server.

`The WPS GetCapabilities operation` provides details about a WPS implementation, including service metadata and a description of the available processes.

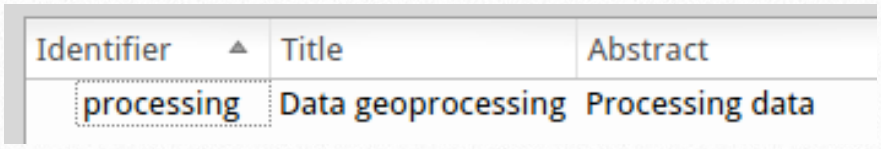
- Click **OK** to validate.
- In the drop-down list select **MyWPS**.



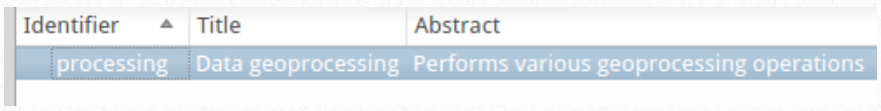
- Click **Connect** to connect your local server.



A list of local processing services with identifier, title and abstract should appear in the lower part of the window. You should at least see the `processing` process, which was published on your local server in Chapter 5.



- From the QGIS WPS client window, select the **processing** row.



- Click **OK**.

The opening window comprises four parameters.

The first parameter is a drop-down list with an identifier (`data`) and a title (`Input data`). The type is set to `text/xml`.

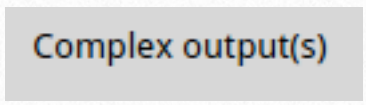
The drop-down list already points to the `Measures` GML layer, because this is the unique layer in your QGIS project. If there were other GML layers in the map, you could select one of them in the list.

The second parameter is a blank field that allows you to enter a value for the buffer distance. The identifier is set to `width` in lower case and the title is set to `Width` with a capital W.

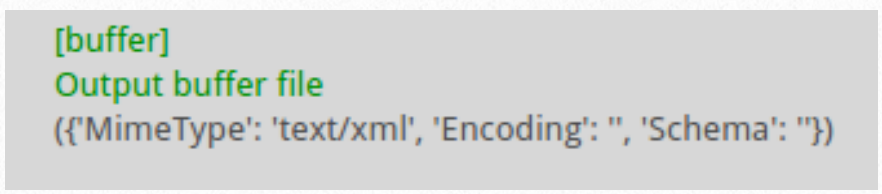
- In the blank field, enter **0.1**

This value is in decimal degrees. 0.1 DD corresponds to about 10 km. This value is arbitrary and does not reflect reality in the field.

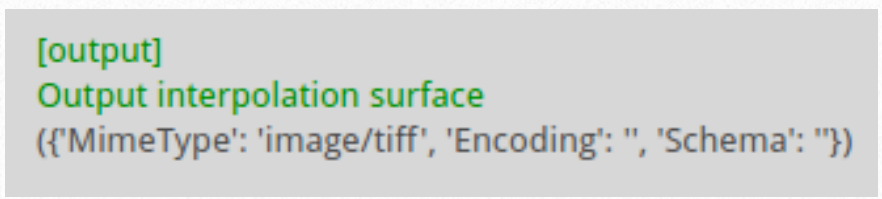
The third and fourth parameters are both complex outputs because they contain geospatial objects (vector / raster).



The third parameter is the output buffer stored in a GML file and encoded in XML.



The fourth parameter is the output interpolation surface stored in TIFF format.

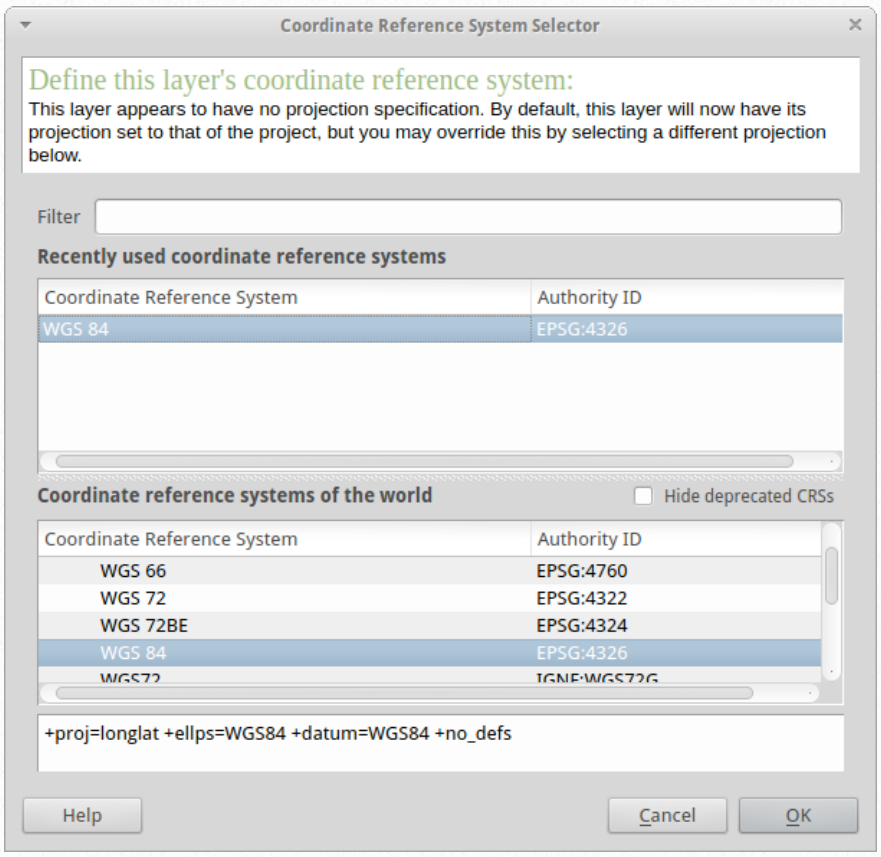


1.4. Run the WPS from your local WPS server

In this section you will run the web processing service from your local server.

- Click **Run** to run the web processing service.

After a few seconds the `Coordinate Reference System Selector` window shows up.



- Select **EPSG:4326** (WGS 84) and click **OK**.

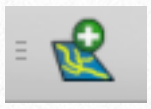
By doing this you ensure that the layers and the map are in the same coordinate system.

A new raster layer is added to the QGIS project. Each cell of the raster contains a contamination value derived from the interpolation algorithm.

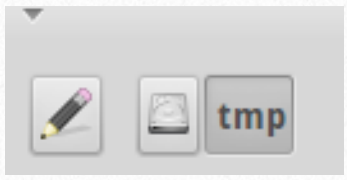
1.5. View the geoprocessing results in QGIS

In addition to the interpolation surface, the geoprocessing has generated a shapefile with the convex hull of the contaminated area, and a GML file with buffers. In what follows you will add these two layers in the map and check that their properties (coordinate system and extent) and their content (contamination values) are consistent with the input points layer. For more readability, you will reorder the layers in QGIS, use labels and transparency, and apply relevant styles.

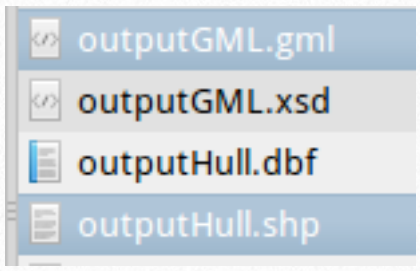
- Click the **Add Vector Layer** tool



- **Browse** to the **/tmp** folder. This folder is located under `File System`.



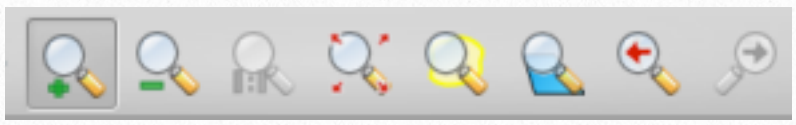
- Select **OutputGML.gml**
- Hold on the **CTRL** button located on your keyboard and select **OutputHull.shp**



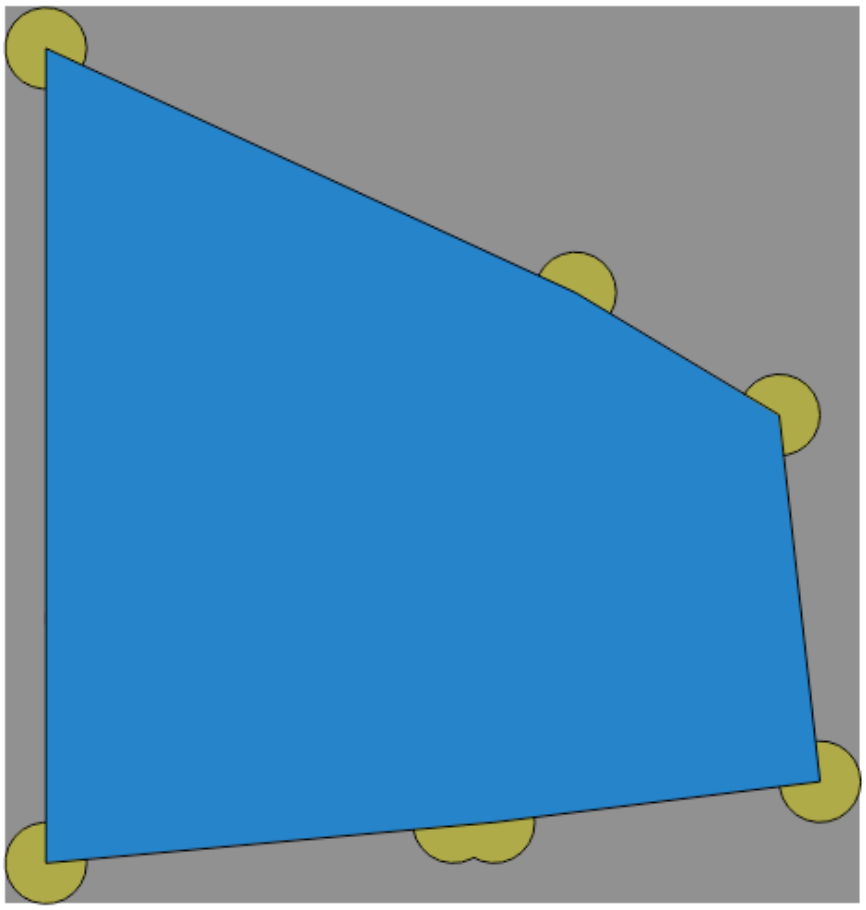
- Click **Open**.
- Click **Open** once again.
- Select the **EPSG:4326** coordinate system.
- Select **EPSG:4326** again.
- Click **OK**.

The convex hull layer and the buffers layer are added to the map.

You can use the *`Map Navigation`* tools if you want to zoom to another extent, e.g. to zoom out.



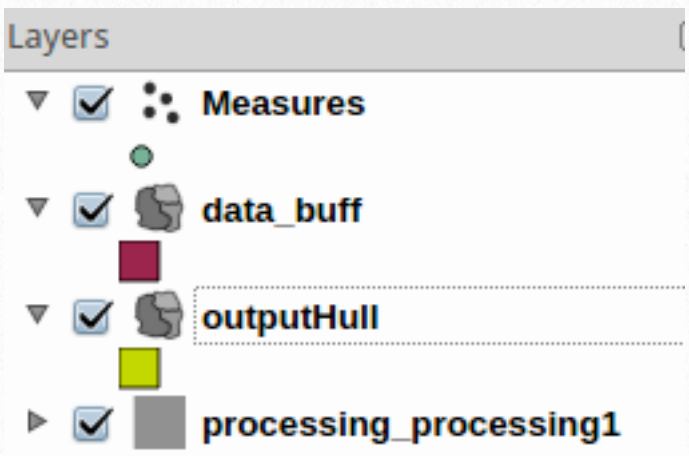
You map should look like:



The four layers need to be reordered in the QGIS table of contents if you want to perform further GIS analysis. For example, the input point layer is not visible because it is placed below the raster layer.

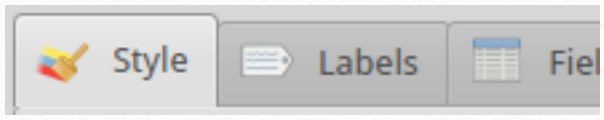
Point layers are usually placed on top. Layers with lines are usually placed below point layers, and polygons below lines. Raster layers, e.g. base maps, are usually placed at the bottom.

- **Drag and drop the layers** to obtain the following configuration:



`outputHull` is a polygon layer, therefore it is represented with a fill symbol. Consequently, the raster layer is hidden. To solve this issue you will apply a transparency to the convex hull layer.

- **Right-click outputHull** and select **Properties**.
- Select the **Style** tab.

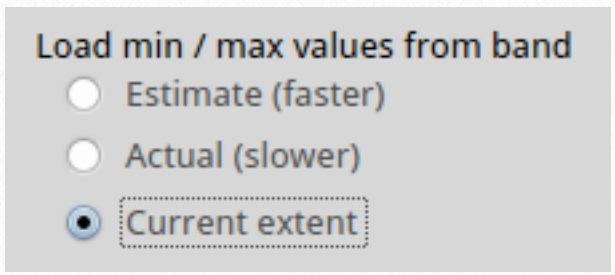


- Move the cursor to the right to set a **transparency of 40%**.

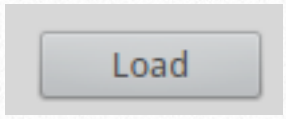
- Click **Apply** then **OK**.

By default, the raster layer is represented with a unique color. Consequently, low contamination values cannot be distinguished from high contamination values. To solve this issue you will apply a color ramp:

- **Right-click the raster layer** and choose **Properties**.
- Select the **Style** tab.
- From the `Load min / max values from band` section, select **Current extent**.



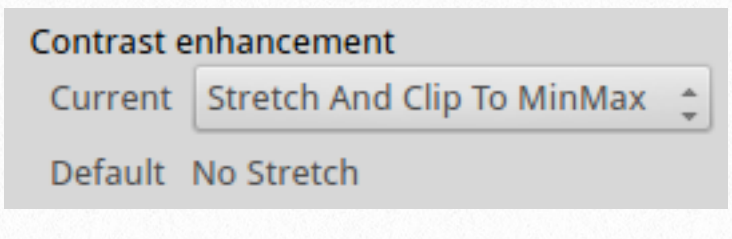
- Click **Load**.



By choosing the `Current extent` option, you ask QGIS to read the statistics of the entire dataset, including the minimum contamination value, the maximum contamination value, the average contamination value and the standard deviation. You can check that contamination values range from 5 to 17, which is consistent with the values stored in the attribute table of the input points layer (See Exercise 5).

QGIS is now ready to apply a color map to the interpolation raster.

- Under `Contrast enhancement` select **Stretch and Clip to MinMax**.



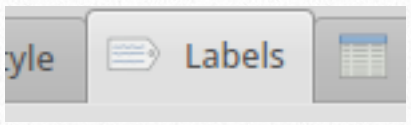
This option allows including all contamination values in the color map (from 5 to 17).

You can also invert the color map if you want low contamination values to be represented with white pixels and high contamination values with dark pixels:

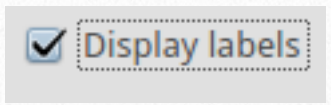
- Click **Apply** then **OK**.

You might want to verify that the input contamination values have been interpolated correctly. To that end you will label the input point layer. This layer contains a field `Measure` storing contamination values (Cf. Exercice 5).

- **Right-click Measures** and select **Properties**.
- Select the **Labels** tab.

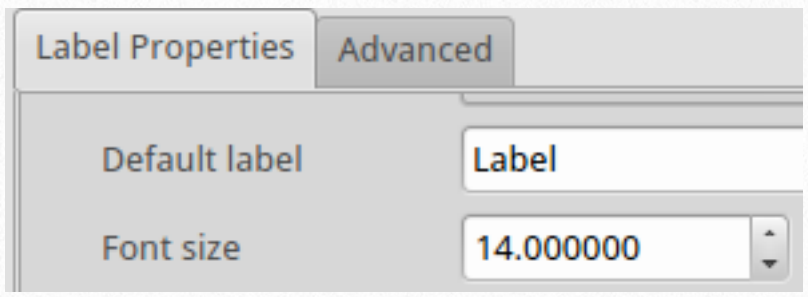


- Check the **Display labels** box.

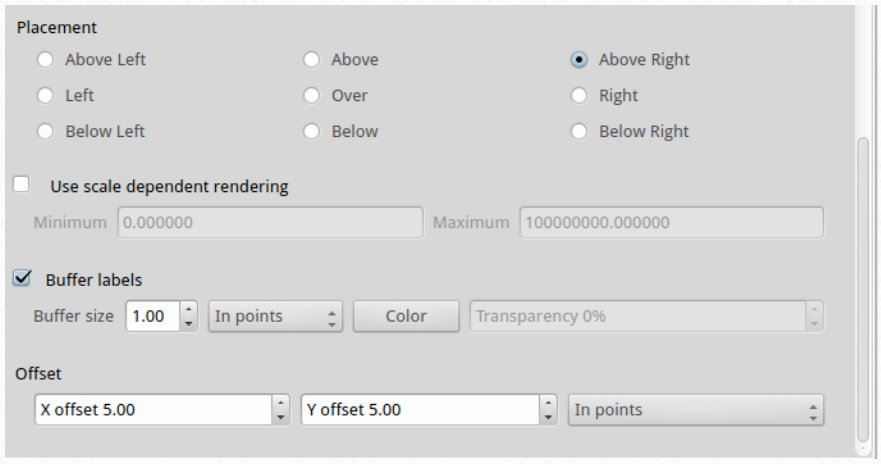


- In the **Field containing label** drop-down list, select **Measure**

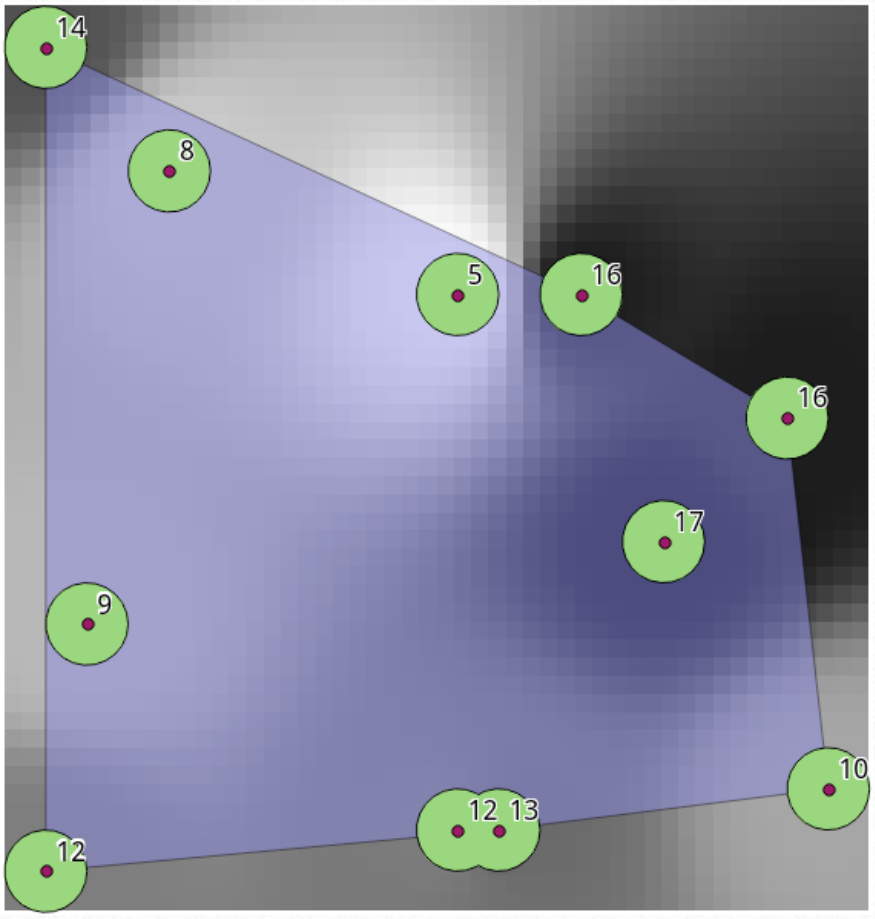
- For **Font Size** enter **14**



- You can also change the font color (for example choose a dark color), place labels **Above Right** of points, create white halos around labels, and apply X and/or Y **offsets** to avoid overlaps between points and labels.



- Click **Apply** then **OK** to validate your choices.



You are now able to visualize the four layers (1 input + 3 outputs) simultaneously. These data are properly georeferenced and have similar extents. The variations of the contamination map also are consistent with input contamination measures.

- You can now **close gedit** if not already done.

In this exercise you have learnt how to analyze data by running a web processing service stored on your local server. The main advantage of hosting a service on your local server is that you keep control over:

- The mode of execution: synchronous / asynchronous;
- The number of simultaneous processes allowed;
- The source code;
- The graphical user interface;
- Security.

This supposes that you have GIS resources and capacity to develop and maintain web processing services.

One step beyond this is to allow remote users to access and execute your processes, e.g. partners that do not have the GIS resources and/or capacity for developing Python scripts and publishing them as WPS.

In Exercise 2 of Chapter 8, you will take the role of a user with low GIS capacity and run processes that have been developed by others. To that end you will use the QGIS WPS client. By default this plugin contains default connections to remote servers and provides you with the opportunity to test and run dozens of processes on vector and raster datasets.

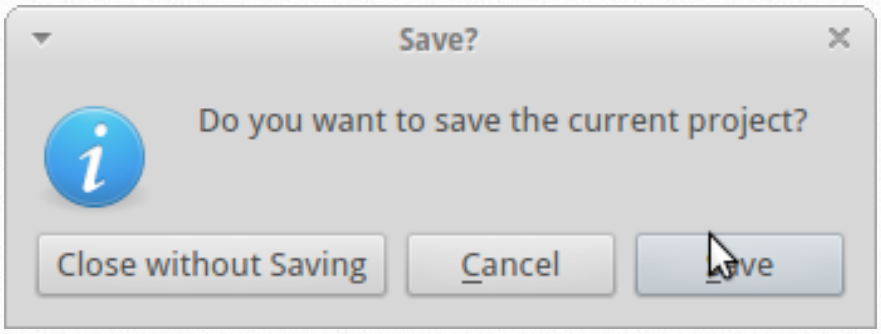
2. Exercise 2: Consume a WPS hosted on a remote server

In this exercise you will navigate through existing remote WPS servers and consume a web processing service from your local QGIS WPS client. More specifically, you will generate the centroids of a set of polygons.

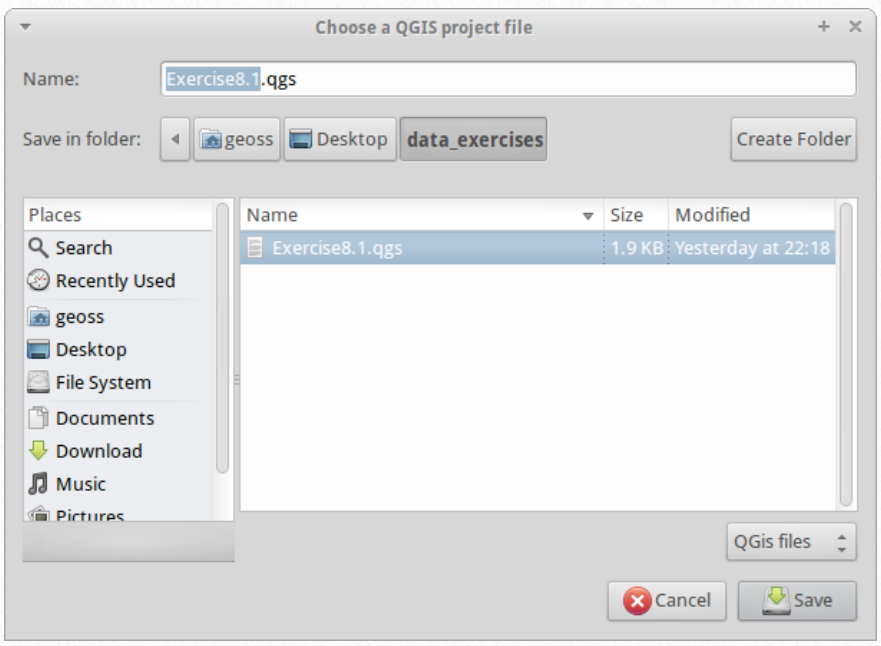
2.1. Navigate through existing remote WPS servers

In a first step you will close the current QGIS project.

- Select **File** > **New project**.
- Choose **Save**.



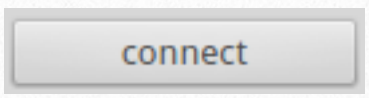
- Click **Open**.
- Browse to the **data_exercises folder** and save the project as **Exercise8.1**



- Click **Save** to validate your choice.

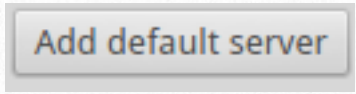
The QGS extension has automatically been added to your QGIS project which is now saved as `Exercise8.1.qgs`

- From the WPS client click **connect**



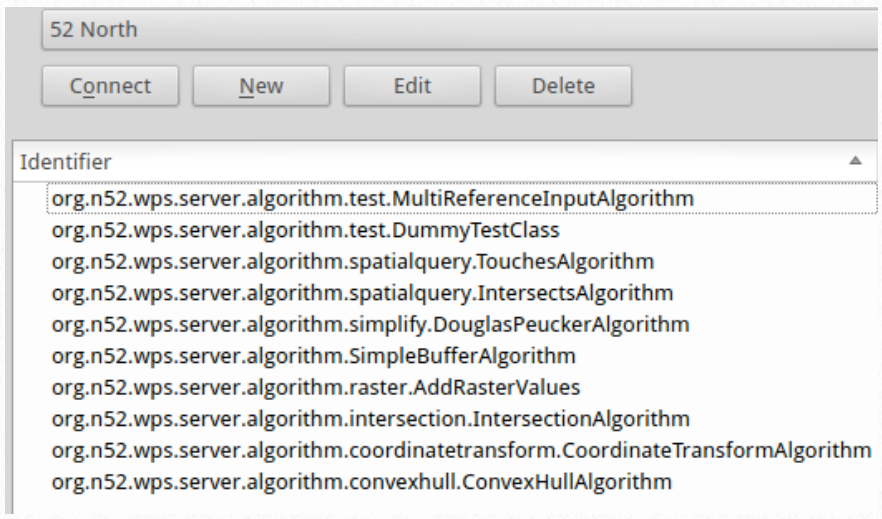
The QGIS WPS client window shows up. It contains a drop-down list. This list should at least contain `MyWPS`, which is the connection to your local server (you have declared this connection in the previous exercise).

The list should also contain several connections to remote WPS server. If it does not, click on **Add default server**.



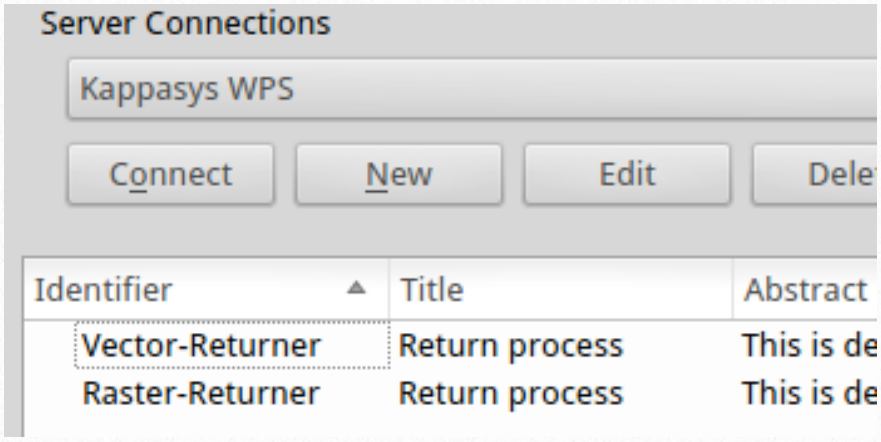
1. You should have a connection to 52 North, which is an open source software initiative. 52°North is an open international network of partners from research, industry and public administration. Its main purpose is to foster innovation in the field of Geoinformatics through a collaborative research and development process. 52°North refers to the degree of latitude which intersects the founding organizations' home - the city of Münster, Germany.

- **Select 52 North** from the drop-down list and click **Connect**. You should see a dozen of web processing services for managing vector geometries:



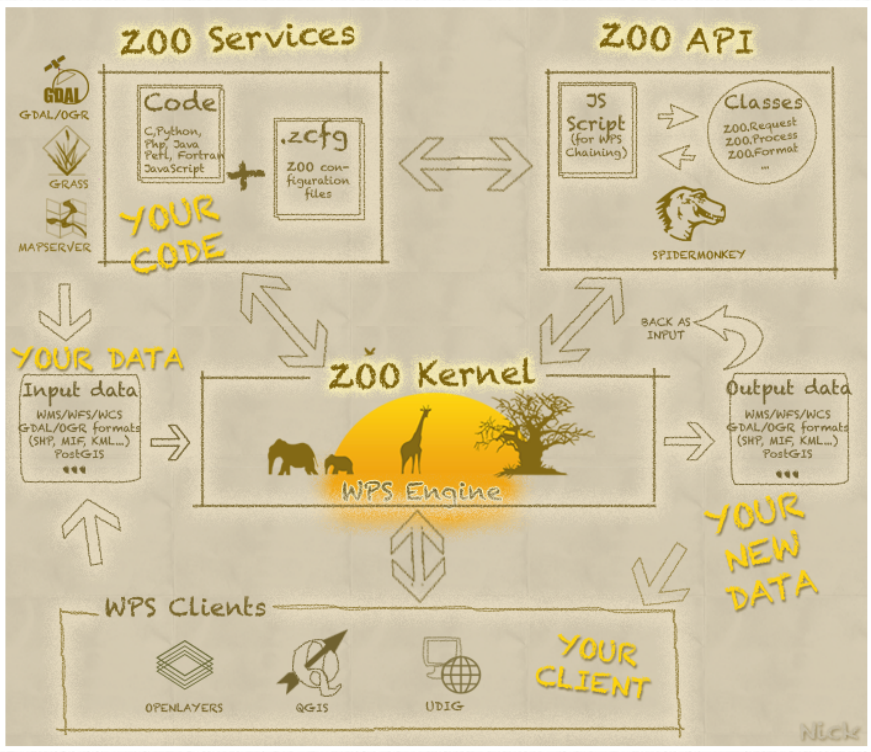
2. You should also see a connection to the Kappasys WPS server. Kappasys is a swiss-german company specialized in the design and construction of distributed spatial data infrastructure based on FOSSGIS.

- **Select Kappasys WPS** from the drop-down list and click **Connect**. Two web processing services are available:

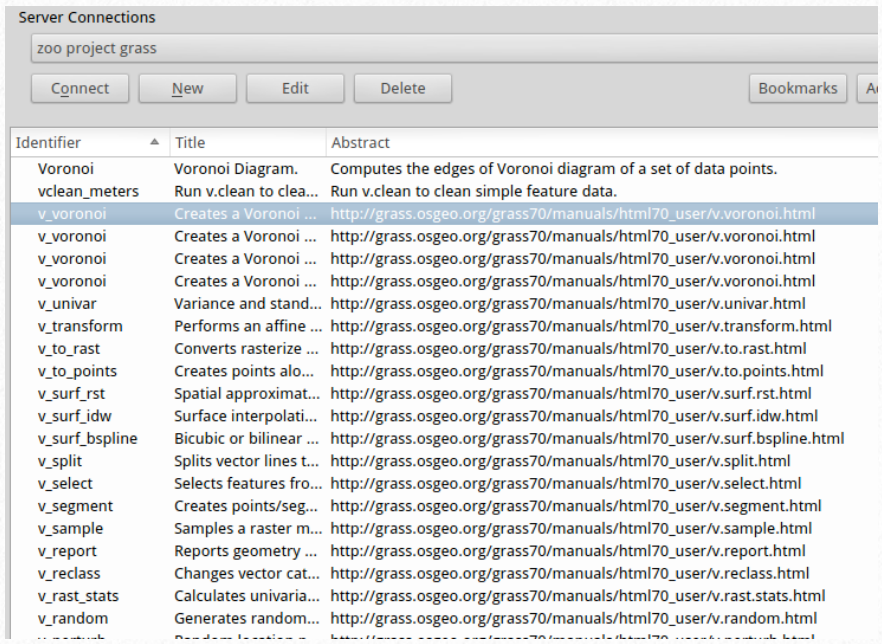


- Take a while to read the information provided in the window: identifier, title and abstract. You can also test the services using vector and raster data stored in `data_exercises`.

3. ZOO is open source project that provides an OGC WPS compliant developer-friendly framework to create and chain WPS Web services developed in different programming languages (Javascript, Grass, C, Python etc.):



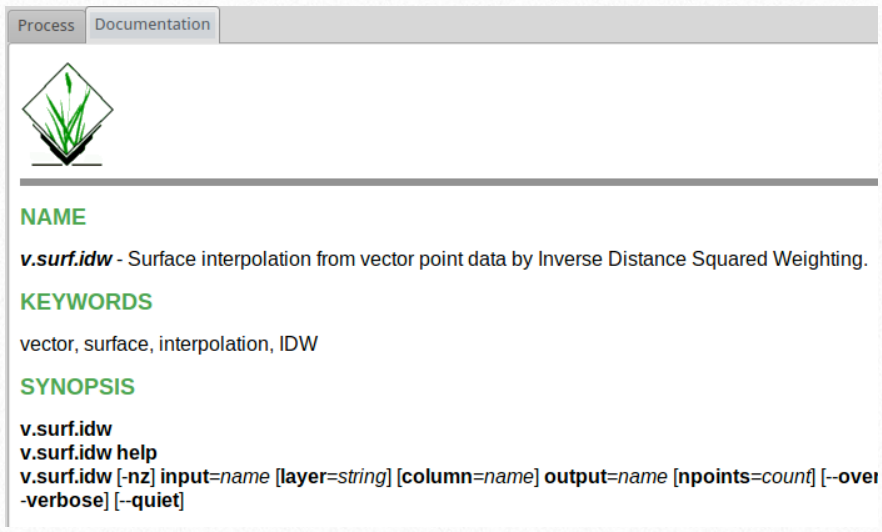
- **Select zoo project grass** from the drop-down list. Several dozens of processes are available.



- **Select** one of these processes, e.g. **v_surf_idw**. Note that you have used the IDW function to generate an interpolation surface during the previous exercise.

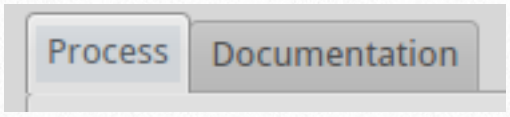
- Click **OK**.

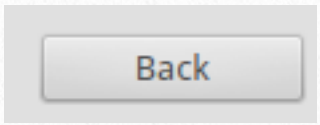
- In the opening window select the **Documentation** tab.



The authors of the ZOO project have documented the `v_surf_idw` process to help non advanced users understand what this process does. More specifically the authors have provided a link to the GRASS online help of the function.

- To close the `v_surf_idw` processing window click the **Process** tab and click the **Back** button.





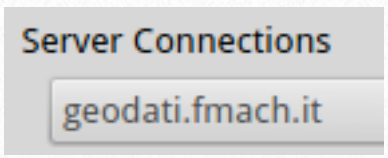
4. geodati.fmach.it is a WPS server developed in the framework of the ZOO project. You will now consume a web processing service hosted on this server.

2.2. Consume a web processing service hosted on a remote WPS server

- Click the **Add Vector Layer** tool

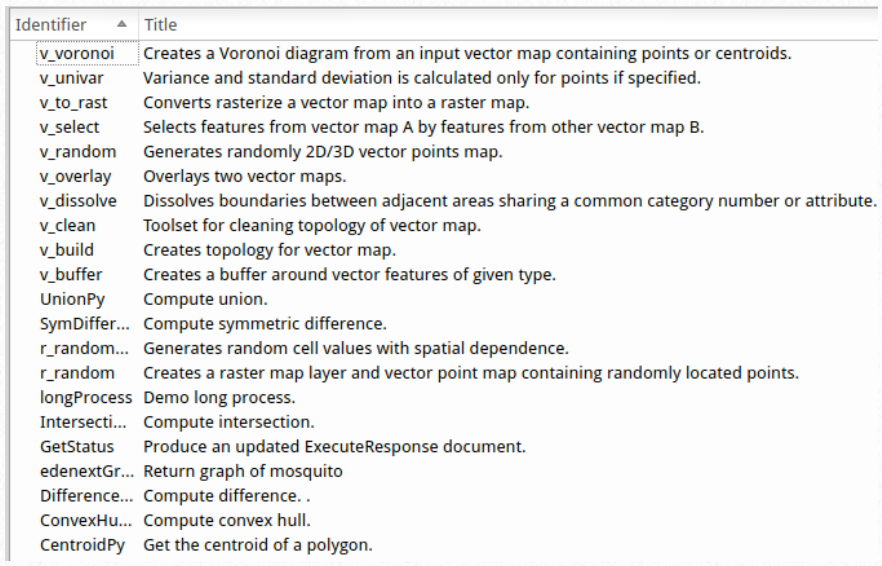
- Browse to the **data_exercises** folder and select **Parcels.gml**

- From the drop-down list select **geodati.fmach.it**



- Click **Connect**

There are about 25 web processing services that correspond to basic operations on vector and raster data such as: buffer, dissolve, select, intersect, boundary, clean, random, random cells and others.



Identifier	Title
v_voronoi	Creates a Voronoi diagram from an input vector map containing points or centroids.
v_univar	Variance and standard deviation is calculated only for points if specified.
v_to_rast	Converts rasterize a vector map into a raster map.
v_select	Selects features from vector map A by features from other vector map B.
v_random	Generates randomly 2D/3D vector points map.
v_overlay	Overlays two vector maps.
v_dissolve	Dissolves boundaries between adjacent areas sharing a common category number or attribute.
v_clean	Toolset for cleaning topology of vector map.
v_build	Creates topology for vector map.
v_buffer	Creates a buffer around vector features of given type.
UnionPy	Compute union.
SymDiffer...	Compute symmetric difference.
r_random...	Generates random cell values with spatial dependence.
r_random	Creates a raster map layer and vector point map containing randomly located points.
longProcess	Demo long process.
Intersecti...	Compute intersection.
GetStatus	Produce an updated ExecuteResponse document.
edenextGr...	Return graph of mosquito
Difference...	Compute difference. .
ConvexHu...	Compute convex hull.
CentroidPy	Get the centroid of a polygon.

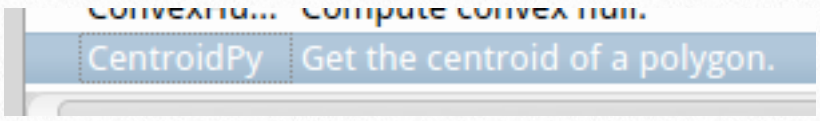
Processes that start with “v” work on vector data. Processes that start with “r” work on raster data.

Take a while to read the title and abstract of each process. Most of these processes have been documented by the authors. At least a link to the online function help has been provided. To consult this documentation, select a process from the window, click `OK` and select the `Documentation tab. To close the` documentation, select the `Process` tab and click the `Back` button.

You will now run the function for creating polygons centroids. The centroid of a polygon is the geometric center of this polygon. The centroid is calculated using a mathematical algorithm. The concept of polygon centroids is widely used to estimate the geometric center or the center of gravity of individual polygons in a collection. In many GIS solutions, polygons centroids are used when plotting maps as the default location for displaying polygon labels.

In this exercise you will generate the centroids of the parcels that have been added to the QGIS map.

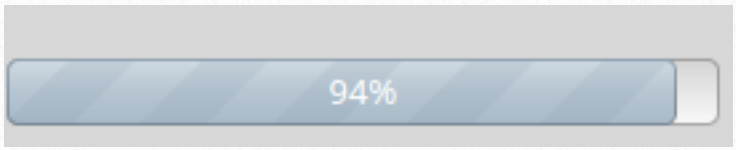
- Select **CentroidPy**



- Click **OK**.

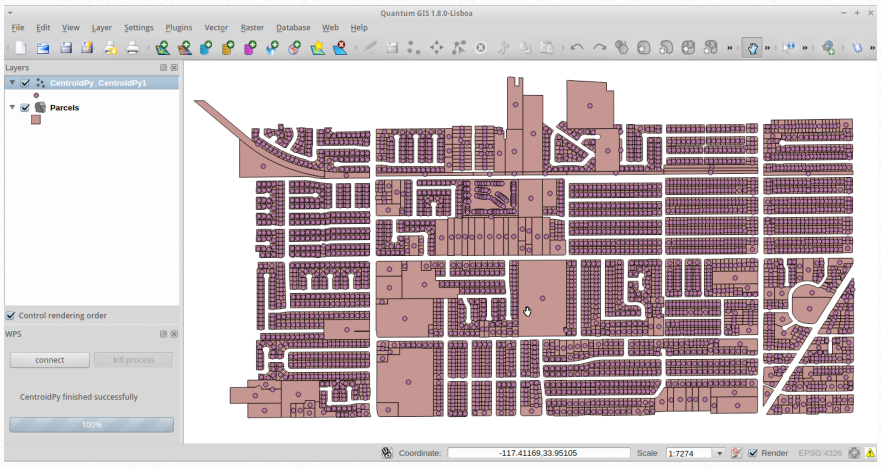
- Click **Run**.

The process should take a few minutes. The progress bar shows the progress of the process.



- When the process is finished, the Coordinate System Reference Selector window shows up. Select **EPSG:4326** (WGS 84) and click **OK twice**.

A new layer is added to the map. A centroid has been created for each parcel. The centroids layer is in WGS 84 as is the case of the parcels layer.



- Take a while to test other web processing services provided by remote servers, e.g by 52 North.

- You can now **close QGIS** and **save** the project as **Exercise8.2**.

In this exercise you have learnt how to analyze data by running a web processing service stored on a remote server. The main advantages of using remote services are the following:

- You do not need to develop and maintain an infrastructure;

- The source code is maintained by expert developers, therefore you can use services without advanced GIS skills or programming competences.

Congratulations! You have successfully performed the two exercises of Chapter 8. In next Chapter, you will learnt how to share web services.

9

How to share services?

Keywords: access; broker; discovery; GEOSS; OGC; web services

What you will learn:

- What is the GEOSS Common Infrastructure
- How to share individual components and services in GEOSS
- How to share a big capacity, such as a Spatial Data Infrastructure, in GEOSS
- How to set up a Discovery and Access Broker for your community of practice

The GEOSS Common Infrastructure

The Global Earth Observation System of Systems aims at enhancing the relevance of Earth observations for the global issues and to offer a public access to comprehensive, near-real time data, information and analyses on the environment. It will provide decision-support tools to a wide variety of users, in a broad range of so-called Societal Benefits Areas:

- Reducing loss of life and property from natural and human-induced disasters,
- Understanding environmental factors affecting human health and well-being,
- Improving the management of energy resources,
- Understanding, assessing, predicting, mitigating, and adapting to climate variability and change,
- Improving water resource management through better understanding of the water cycle,
- Improving weather information, forecasting and warning,
- Improving the management and protection of terrestrial, coastal and marine ecosystems,
- Supporting sustainable agriculture and combating desertification, and
- Understanding, monitoring and conserving biodiversity.

GEOSS is not designed as a centralized facility, but rather builds upon already existing systems, SDIs and Earth Observations assets, encouraging interoperability and open access to their data and services.

The GEOSS 10-Year Implementation Plan for the period 2005 to 2015 specifies principles for data and information sharing and dissemination.

GEOSS members must fully endorse the following data sharing principles:

1. There will be full and open exchange of data, metadata, and products shared within GEOSS, recognizing relevant international instruments and national policies and legislation.

2. All shared data, metadata, and products will be made available with minimum time delay and at minimum cost.

3. All shared data, metadata, and products being free of charge or no more than cost of reproduction will be encouraged for research and education.

Moreover, the data providers must accept and implement “a set of interoperability arrangements, including technical specifications for col-

lecting, processing, storing, and dissemination shared data, metadata and products."

To minimize the impact on participating systems, GEOSS interoperability arrangements only define how system components interface with each other. GEOSS is based on non-proprietary standards, with preference to formal international standards, existing technologies, and internet-based services.

This 'system of systems' proactively links to existing observing systems around the world and supports the interconnection of new capacities where gaps currently exist.

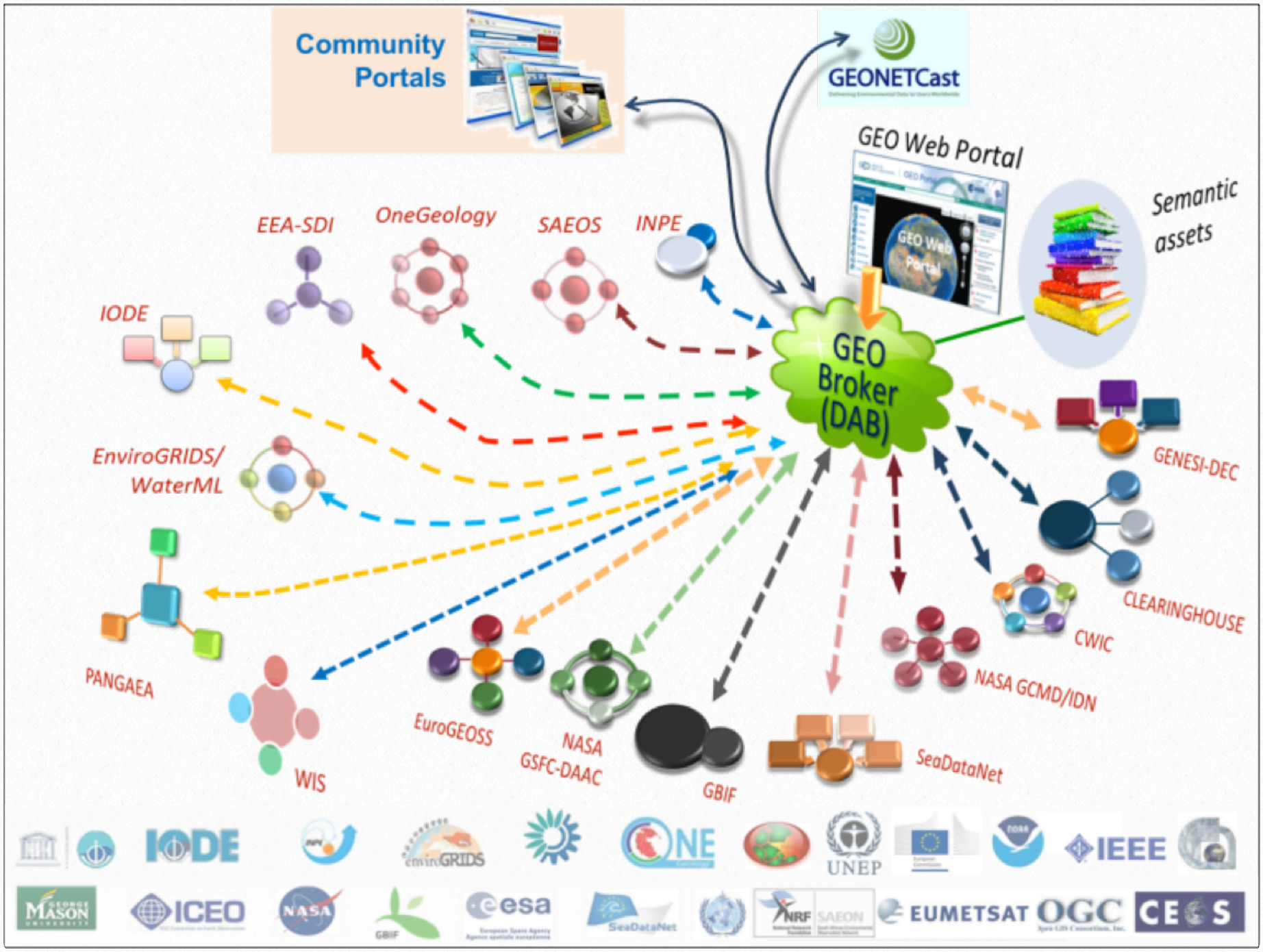
To effectively act as a gateway between the producers and the end users of environmental data, GEOSS provide to all its members a framework of value-added services that enhance interoperability among the growing number of heterogeneous Earth observing systems serving different disciplines and applications, at the regional and global level.

Such framework is termed the GEOSS Common Infrastructure (GCI). It allows the members to publish, discover, evaluate, access, and use the data, information, tools and services made available through the Global Earth Observation System of Systems.

The GCI is not another infrastructure for data storage. In fact, it is a brokering infrastructure. This means that it connects existing Earth observing systems and data infrastructures without requiring neither the data providers nor the users to make any change to their technologies or standards.

Brokering components realize all the necessary mediation, adaptation, indexing, distribution, semantic mapping, and even quality checking required to address the growing complexity of the GEOSS cyber-infrastructure

At present, GEOSS includes such diverse capacities as the Global Biodiversity Information Facility (GBIF), the WMO Information System (WIS), OneGeology, and many more (see the next figure).



A simplified schema of the GCI and some of the GEOSS data providers

For the purpose of this Chapter, the GCI components of main interest for sharing your components and services and make them available through GEOSS are:

- The GEO Web Portal, by which the user may discover, evaluate, and access the information and services made available by GEOSS (please note that the GCI also supports additional Community Portals, possibly specialized for a given application field).

- The Clearinghouse, a catalogue of references to components and services, along with their related metadata.

- The GEO Broker, or Discovery & Access Broker (GEO DAB), which inter-connects the various GEOSS components and supports index-

Screenshot of the GEO Web Portal
www.geoportal.org/

ing, discovery, mediation, adaptation, distribution, as well as other value-added functionalities.

The following sections elaborate on three scenarios:

- Sharing an individual component or service.
- Sharing a big capacity (federation, cyber-infrastructure, SDI, big data system).
- Sharing multiple resources of a community of practice.

Sharing an individual component

If you want to share an individual component, such as a database, a catalogue, or a specific tool, you can register it to the GEOSS Components and Services Registry (CSR).

The CSR is similar to a library catalogue providing essential details about the name, contents, and management of the components and services contributed by a GEO Member or Participating organization.

The records stored in the CSR are accessed by the Clearinghouse, which is in turn accessed by the Broker and ultimately by the user (possibly through the GEO Portal), to identify the GEOSS resources of interest.

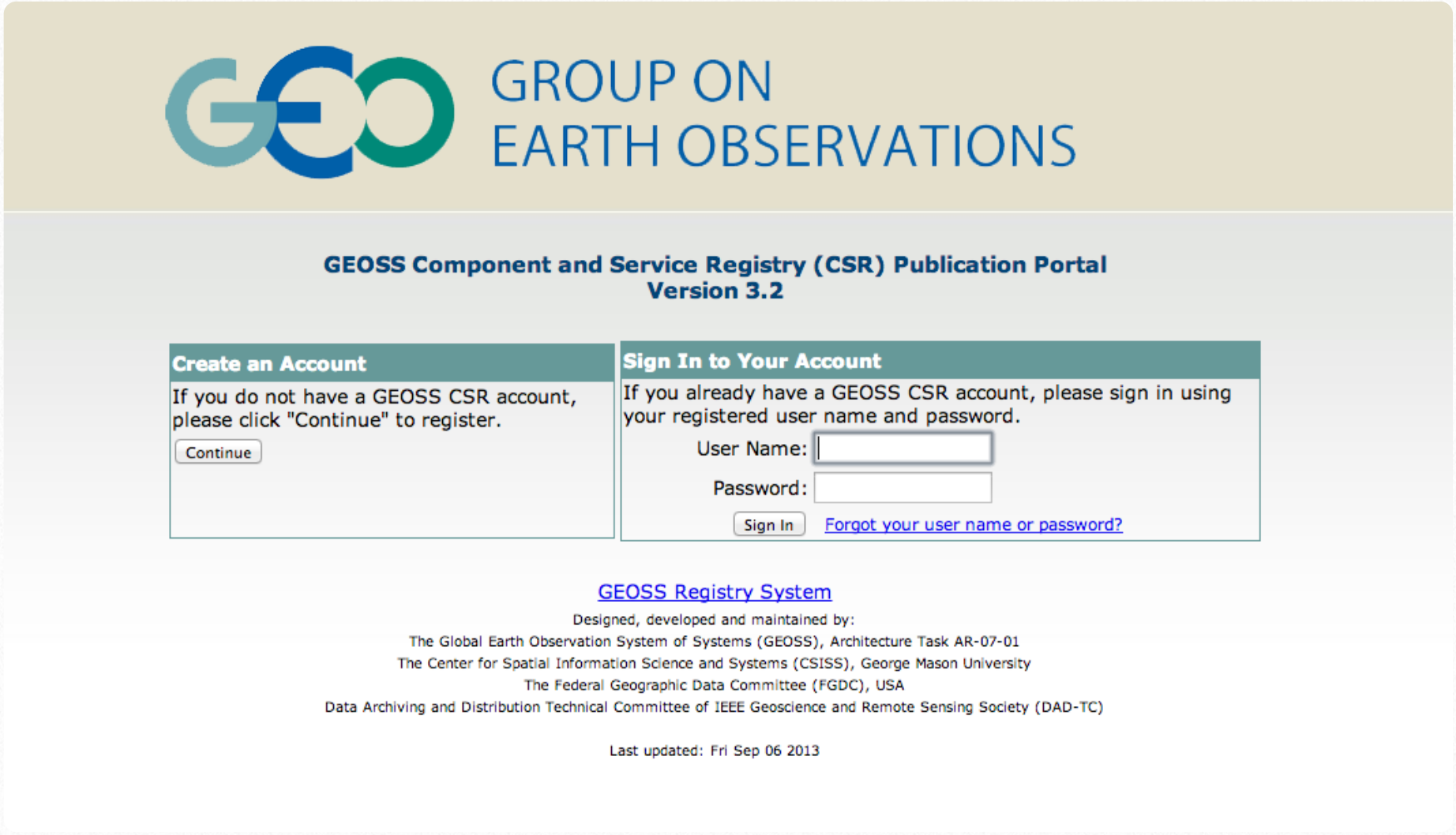
The CSR provides a formal listing and description of various Earth observation systems, data sets, models and other services and tools that are part of the Global Earth Observation System of Systems.

Components and services supported at present are:

- Datasets,
- Monitoring and Observation Systems,
- Computational Models,
- Initiatives,
- Websites and Documents,
- Analysis and Visualization,
- Alerts, RSS, and Information FeedsCatalogues, Inventories and Metadata Collections,
- Software and Applications.

These various components are interlinked using standards and protocols that allow data and information from different sources to be integrated.

The components and services listed on the CSR can be searched and explored by decision-makers, managers and other GEOSS users via the GEO Portal.

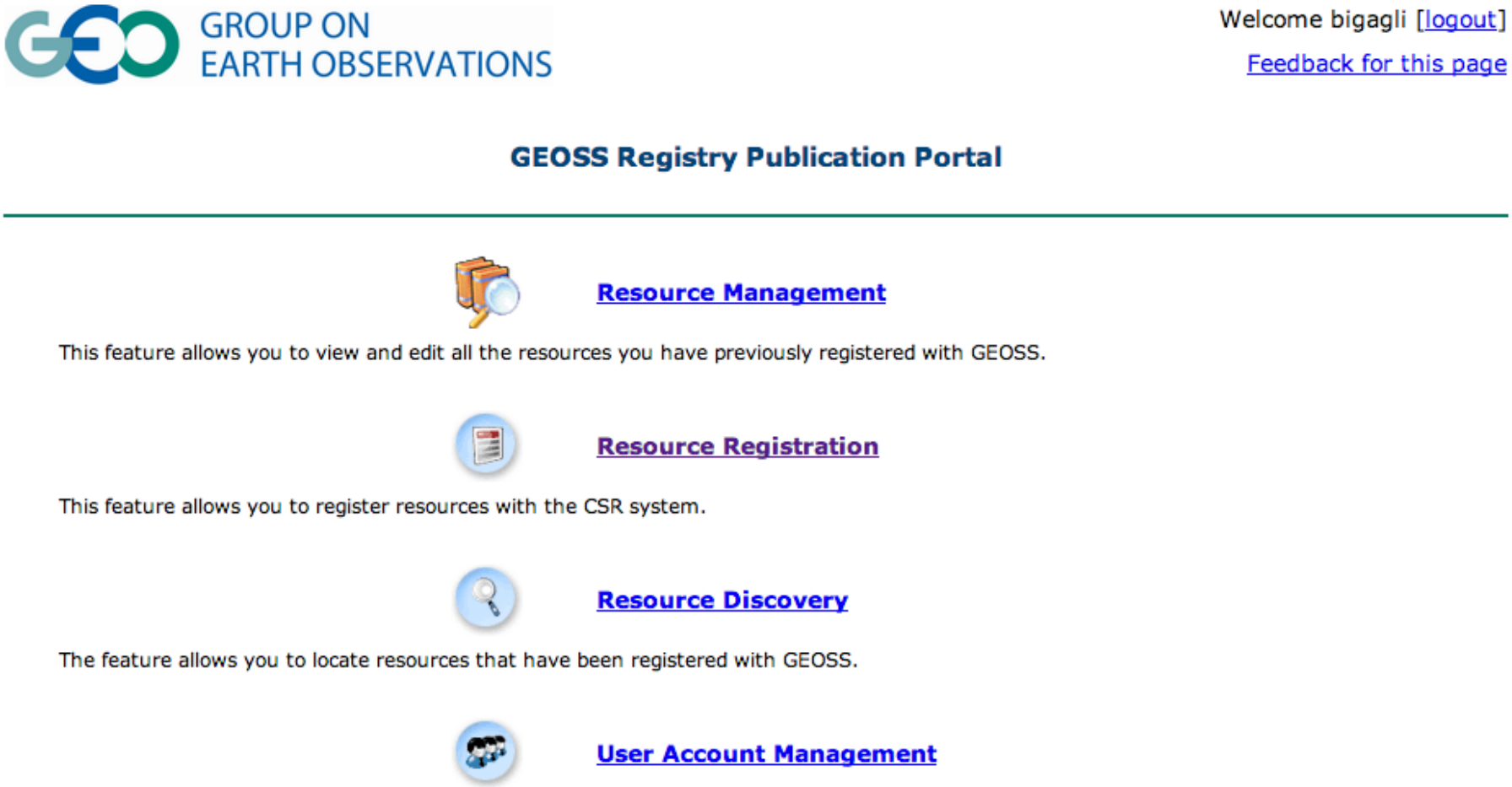


Signing in to the GEOSS CSR Publication Portal allows to register individual components or services

Registering a component to the CSR involves completing a simple on-line form, accessible from the `Components and services` link on the GEO Portal:

http://geossregistries.info/geosspub

You will be asked to sign in (possibly after creating an account) to access the `GEOSS Registry Publication` page.

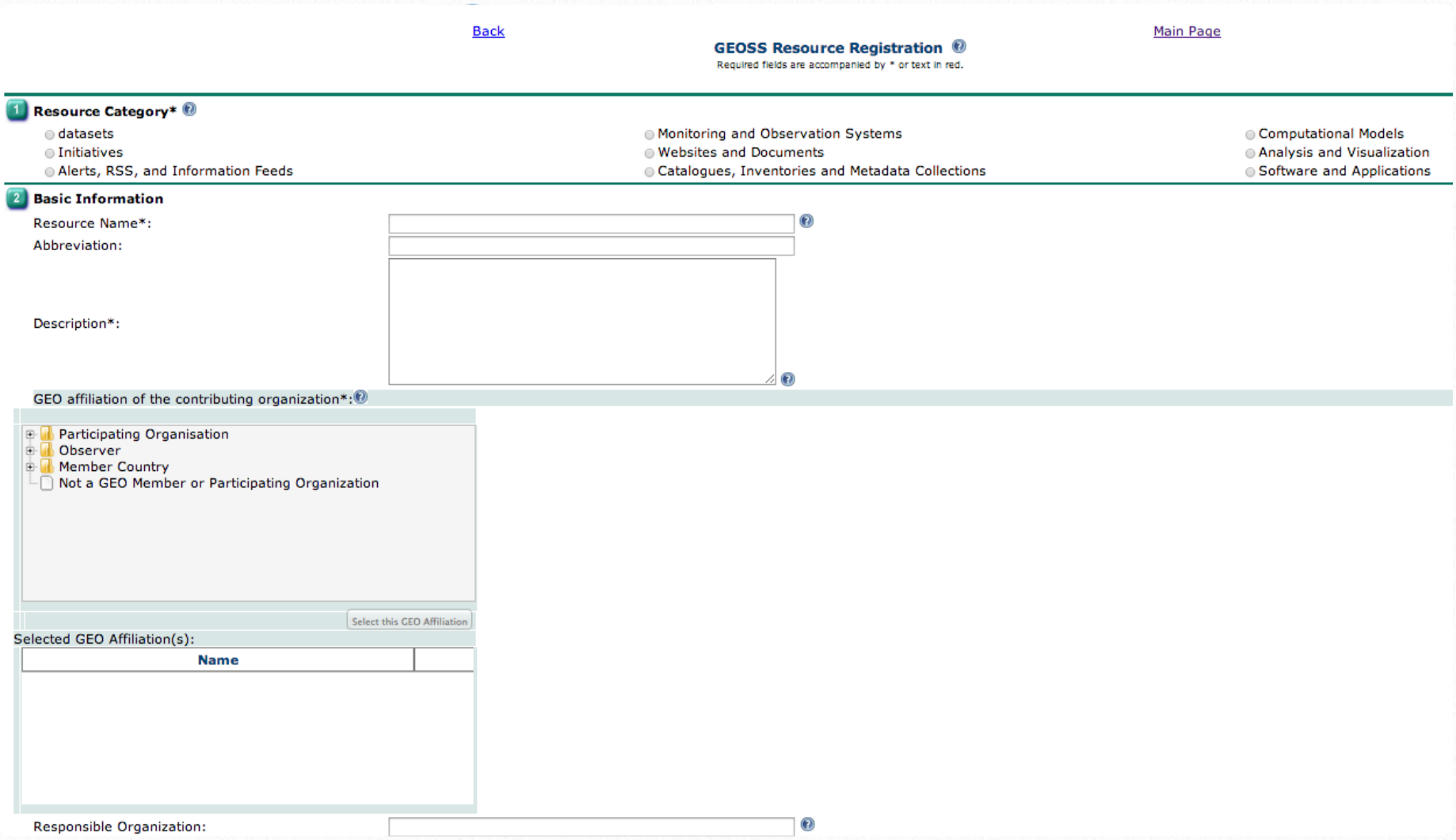


Once logged into the GEOSS CSR Publication Portal, clicking on "Resource Registration" allows to proceed to the registration page

Once logged into the GEOSS Registry Publication Portal, you will need to click on `Resource Registration` to proceed to the registration form.

From here, you may click on `Resource Discovery` to locate resources that have been registered with GEOSS, or on `Resource Management` to manage all your previously registered resources.

You may also manage your account data.



The GEOSS Resource Registration form allows to fully describe the component/service

In the registration page, you need to specify resource category and other basic information such as the resource `name`, `description`, `GEO affiliation`, `URL to Resource`, `contact name`, `email` etc.

To select a GEO affiliation you have to select a node from the `GEO member` list and click the `Select this GEO Affiliation` button. Once a GEO affiliation is selected, its name will be displayed in the `Selected GEO Affiliation(s)` list. You may select more than one affiliation for the component you are registering.

`Societal Benefit Areas` and `Resource Availability` are also required fields for Resource registration. It is needed to select all relevant social benefit areas for your resource and specify the availability.

At the bottom of the page, a check box allows to also send out a Request for Approval notice to the GEOSS CSR Record Review Working Group.

Registered components must be approved by the CSR Record Review Working Group before actual publication

By choosing this option, you ask the GEOSS CSR Record Review Working Group to manually review this record.

You may also request for approval of this resource later in the Resource Management page.

Click "Register the Resource" button when all the required information are ready.

Upon successful registration, your resource will be assigned a Resource Id, like:

urn:geoss:csr:resource:urn:uuid:982ba9c5-e88d-9e53-7565-db9bb2239e35

Only approved records are searchable through the public CSR search interface and accessed by the GEOSS Clearinghouse.

Sharing a big capacity

A big capacity, such as a federation of services, a cyber-infrastructure, a Spatial Data Infrastructure, or a big data system, may include a great number of services and components, whose individual registration in the Clearinghouse may be impractical.

Moreover, big capacities are usually structured in service layers, typically including a catalogue or other high level services, whose functionalities may be better leveraged in the context of the whole e-infrastructure, instead of being suitable for individual consumption.

In that case, it is usually preferable to broker the whole e-infrastructure as a first-level capacity in the GCI, connecting it to GEOSS via the brokering framework.

In this way, all the existing GEOSS users will be able to seamlessly discover and access the shared resources, through the GEO Portal or any other tools in use. Besides, this may reduce possible problems of data duplication.

Provided that the big capacity complies with the GEOSS general interoperability arrangements for data and information sharing and dissemination, you do not have to modify the system, nor to adopt any specific technology or protocol.

The GEO DAB will be in charge of the necessary mediation, mapping, and adaptation processes, as well as of distributing the user requests to the suitable service endpoint of the shared big capacity.

If the big capacity includes a client/portal, you may also want to connect it to the GEO DAB, to take advantage of GEOSS multidisciplinary resources. Also in this case, you do not have to modify the client/portal, as the GEO DAB will be in charge of the necessary mediation.

To share a big capacity as described above, contact the GEO Secretariat. You will be requested to provide the system documentation, for implementing the necessary DAB adaptation modules.

You may still want to register in the CSR the big capacity as a whole, for example for documentation and outreach. In that case, it is sufficient to provide policy and institutional information, neglecting the technological details.

Sharing multiple resources of a community of practice

Communities of practice are defined as distributed groups of people who share a concern, a set of problems, or the interest for a domain, and who deepen their knowledge and expertise in this area by interacting practically on an ongoing basis.

As this definition applies in particular to scientific activity in Earth observation, GEO provides the more specific definition of a GEO Community of Practice (GEO CoP), as a self-organized group of people who commit to working together as part of GEO to foster application of Earth observations for societal benefit in their shared field of interest and expertise.

GEO formally recognizes and supports CoPs. In fact, CoPs play a critical role in implementing GEO's mission, connecting GEO to the broader scientific and user communities, and leveraging the synergies and potential that exist when groups and individuals collaborate toward a common goal. GEO CoPs form when a critical mass of interest and commitment coalesces. Currently, GEOSS has 10 active GEO CoP, and 2 others are emerging.

If you are part of a community of practice and you see the advantages of the GEO network, you may consider becoming an integral part of that network and participate in GEO as a formal GEO CoP. Becoming a GEO CoP is not intended to overwhelm or replace the existing community, but rather to strengthen and advance joint goals and objectives.

The objectives of a CoP include providing a forum for cooperation of activities where GEOSS adds value to existing initiatives, to identify linkages and opportunities for collaborative strategic and technical projects and to coordinate the delivery of some GEOSS targets to enable the realization of societal benefits.

One approach for structuring a CoP so as to be interoperable with GEOSS consists on choosing a common set of technologies for information interchange, a so-called service bus where interoperability is realized; this is the approach chosen by the EU INSPIRE Directive, the GMES initiative, or any other Federated Information System.

An alternative option is to replicate the very approach of the GCI at the CoP scale, that is structuring it as a Brokered System, moving the burden of interoperability to a brokering service, and leaving the participating members autonomous and untouched; this is the approach chosen by the GEO BON initiative, for example, and in fact by GEOSS itself.

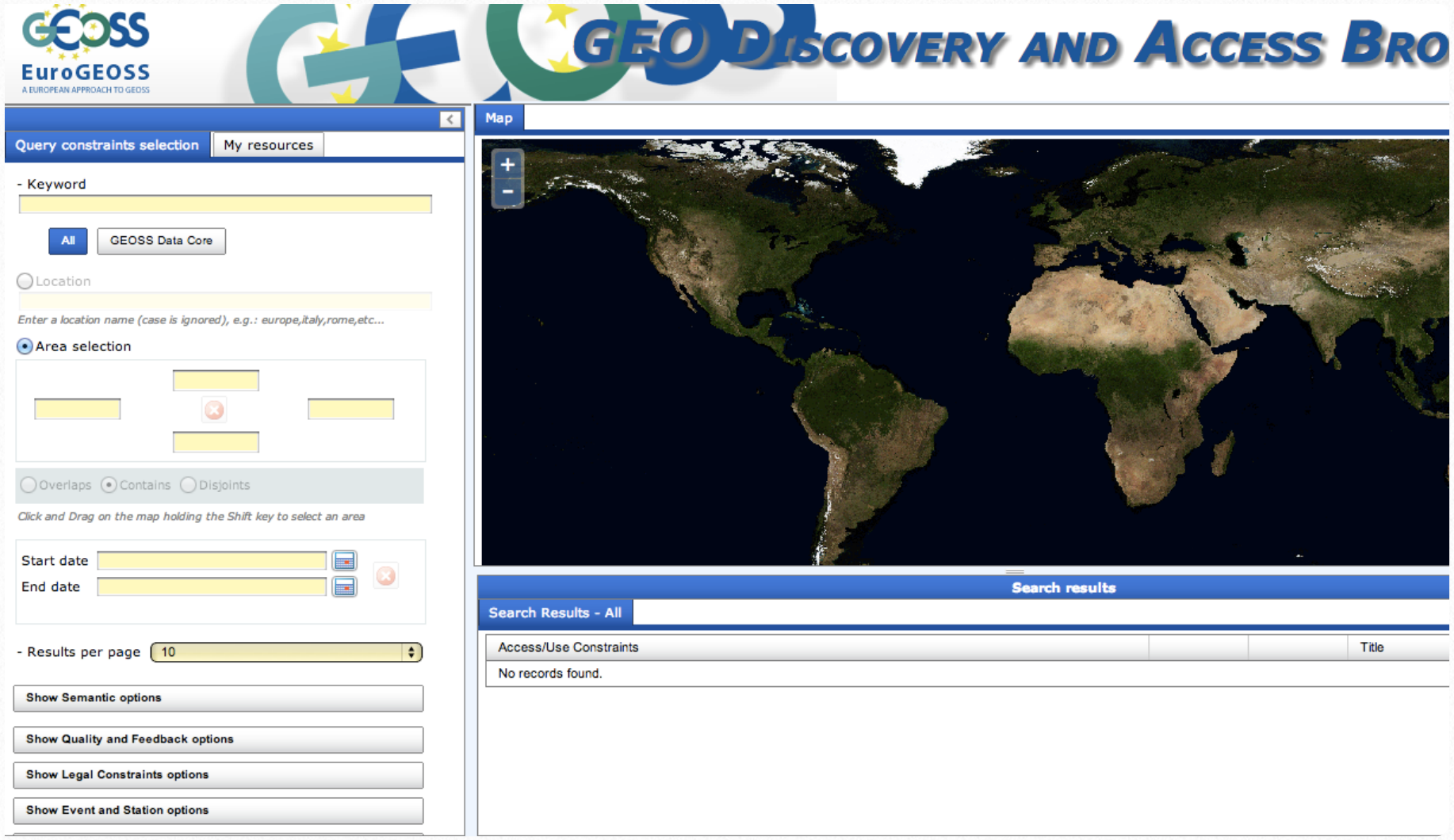
Either way, once the CoP is structured, sharing it with the GEOSS community is similar to the big capacity scenario described in the previous section.

The rest of this section provides information on two software packages that power the GCI brokering framework, which you can install and configure for your CoP:

- GI-cat is a discovery broker, supporting traditional Where-When-What-Who queries, as well as advanced semantically-augmented discovery, result tagging and clustering. GI-cat download and installation instructions are available at:
  http://essi-lab.eu/do/view/GIcat

- GI-axe is an access broker, supporting data download, as well as advanced data processing and transformation, e.g. harmonization of different datasets on the same Coordinate Reference System, resolution, format, etc. GI-axe download and installation instructions are available at:
  http://essi-lab.eu/do/view/GIaxe

You can replicate the GCI Brokered System approach in your community of practice installing GI-cat and GI-axe



GI-cat includes a Web client with a map viewer and GUI controls to specify several query criteria. The results are shown in the lower panel

GI-cat acts as a mediating broker towards disparate catalog and access services. By implementing metadata harmonization and protocol adaptation, it is able to distribute a query and transform the results to a uniform and consistent aggregation.

GI-cat is based on a service-oriented framework of modular components and can be customized and tailored to support different deployment scenarios.

GI-cat itself exposes several interfaces, including profiles of the OGC CSW standard (Core, ISO, ebRIM EO and ebRIM CIM).

By showing sources selection, the query scope can be limited to a subset of the brokered data sources, as well as to individual resources

GI-cat can broker a multiplicity of data sources (e.g. catalogs services, inventories, access services), to discover and access heterogeneous geospatial datasets. Specific components, termed Accessors, implement mediation services for interfacing heterogeneous service providers which expose multiple standard specifications.

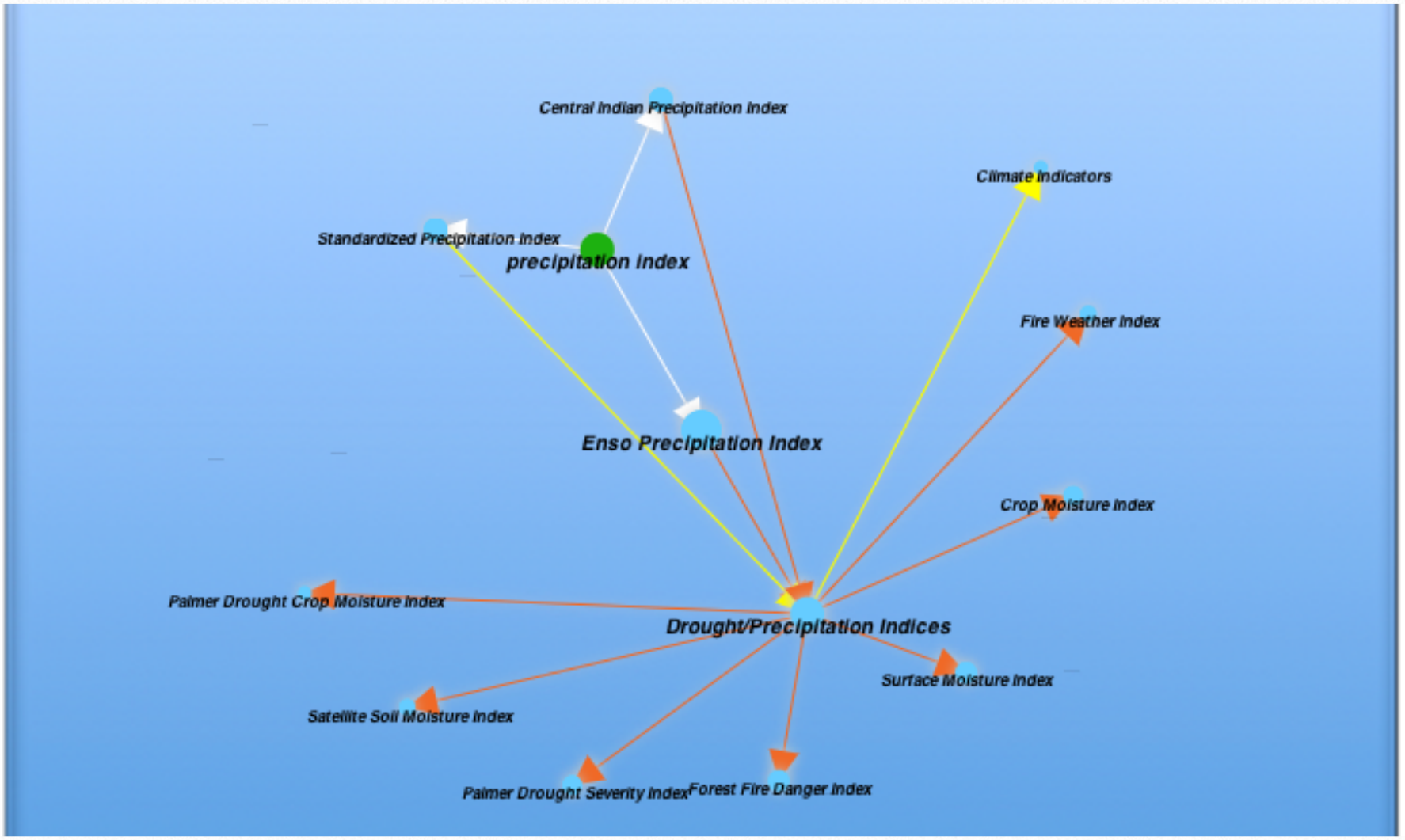
These mediating components map the heterogeneous providers metadata models into a uniform data model based on ISO 19115.

Accessors also implement the query protocol mapping and translate the query requests expressed according to the interface protocols exposed by GI-cat, into the multiple query dialects spoken by the resource service providers.

Query results can be further evaluated, e.g. retrieving their full metadata description, or saved for download

Example user query for the keyword “drought” on the Mediterranean. GI-cat found forty-one results (note that this result may vary depending on how you configured GI-cat), which are ranked (accessible data first) and clustered according to the GEOSS category. By selecting the `GEOSS Data Core` button, the query may be limited to the `GEOSS Data Collection of Open Resources for Everyone`, a distributed pool of documented datasets with full, open and unrestricted access at no more than the cost of reproduction and distribution. GEOSS data providers are encouraged to provide datasets falling into this category.

Additional icons and buttons provide information about each dataset (e.g. its full ISO record) and allow adding it for later download in the “My resources” list.

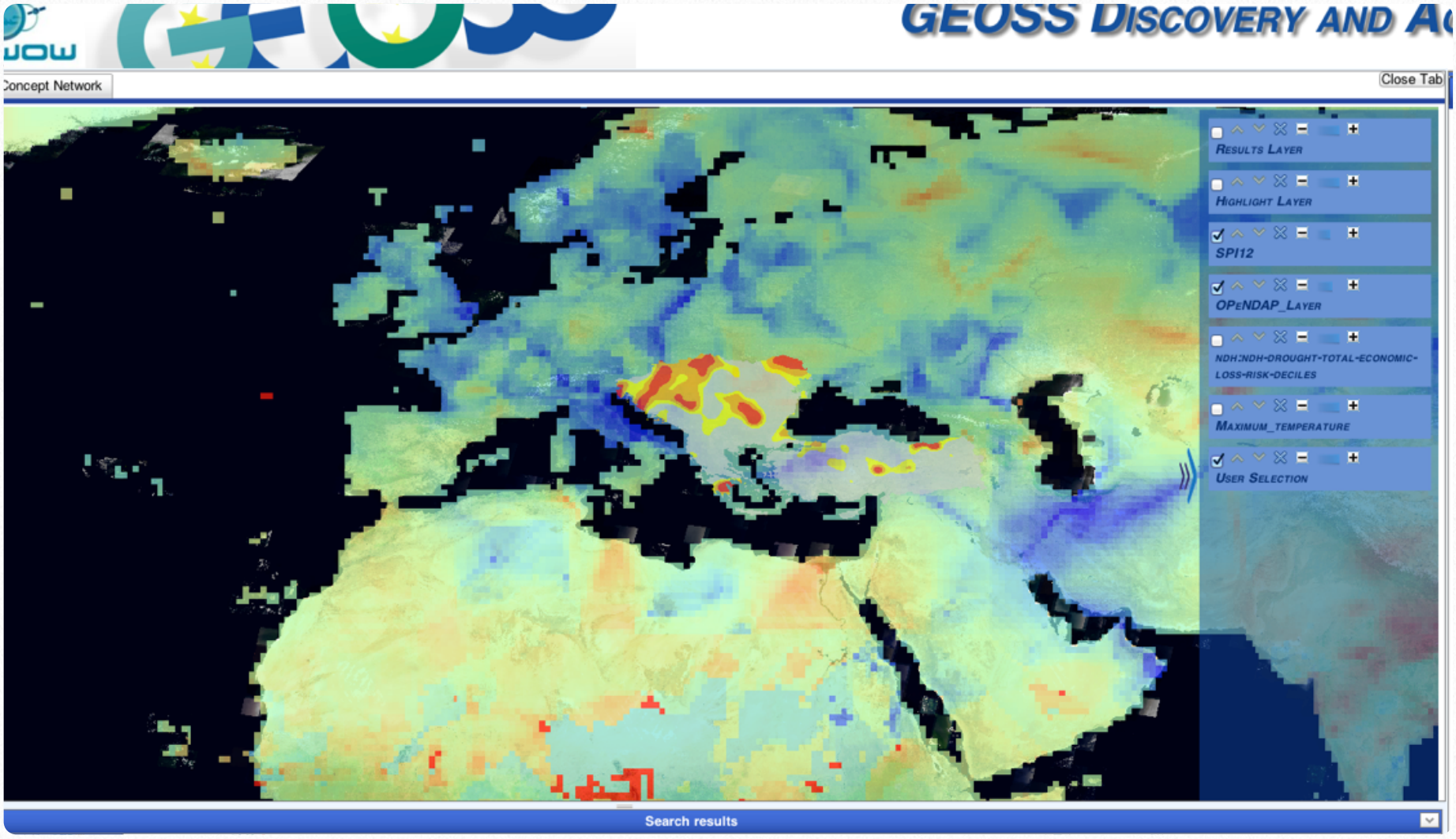


By showing the semantic options, the query scope can be limited to the concepts of interest, independently of the language

In addition to the normal keyword search on title, abstract, and other textual metadata, GI-cat supports querying by conceptual terms selected from a formal ontology.

This supports cross-domain and multilingual queries, as well as generalizations, specializations, etc.

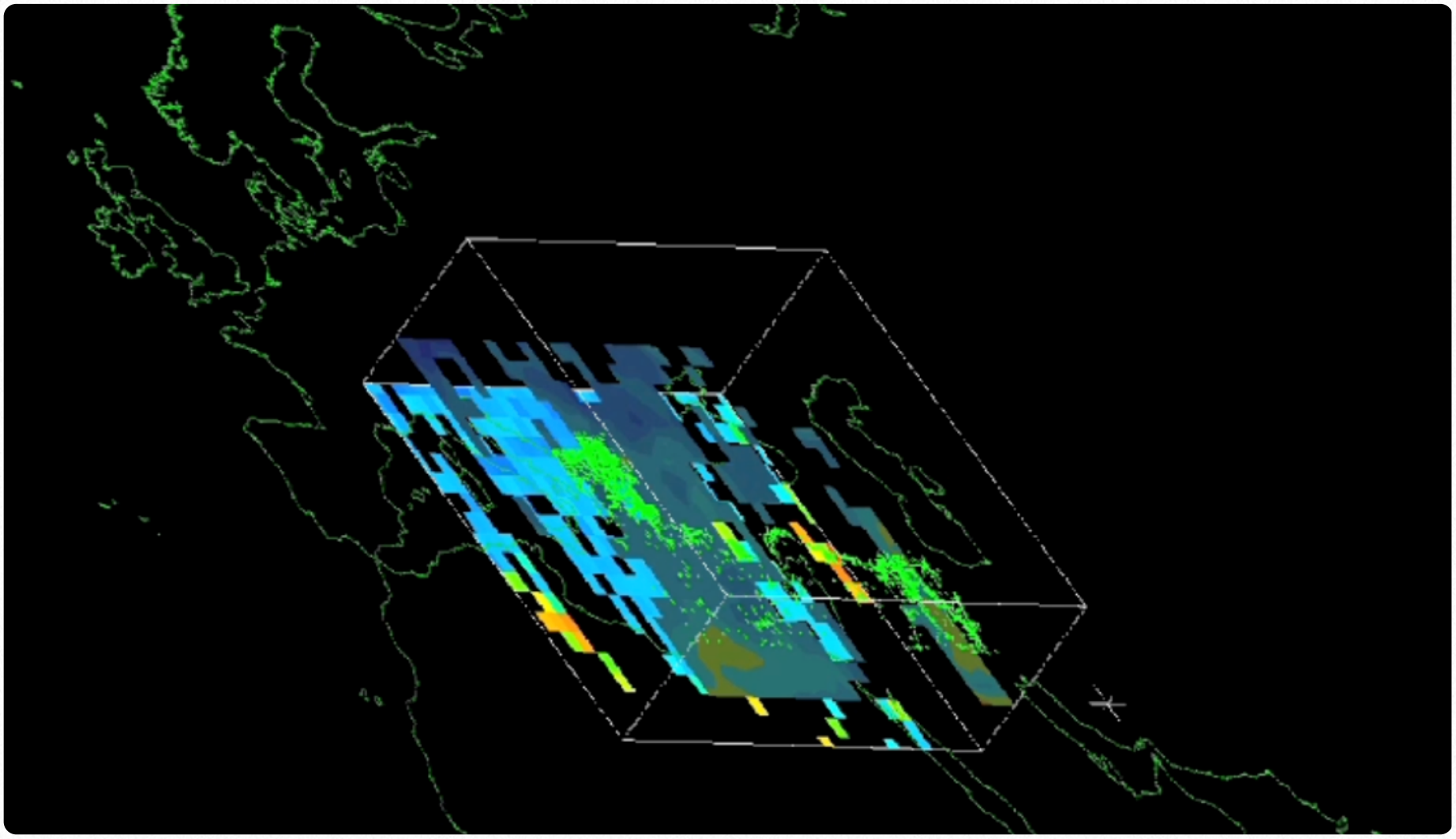
A browsable concept network represents with different colors the relationship between concepts (e.g. “more general”, “more specific”, “related”).



The datasets of interest can be overlaid and previewed, for quick evaluation

GI-cat supports the dynamic visualization of global and regional layers from heterogeneous data sources.

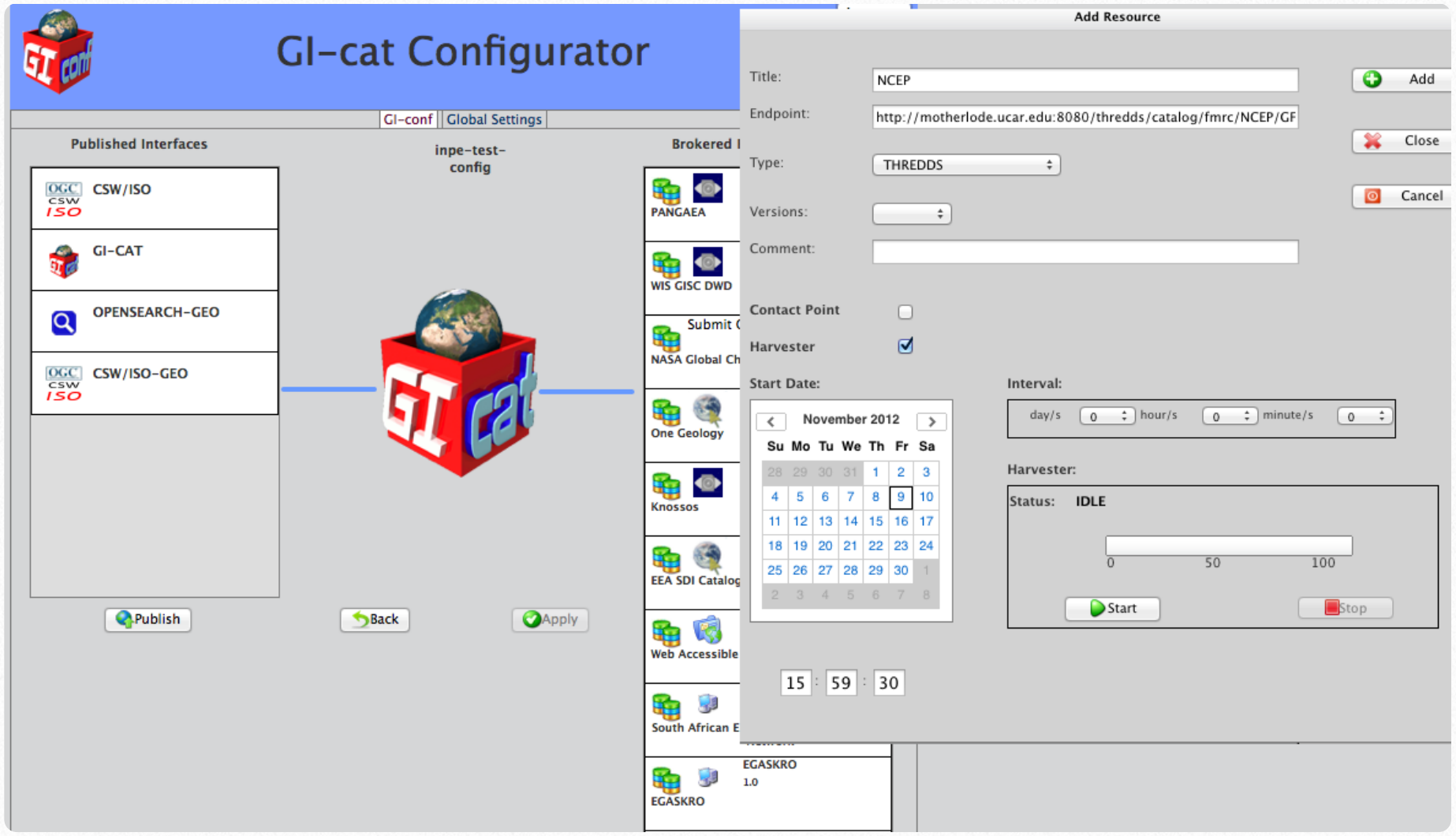
This is transparently enabled by GI-axe, which provides generic portrayal services reconciling data access to the standard OGC Web Map Service, even when the remote source actually support a different service interfaces (e.g. OPeNDAP).



Upon download, heterogeneous datasets can be harmonized to a common grid environment

GI-axe also allows downloading the datasets in the `My resources` list according to user-specified characteristics, with respect to the following features: Coordinate Reference System, spatial resolution, spatial extent, and encoding format.

All the required processing steps are transparent to the user.



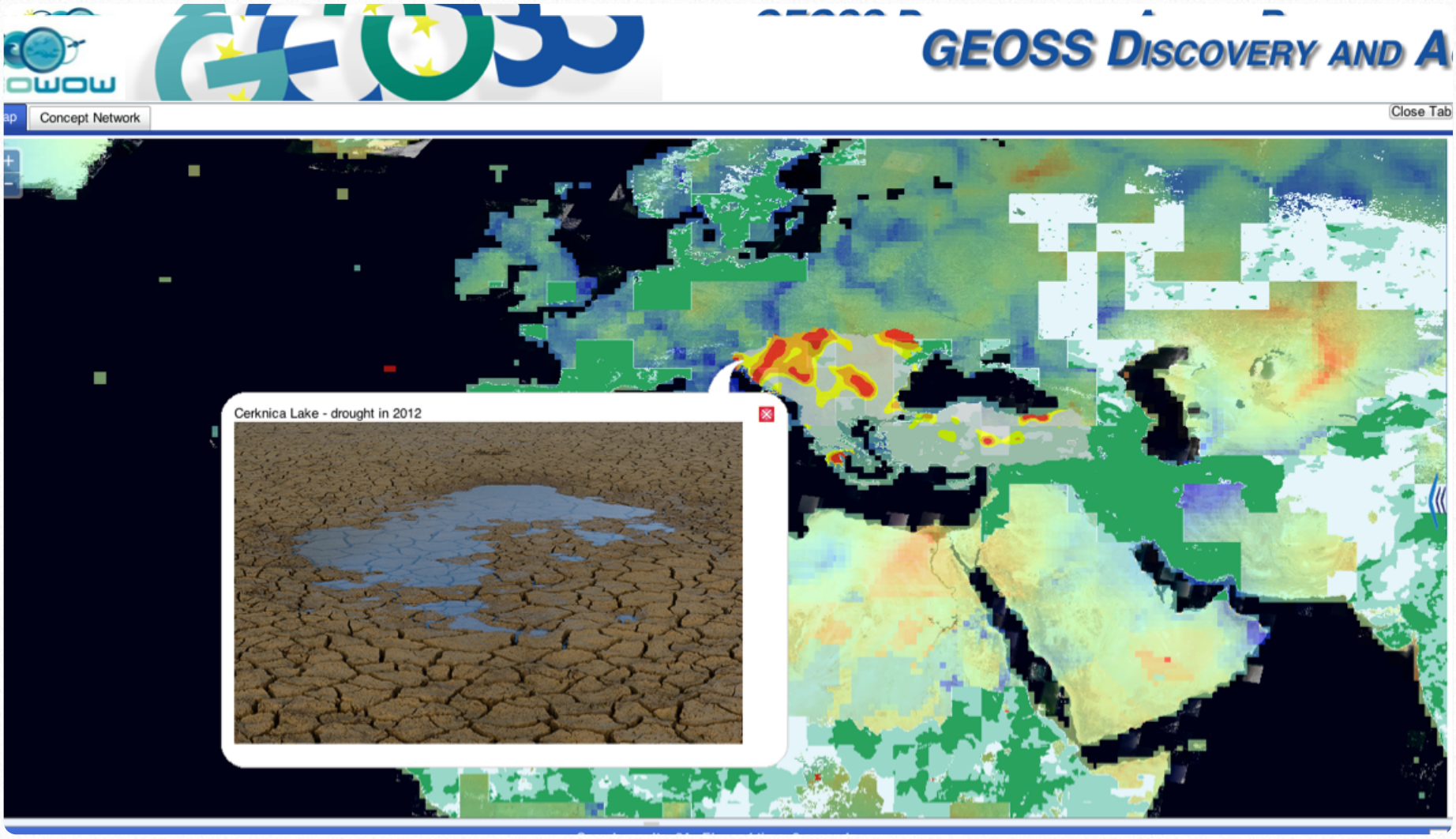
A Web application allows to broker additional data sources and to configure the service endpoints

GI-cat is configurable with respect to the data sources that it brokers, the published interface, the harvesting policies, etc.

Here the user adds one of the NCEP model forecasts repositories to the list of brokered resources:

http://motherlode.ucar.edu/thredds/catalog/fmrc/NCEP/GFS/Global_0p5deg/catalog.xml

GI-cat supports the main OGC Web Services, including WCS, WMS, and CSW, the THREDDS Data Server, SeaDataNet Common Data Index, GBIF, and many more.



GI-cat also supports Web 2.0 data sources, such as Flickr and other crowdsourcing resources

To retrieve recent local images in the area of interest, you can add Flickr as a new data source of the framework.

After selecting Flickr as the target of the next search, you can search and visualize drought-related images in the area of interest.

10

Annexes

Keywords: CSW; GEOSS; GMES; GML; INSPIRE; interoperability; KML; OGC; UNSDI; standards; WCS; WFS; WMS; WPS

What you will learn:

- Main Initiatives
- Standards organizations
- Standards descriptions

Initiatives

Different initiatives at the regional and global level are influencing and promoting the creation of Spatial Data Infrastructures and the use of open standards. They all are concerned about data access, harmonization, standardization, interoperability, seamless integration and services. They coordinate actions that promote awareness and implementation of complimentary policies, common standards and effective mechanisms for the development and availability of interoperable digital geographic data and technologies to support decision making at all scales for multiple purposes. These actions encompass the policies, organizational structures, data, technologies, standards, delivery mechanisms, and financial and human resources necessary to ensure that those working at the global and regional scale are not impeded in meeting their objectives.

Infrastructure for Spatial Information in the European Community (INSPIRE)

Website: http://inspire.jrc.ec.europa.eu/

The Infrastructure for Spatial Information in the European Community, namely INSPIRE, is of particular interest for the EnviroGRIDS project. INSPIRE is a European Directive (entered into force in May 2007 and fully operational by 2019) that aims to create a European Union Spatial Data Infrastructure. This will enable the sharing of environmental spatial information among public sector organizations and better facilitate public access to spatial information across Europe (EU, 2007). When fully implemented, it will, theoretically enable data from one Member State to be seamlessly combined with data from all other States. This is particularly important for activities relating to the environment.

The main purpose of INSPIRE is to support the formulation, implementation, monitoring, and evaluation of Community environmental policies (EU, 2008). Therefore the spatial information considered under the directive is extensive and includes a great variety of topical and technical themes and will be based on Spatial Data Infrastructures established and operated by the Member States.

This initiative wishes to overcome the barriers affecting data access and exchange in Europe, including (EU, 2008):

- Inconsistencies in collection of georeferenced data: geodata are often missing and/or incomplete, or are collected twice by different organizations.
- Lacking of documentation, description (metadata) of the data.
- Geodata are often incompatible and thus cannot be combined.
- Infrastructures used to find, access and use geodata often function in isolation and are incompatible.

- Barriers to sharing: cultural, linguistic, institutional, financial and legal.

In order to overcome these barriers, it has been recognized that it would be necessary to develop a legislative framework asking the Member States to coordinate their activities and to agree on a set of requirements, common standards and processes. In consequence, INSPIRE is unique in the sense that it is an important collaborative and participative process to formulate the directive, create implementing rules and develop relative specifications and services.

INSPIRE seeks to create a European SDI and the INSPIRE Directive defines it: “infrastructure for spatial information means metadata, spatial data sets and spatial data services; network services and technologies; agreements on sharing, access and use; and coordination and monitoring mechanisms, processes and procedures, established, operated or made available in accordance with this Directive”. (EU, 2004)

The end users of INSPIRE include policymakers, planners and managers at the local, national and regional levels, and the citizens and their organizations.

INSPIRE is based on common principles (EU, 2007):

1. Data should be collected only once and kept where it can be maintained most effectively.
2. It should be possible to combine seamless spatial information from different sources across Europe and share it with many users and applications.
3. It should be possible for information collected at one level/scale to be shared with all levels/scales; detailed for thorough investigations, general for strategic purposes.
4. Geographic information needed for good governance at all levels should be readily and transparently available.
5. Easy to find what geographic information is available, how it can be used to meet a particular need, and under which conditions it can be acquired and used.

A step by step approach is used to implement and develop the infrastructure because such an initiative cannot be built from one day to another and is asking Member States to drastically change their existing infrastructure. Thus the implementation of services has been stated just after the adoption of the Directive, whereas the harmonization of INSPIRE data themes will be made in three phases up to 2013.

The European Commission Joint Research Center (JRC) plays a major role in this initiative as it has supported the development of the proposal and now endorses the responsibility of the overall technical coordination of the Directive, providing support to the preparation of the technical rules on implementation, data harmonization, documentation and the required services to discover, view and download data.

The Directive provides five sets of Implementing Rules (IR) that set out how the various elements of the system (metadata, data sharing, data specification, network services, monitoring and reporting) will operate and to ensure that the spatial data infrastructures of the Member States are compatible and usable in a Community and

transboundary context. The Drafting Teams now working on these IRs are composed of international experts and the process includes open consultation – particularly with Spatial Data Interest Communities (SDIC) and Legally Mandated Organizations (LMO).

The Directive specifically states that no new data will need to be collected. However it does require that two years after adoption of the Implementing Rules for data sets and their related services each Member State will have to ensure that all newly collected spatial data sets are available in conformity with the IR. Other data sets must conform to the Rules within 7 years of their adoption. Implementing Rules will be adopted in a phased manner between 2008 and 2012 with compliance required between 2010 and 2019 (EU, 2008).

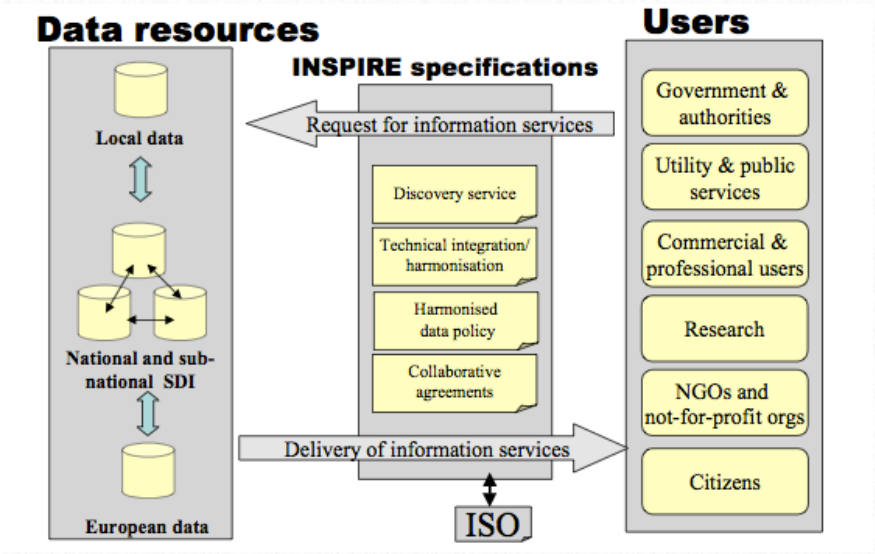


Fig.6: Data and information flow within the INSPIRE framework (Source: INSPIRE).

The envisioned interoperability in INSPIRE is a possibility offered to the user to combine geospatial data and services from different sources across the European Community in a consistent way without involving specific efforts of humans or computers (fig.6). Thus users will spend less time and efforts to integrate data delivered within the INSPIRE framework.

The Directive (EU, 2007) defines 34 “spatial data themes” that have been defined in three Annexes sorted in order of priority. Annex 1 datasets cover the ‘basic’ spatial building blocks such as spatial referencing systems, geographic names, addresses, transport networks, hydrography and land parcels. Because of the range of data types involved, the impact of INSPIRE is comprehensive. Annex 1 datasets have to be prepared and made available from 2011, with the other Annexes at later dates. In order to enable full system interoperability across the EU, each spatial data theme is described in a data specification. As mentioned on the INSPIRE website “The process for developing harmonized data specifications is designed to maximize the re-use of existing requirements and specifications, in order to minimize the burden for Member States’ organizations at the time of implementation. The consequence of this is that the process of developing Implementing Rules for interoperability of spatial datasets and services may be perceived as being complex: it involves a large number of stakeholders, with many interactions and consultations”.

Finally, all the data, information and services shared within INSPIRE would be accessible through the INSPIRE Community Geoportal. For Luraschi et al. (2009) because the geoportal does not store or maintain data and metadata, it could be seen as a gateway aggregating a number of instances of specific geospatial informa-

tion services distributed across the Europe and maintained by the organization responsible for the data.

According to the INSPIRE network architecture (EU, 2008), Member States shall establish, operate and provide access to the following network services (fig.7):

- Discovery services: support discovery of data, evaluation and use of spatial data and services through their metadata properties
- View services: as a minimum, display, navigate, zoom in/out, pan, or overlay spatial data sets and display legend information and any relevant content of metadata.
- Download services: enabling copies of complete spatial data sets, or parts of such sets, to be downloaded.
- Transformation services: enabling spatial data sets to be transformed (projection and harmonization).
- Invoke spatial data services: enabling data services to be invoked.

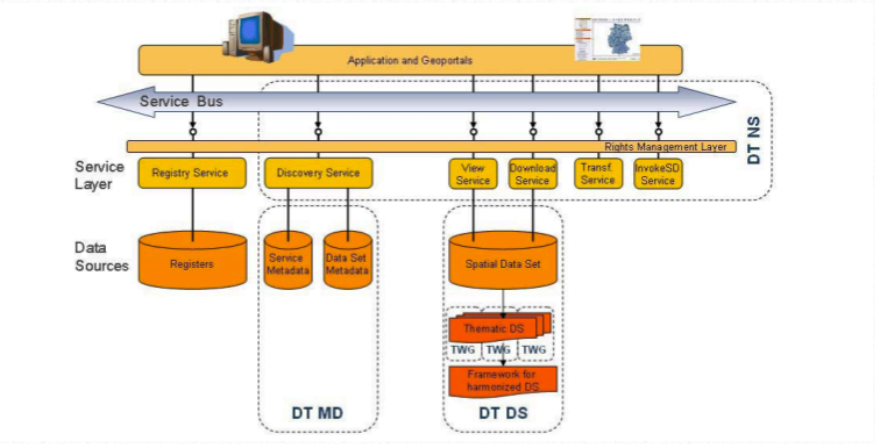


Fig.7: INSPIRE network architecture (Source: INSPIRE).

Global Earth Observation System of Systems (GEOSS)

Website: http://www.earthobservations.org

The GEOSS is being established by the intergovernmental Group on Earth Observations (GEO) and is a worldwide effort to build a system of systems on the basis of a 10-Year Implementation Plan for the period 2005 to 2015 (GEO, 2005). GEO is voluntary partnership of governments and international organizations where membership and participation is contingent upon formal endorsement of the Implementation Plan mentioned above.

GEOSS is an effort to connect already existing SDIs and Earth Observations infrastructures and thus will not create and/or store data but rather works with and build upon existing systems. GEOSS, through its developing GEOportal, is foreseen to act as a gateway between the producers of environmental data and the end users, with the aim of enhancing the relevance of Earth observations for the global issues and to offer a public access to comprehensive, near-real time data, information and analyses on the environment (GEO, 2007).

GEOSS aims to provide a broad range of so-called Societal Benefits Areas (GEO, 2005):

- Reducing loss of life and property from natural and human-induced disasters,
- Understanding environmental factors affecting human health and well-being,
- Improving the management of energy resources,
- Understanding, assessing, predicting, mitigating, and adapting to climate variability and change,

- Improving water resource management through better understanding of the water cycle,
- Improving weather information, forecasting and warning,
- Improving the management and protection of terrestrial, coastal and marine ecosystems,
- Supporting sustainable agriculture and combating desertification, and
- Understanding, monitoring and conserving biodiversity.

The mechanisms for data and information sharing and dissemination are presented and described in the 10-Year Implementation Plan Reference Document (GEO, 2005) where information providers must accept and implement "a set of interoperability arrangements, including technical specifications for collecting, processing, storing , and dissemination shared data, metadata and products. GEOSS interoperability will be based on non-proprietary standards, with preference to formal international standards. Interoperability will be focused on interfaces, defining only how system components interface with each other and thereby minimizing any impact on affected systems". GEOSS is based on existing technologies using internet-based services.

Moreover members must fully endorse the following data sharing principles:

1. There will be full and open exchange of data, metadata, and products shared within GEOSS, recognizing relevant international instruments and national policies and legislation.
2. All shared data, metadata, and products will be made available with minimum time delay and at minimum cost.
3. All shared data, metadata, and products being free of charge or no more than cost of reproduction will be encouraged for research and education.

These principles push data owners to go "open" and to share their data using standards and thus becoming interoperable.

United Nations Spatial Data Infrastructure (UNSDI)

Website:
http://www.ungiwg.org/content/united-nations-spatial-data-infrastructure-unsdi

The United Nations Spatial Data Infrastructure in an initiative conducted by the United Nations Geographic Working Group (UNGIWG) that aims at building an institutional and technical mechanism to establish a coherent system to exchange data and services concerning geospatial data and information within the United Nations, and also supporting SDI development activities in the Member Countries.

As stated in the UNSDI Compendium (UNGIWG, 2007), "Historically, the production and use of geospatial data have been accomplished within the United Nations by its component organizations, in accordance with their individual needs and expertise. But concordant with the recent, rapid increase in the use of geospatial data for UN activities is the need for greater coherence in its management system-wide".

This initiative aims to contribute to the general mission of the United Nations to maintain peace and security, to address humanitarian emergencies, to assist sustainable development and support achievement of the UN Millennium Development Goals. The hope is to facilitate efficient access, exchange and utilization of geo-referenced information in order to make the UN system more effective, increase the system coherence and support its "Delivering as One" policies.

The UNSDI provides an institutional and technical foundation of policies, interoperable standards procedures and guidelines that enable organizations and technologies to interact in a way that facilitates spatial discovery, evaluation and applications (UNGIWG, 2007).

Global Monitoring for the Environment and Security (GMES)

Websites: http://www.gmes.info/ and http://ec.europa.eu/gmes/

The GMES is a European program, coordinated by the European Commission and European Space Agency, for the implementation of a European capacity for Earth observation with the objective to monitor and better understand the environment and thus contribute to the security of every citizen. This initiative aims at providing decision-makers and other users who rely on strategic information with regard to environmental and security issues an autonomous, independent and permanent access to timely, reliable and accurate data and services.

The objective is to integrate data on atmosphere, oceans and continental/land processes giving an overview of the state of health of our planet and to deliver information through five thematic areas (served by different services) covering:

- Land,
- Marine,
- Emergency,
- Atmosphere,
- Security.

To gather data and information on Earth Observation, GMES proposes to build and infrastructure around four components:

- Space: environmental satellites
- In-situ measurements: ground-based and air-borne sensors
- Data harmonization and standardization
- Services to users.

Like various other data sources, Earth-observation-based services already exist in Europe but are dispersed and fragmented at national and regional level avoiding a sustainable observation capacity (meaning that long-term availability of information is not guaranteed). In consequence, GMES is the answer of the European Commission to develop a reliable and sustainable Earth Observation system and also contributing the GEOSS initiative.

GMES stated that "By securing the sustainability of an information infrastructure necessary to produce output information in the form of maps, datasets, reports, targeted alerts, etc…, GMES

helps people and organizations to take action, make appropriate policy decisions and decide on necessary investments. GMES also represents a great potential for businesses in the services market, which will be able to make use of the data and information it provides according to a full and open access principle." (GMES, 2009).

Global Spatial Data Infrastructure (GSDI)

Website: http://www.gsdi.org

The mission of the GSDI Association, a worldwide inclusive body of organizations, agencies, firms and people, is to "promote international cooperation and collaboration in support of local, national, and international spatial data infrastructure developments that will allow nations to better address social, economic and environmental issues of pressing importance" (GSDI, 2004).

Its purpose is to focus on communication, education, scientific, research and partnership efforts to support all societal needs for access to and use of spatial data.

This is an association, guided by a board and funded by membership fees, to:

- Serve as a point of contact and effective voice for those in the global community involved in developing, implementing and advancing spatial data infrastructure concepts,
- Foster spatial data infrastructures that support sustainable social, economic, and environmental systems integrated from local to global scales, and
- Promote the informed and responsible use of geographic information and spatial technologies for the benefit of society.

The GSDI community aims to truly develop and achieve the goal of a Global Spatial Data Infrastructure relying on international and open standards, guidelines and policies to enhance data management and access, and support global economic growth, and associated social and environmental objectives (UNGWIG, 2007), through interoperable standards-based services, systems, softwares and products that operate in a web-based environment.

This vision is guided by five goals (Stevens, 2005):

- Continue to promote and develop awareness and exchanges on infrastructure issues for all relevant levels from local to global.
- Promote and facilitate standards-based data access/discovery through the Internet.
- Promote, encourage, support, and conduct capacity building.
- Promote and conduct SDI development research.
- Collaborate with others to accomplish its Vision and Goals.

To support this vision, the GSDI association acts as a platform and offers a vast choice of publications, conferences, workshops, projects and programs allowing people interested in SDI to learn, exchange, share their knowledge and expertise, because capacity building is one of the key points of SDIs.

Standards organizations relevant for GIS/SDI

In the field of geomatics there are several organizations involved in publishing standards to effectively achieve the goal of interoperability. Such standards are increasingly important in the geospatial community allowing the increase of interoperability between systems and data and thus to “geo-enable” the Web.

Open Geospatial Consortium (OGC)

Website: http://www.opengeospatial.org

The Open Geospatial Consortium (OGC) is a non-profit, international, voluntary consortium of more than 380 companies, government agencies and universities that is leading the development of standards for the geospatial community. Its approach is based on a member-driven consensus process to develop open and publicly available standards and software application programming interface for the geospatial community (UNGIWG, 2007). These standards offer to the developers the possibility to create complex georeferenced information and services accessible to a wide variety of applications and share data in a standardized and interoperable way.

The OGC standards are based on a generalized architecture presented in the Abstract Specification and Reference Model (OGC, 2007). On top of the Abstract Specification, there is a set of standards that have been developed and/or proposed to serve specific needs of the Geographical Information community in order to be interoperable.

These standards are mostly based upon the use of the http protocol to interact through messages over the Internet. In the last two years, the OGC members have been looking with interest to a more common approach used in the Service Oriented Architecture using SOAP protocol and WSDL bindings. There is also work in progress around the Representational State Transfer (REST) protocol for web services.

The OGC is also closely working with the International Organization for Standardization (ISO) through a partnership with the ISO Technical Committee 211 (TC211) to promote and endorse the OGC standards to a higher level of standardization becoming part of the ISO 19100 series. For example, the WMS or the GML are now ISO standards.

The OGC vision is the realization of a full societal, economic and scientific benefits of integrating electronic location resources into commercial and institutional processes worldwide. Its mission is to serve as a global forum for the collaboration of developers and users of spatial data products and services, and to advance the development of international standards for geospatial interoperability.

More specifically the OGC aims to (http://www.opengeospatial.org/ogc/vision):

- Provide free and openly available standards to the market, tangible value to Members, and measurable benefits to users.
- Lead the creation and establishment of standards that allow geospatial content and services to be seamlessly integrated into business and civic processes, the spatial web and enterprise computing.
- Facilitate the adoption of open, spatially enabled reference architectures in enterprise environments worldwide.
- Advance standards in support of the formation of new and innovative markets and applications for geospatial technologies.
- Accelerate market assimilation of interoperability research through collaborative consortium processes.

It must be noticed that the OGC and its members want to help users and developers to make usage of OGC's standards offering them different resources (e.g. technical documents, training, best practices) through the OGC Network (http://www.ogcnetwork.net/).

International Organization for Standardization (ISO)

Website: http://www.iso.org

The International Organization for Standardization (ISO), the world's largest developer, publisher and promoter of international standards, is a non-governmental organization made of a network of more than 160 countries representatives (one per country) with a central secretariat based in Geneva (Switzerland) that coordinates the system.

Even if the main focus of ISO is the development of technical standards, they have an important impact on the economy and the society because many members are coming from a governmental structure or from the private sector. Therefore, ISO is an ideal place to build consensus and solutions that meet the needs and requirements from both the economy and the society.

Within the ISO system there is a Technical committee (http://www.isotc211.org/) whose main area of interest are the Geographical information and the Geomatics aiming to establish a structured set of standards for information concerning objects or phenomena that are directly or indirectly associated with a location relative to the Earth.

At the present day, they have more than 55 standards in the field of Geographical information specifying methods, tools and services for data management (including definition and description), acquiring, processing, analyzing, accessing, presenting and transferring such data in digital/electronic form between different users, systems and locations.

The World Wide Web Consortium (W3C)

Website: http://www.w3c.org

The World Wide Web Consortium (W3C) is an international consortium where members, organi-

zations, staff and the public work together to create and develop Web standards, protocols and guidelines. Since 1994, the W3C has published more that 110 Recommendations (the W3C standards) aiming to " lead the World Wide Web to its full potential by developing protocols and guidelines that ensure long-term growth for the Web", accommodating the growing diversity of people, hardware and software and ensuring the core principles and components of these standards would be supported by everyone.

For the W3C it is crucial to reach the goal of the "web interoperability" allowing the Web to reach its full potential by using technologies that must be compatible with one another and allowing any hardware and/or software to access the Web. By publishing open and non-proprietary standards, the W3C seeks to avoid market fragmentation and thus Web fragmentation.

Therefore the W3C engages in education and outreach, develops software and interoperable technologies that support this mission and acts as an open and vendor-neutral forum for discussion about the Web.

Organization for the Advancement of Structured Information Standards (OASIS)

Website: http://www.oasis-open.org/

The Organization for the Advancement of Structures Information Standards is a non-profit, global consortium (with 5000 members coming from more than 600 organizations in 100 countries) that drives the development, convergence and adoption of open standards for the global information society, the so-called e-business.

OASIS produces different web standards concerning the following categories: Web Services, e-Commerce, Security, Law & Government, Supply Chain, Computing Management, Application Focus, Document-Centric, XML Processing, Conformance/Interop, and Industry Domains.

Standards description

ISO and OGC are providing a lot of different specifications regarding geographical data but in the context of the EnviroGRIDS project we propose to concentrate on those that are mostly used in the geospatial community. The aim is to place the first building blocks of a regional SDI for the Black Sea catchment.

The general aim of these standards is to abstract data delivery mechanisms from physical storage formats and offer services that could be consumed by applications through different interfaces.

The OGC defines a general architecture for the geoportal (OGC, 2004) called The Geospatial Portal Reference Architecture. It provides the basis for an open, vendor-neutral portal that is intended to be a first point of discovery for geospatial content in the context of designing and implementing the Spatial Data Infrastructures. The Geospatial Portal Reference Architecture is founded on the tenants of a Service Oriented Architecture (SOA). An SOA is an architecture that represents software functionality as discoverable services on a network yielding the following benefits:

- Easier extension of legacy logic to work with new business functionality
- Greater flexibility to change without the need to constantly re-architect for growth
- Cost savings by providing straight-forward integration.

The Geospatial Portal Reference Architecture specifies also four services that are needed for creating a interoperable and standardized geoportal (fig.8):

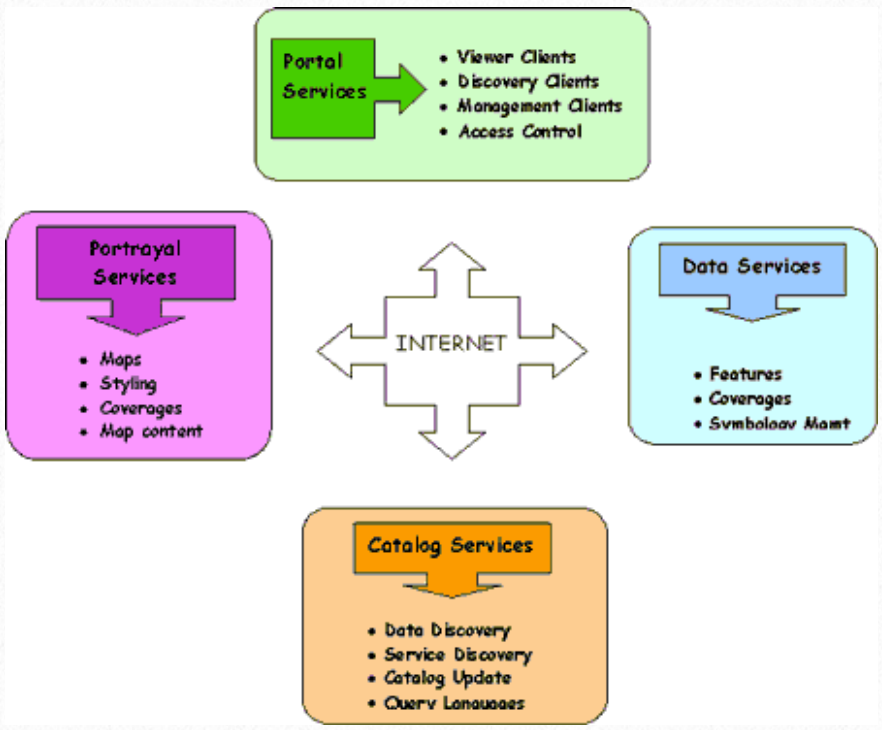


Fig.8: Geospatial Portal Reference Architecture (Source: GeoNetwork).

- Portal Services: provide the single point access to the geospatial information on the portal. In addition, these services provide the management and administration of the portal.
- Catalog Services: used to locate geospatial services and information wherever it is located and provide information on the services and information if finds to the user.

- Portrayal Services: used to process the geospatial information and prepare it for presentation to the user.
- Data Services: used to provide geospatial content and data processing.

To implement and deploy these different service classes, the OGC propose to use web services that are applications running on a computer connecting to a remote web service via a URL allowing access to distributed data and services. As stated by the CGDI “Web service architectures provide a distributed environment in which you can deploy and invoke services using standard Internet protocols. In this context, a service is a collection of operations, accessible through one or more interfaces, that allows you to evoke a behavior of value to you.” (http://www.geoconnections.org/publications/Technical_Manual/html_e/s4_ch10.html#10.1)

Using such a Service Oriented Architecture (SOA) provides a distributed computing platform over a network, typically the Internet, allowing to publish standardized services no matter how it is implemented or on which platform it is executed. This is leveraging the full potential of the interoperability and thus web services to be seamlessly coupled, reusable and available for a variety of applications.

A traditional open web service must have the ability to describe its capabilities and provide a standard way to communicate with it, enabling applications and other web services to communicate and interoperate. Through OGC standards, different GIS softwares and/or components can interoperate, work together and exchange information over a network by means of agreed standards.

For example, when two softwares implement the same OGC standard, they can immediately work together without the necessity to develop new components to translate from one file format (used in one software) to another file format (used in a second software). This means that in a SOA environment that implement OGC standards, a user can access in a transparent way the data stored in different databases, with different formats, and running on different Operating Systems.

Without interoperability and standardization, accessing and integrating different data sources is really difficult or in the worst case impossible. This leads to a fragmentation of geospatial data sources and limit organizations to work only within a single software package.

Catalogue Service for the Web (CSW)

OGC Catalogue Service Specification: http://www.opengeospatial.org/standards/cat

The Catalogue Service defines an interface to publish, discover, search and query metadata about georeferenced data, services and related resources. CSW uses queryable properties, which enable clients to search for geospatial resources by subject, title, abstract , data format, data type, geographic extent, coordinate reference system, originator, publisher, purpose,...

Fig.9: GeoNetwork, a catalogue system using CSW.

Web Map Service (WMS)

OGC Web Map Service Specification: http://www.opengeospatial.org/standards/wms

The Web Map Service defines an interface that allows a client to retrieve maps of georeferenced data. In WMS context, a map means a graphical representation (jpeg ,gif or png files) of a geospatial data meaning that a WMS service does not give access to the data itself. It is used for mapping purposes and can be combined with other WMS services.

A traditional WMS interface, invoked by a URL, consists of the following operations:

- *GetCapabilities:* answer to a client telling him what kind of layers are available and which one are queryable.
- *GetMap:* produce a map as a picture showing selected layers,
- *GetFeatureInfo:* answer simple queries about the content of the map

As seen on the following examples, invoking a WMS service need to specify different parameters (mandatory or optional) in the URL. For the purpose of this guideline we will focus our attention on the basic operations of the service that provides map layers in predefined styles (made available by the data provider) thus we will not discuss the Styled Layer Descriptor (SLD) capabilities.

- Example of a WMS URL with a GetCapabilities request:

http://metafunctions.grid.unep.ch:8080/geoserver/ows?service=WMS&request=GetCapabilities&version=1.3.0

The GetCapabilities operation returns to the user an XML document describing the service and the data sets available from which either desktop and/or web clients may request maps. This operation is common for all the OWS and is presented in details in the OpenGIS Web Service Common Implementation Specification (OGC, 2007). To invoke the operation, the user has only to define the service and the request parameters.

```
- <wfs:WFS_Capabilities version="1.1.0" xsi:schemaLocation="http://www.opengis.net/wfs http://preview.grid.unep.ch:8080/geoserver/schemas
 /wfs/1.1.0/wfs.xsd" updateSequence="262">
  - <ows:ServiceIdentification>
      <ows:Title>enviroSDI Web Feature Service</ows:Title>
    - <ows:Abstract>
        enviroSDI is the Spatial Data Infrastructure of the UNEP/DEWA/GRID-Europe (http://www.grid.unep.ch). This is the reference
        implementation of WFS 1.0.0 and WFS 1.1.0, supports all WFS operations including Transaction.
      </ows:Abstract>
    - <ows:Keywords>
        <ows:Keyword>enviroSDI</ows:Keyword>
        <ows:Keyword>UNEP</ows:Keyword>
        <ows:Keyword>GRID</ows:Keyword>
        <ows:Keyword>EUROPE</ows:Keyword>
        <ows:Keyword>WFS</ows:Keyword>
        <ows:Keyword>GEOSERVER</ows:Keyword>
      </ows:Keywords>
      <ows:ServiceType>WFS</ows:ServiceType>
      <ows:ServiceTypeVersion>1.1.0</ows:ServiceTypeVersion>
      <ows:Fees>NONE</ows:Fees>
      <ows:AccessConstraints>NONE</ows:AccessConstraints>
  </ows:ServiceIdentification>
  - <ows:ServiceProvider>
      <ows:ProviderName>UNEP/DEWA/GRID-Europe</ows:ProviderName>
    - <ows:ServiceContact>
        <ows:IndividualName>Gregory Giuliani</ows:IndividualName>
        <ows:PositionName>enviroSDI coordinator</ows:PositionName>
      - <ows:ContactInfo>
        - <ows:Phone>
            <ows:Voice>+41 22 917 84 17</ows:Voice>
            <ows:Facsimile/>
```


Fig.10: Example of the XML file returned after a GetCapabilites request.

- Example of a WMS URL with a GeMap request:

http://preview.grid.unep.ch:8080/geoserver/wms?bbox=84.95293,19.82194,-74.13126,23.19403&styles=&Format=image/png&request=GetMap&version=1.1.1&layers=preview:cy_buffers&width=640&height=309 &srs=EPSG:4326

The GetMap operation is the most important of the three basic operations of the WMS interface as it returns to a client request a map of selected geospatial layers.

In comparison of the GetCapabilities request that needs only two parameters, we can see on the above example that GetMap operation needs several parameters (also mandatory or optional) that we describe hereafter:

Mandatory parameters for the GetMap operation:

- BBOX: coordinates of the bounding box following minx,miny,maxx,maxy,
- STYLES: list of style names separated by a comma. It's necessary to have an exact correspondence between the number of layers and the number of styles. If this parameter has a empty value, the default style provided by the data custodian will be applied.
- FORMAT: graphical format of the returned map (eg: image/png, image/gif, image/jpeg).
- REQUEST: value “GetMap”, this the request itself to invoke the specific operation.
- VERSION: the version of the specification.
- LAYERS: list of selected layers separated by a comma.
- WIDTH: specify the width of the returned map (in pixels).
- HEIGHT: specify the height of the returned map (in pixels).
- SRS: identifier of the Spatial Reference System.

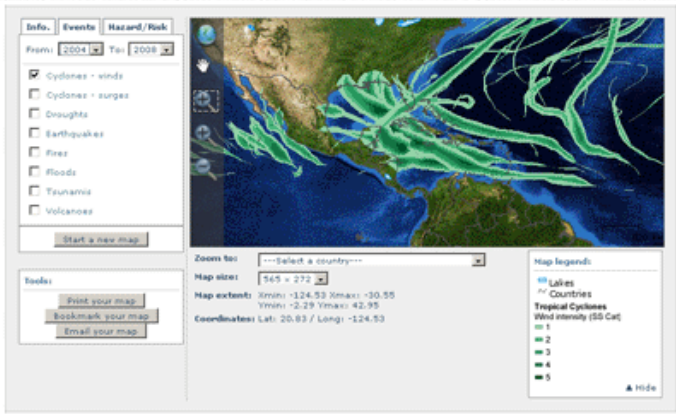


Fig.11: Returned image after a WMS request.

- Example of a WMS URL with a GetFeatureInfo request:

http://webmapping.mgis.psu.edu/geoserver/wms?version=1.1.1&request=getfeatureinfo&layers=topp:states&styles=population&SRS=EPSG:4326&bbox=-125,24,-67,50&width=400&height=200&format=text/html&X=100&y=100&query_layers=topp:states

The GetFeatureInfo operation is used to query the attribute table of a selected layer and get information on a specific feature. For example, a user can click on point of a map (retrieved by a

GetMap request) and he obtains more information.

Mandatory parameters for the GetFeatureInfo operation:

- VERSION: the version of the specification.
- REQUEST: value “GetFeatureInfo”, this the request itself to invoke the specific operation.
- LAYERS: list of selected layers separated by a comma.
- STYLES: list of style names separated by a comma. It's necessary to have an exact correspondence between the number of layers and the number of styles. If this parameter has a empty value, the default style provided by the data custodian will be applied.
- SRS: identifier of the Spatial Reference System.
- BBOX: coordinates of the bounding box following minx,miny,maxx,maxy.
- FORMAT: the format of the returned information (text/xml, text/html, text/plain)
- X,Y: coordinates of the clicked point on the map (in pixels). The origin is at the upper left corner.
- QUERY_LAYERS: list of selected layers to query separated by a comma.

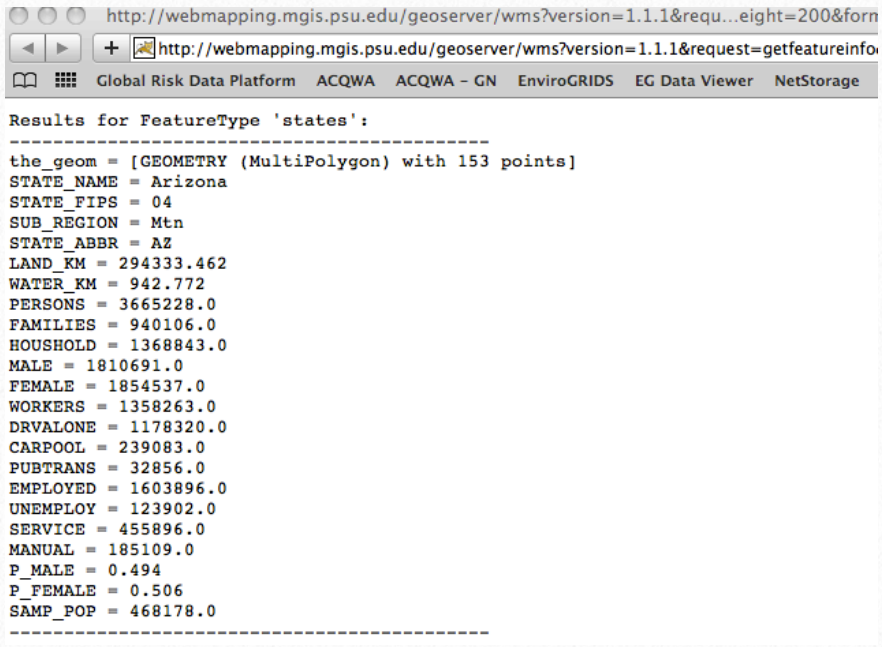


Fig.12: Result of a GetFeatureInfo query.

Web Feature Service (WFS)

OGC Web Feature Service specification: http://www.opengeospatial.org/standards/wfs

The Web Feature Service defines an interface that allows a client to retrieve and update features of georeferenced data encoded in Geography Markup Language (GML). The main difference between WMS and WFS is that WFS gives direct access to the geometry and the attributes of a selected geospatial data, meaning that a user can work with a dataset provided by WFS. In brief, the WFS is the specification to access vector datasets.

Similar to the WMS, a WFS interface is invoked by a URL and can perform a certain number of operations allowing a client to manipulate the data. Following the type of operations needed to manipulate the data, we can define two classes of WFS services:

- Basic WFS: a client can retrieve and/or query features,

- Transactional WFS: a client can create, delete or update a feature.

A transaction is defined as one or more data manipulation operations that form a logical unit.

The concept of a geographic feature is described in the OGC Abstract Specification (OGC, 2009) and the retrieved or created data are encoded in the Geographic Markup Language (OGC, 2007).

- Example of a basic WFS URL:

http://preview.grid.unep.ch:8080/geoserver/wfs?bbox=-84.95293,19.82194,-74.13126,23.19403&styles=&request=GetFeature&version=1.0.0&typename=preview:cy_buffers&srs=EPSG:4326

Like the WMS, WFS service is supported by a set of defined operations:

- *GetCapabilities:* answer to a client describing its capabilities. It tells the client which kind of features are available and what operations are supported on each feature.
- *DescribeFeatureType:* describe the structure of a selected feature (point, line, polygon).
- *GetFeature:* retrieve a selected feature encoded in GML. The client can constrain the query both spatially and non-spatially and also specify the feature properties to fetch.
- *Transaction:* this type of request is made of operations that allow a client to modify features: create, delete and/or update operations. In addition, a client can invoke the LockFeature, in order to be sure that only one user is updating a specific feature, avoiding the risk of multi-edition at the same time.

Web Coverage Service (WCS)

OGC Web Coverage Service specification: http://www.opengeospatial.org/standards/wcs

Like the WFS allows a client to access vector datasets, the Web Coverage Service allow a client to access raster datasets. By rasters we mean data that are represented as a matrix of cells in continuous space organized in rows and columns where each cells contains a value. Thus WCS service provide access to different types of gridded data such as Digial Elevation Model (DEM), remote sensing imagery, etc... It must be noted that WCS gives only access to the raw data and does not have transactional capabilities.

Like all the OGC Web Services, a WCS interface consists of different operations:

- *GetCapabilities:* answer to a client describing its capabilities. It tells the client which kind of raster data (or coverage) are available.
- *DescribeCoverage:* describe the structure of a selected coverage.
- *GetCoverage:* retrieve the selected coverage.

- Example of a WCS URL with a GetCapabilities request:

http://preview.grid.unep.ch:8080/geoserver/ows?service=WCS&request=GetCapabilities

The GetCapabilities operation returns to a client an XML document describing the service and the data sets available from which either desktop and/or web clients may request coverages.

To invoke the operation, the user has only to define the service and the request parameters.

- Example of a WCS URL with a DescribeCoverage request:

http://preview.grid.unep.ch:8080/geoserver/wcs?service=WCS&request=DescribeCoverage&version=1.0.0&coverage=preview:cy_frequency

The DescribeCoverage operation returns to a client an XML document describing selected coverages. The information provided must be sufficient for a client to assess the fitness for use of the data. It gives different useful pieces of information such as the supported raster formats, supported SRS, supported interpolation methods, etc...

Mandatory parameters for the DescribeCoverage request:

- SERVICE: value “WCS”, this is the name of the invoked service.
- REQUEST: value “DescribeCoverage”, this is the request to invoke the specific operation.
- VERSION: the version of the specification.
- COVERAGE: list of selected coverages separated by a comma.

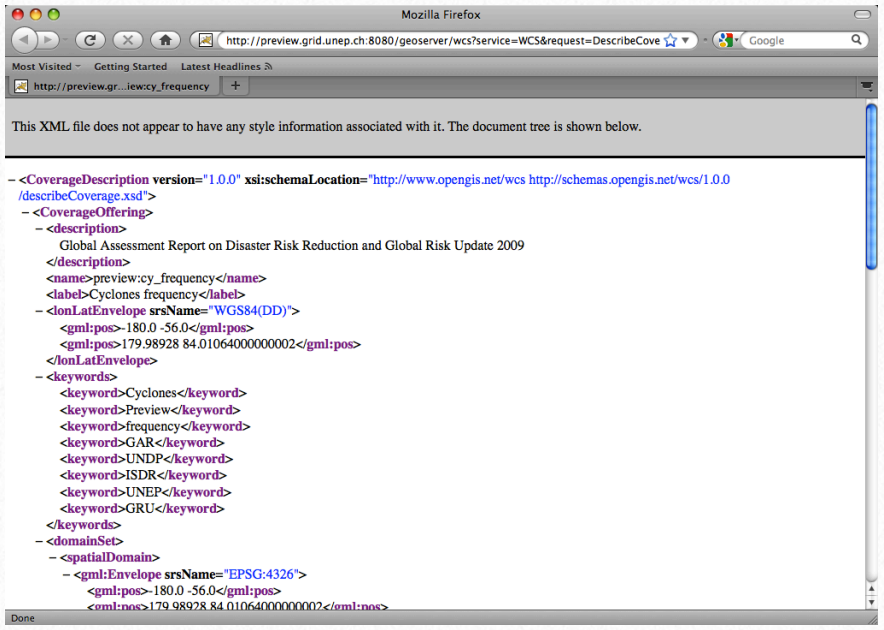


Fig.13: WCS DescribeCoverage

- Example of a WCS URL with a GetCoverage request:

http://preview.grid.unep.ch:8080/geoserver/wcs?bbox=-84.95293,19.82194,-74.13126,23.19403&service=WCS&styles=&request=GetCoverage&version=1.0.0&coverage=preview:cy_frequency&width=640&height=309&crs=EPSG:4326 &Format=GeoTiff

The GetCoverage request returns to a client the requested raster data. The syntax and the parameters of the URL are similar to those used in a WMS GetMap resquest.

Mandatory parameters for the GetCoverage request:

- BBOX: coordinates of the bounding box following minx,miny,maxx,maxy
- SERVICE: value “WCS”, this the name of the invoked service.
- STYLES: list of style names separated by a comma. It is necessary to have an exact correspondence between the number of layers and the number of styles. If this parameter has an

empty value, the default style provided by the data custodian will be applied.
- REQUEST: value “GetCoverage”, this the request to invoke the specific operation.
- VERSION: the version of the specification.
- COVERAGE: name of a single selected coverage.
- WIDTH: specify the width of the returned coverage (in pixels).
- HEIGHT: specify the height of the returned coverage (in pixels).
- CRS: identifier of the Coordinate Reference System.
- FORMAT: the desired format to be used for returning the coverage (eg: GeoTiff, ARCGRID, GTOPO30,...)

If a request is validated then a coverage is extracted (using the BBOX, FORMAT and the different parameters set in the URL) from the selected coverage and sent to the client. If the client is a web browser then the user can download the coverage file. If the request is sent through a Desktop GIS client like ArcGIS then the user gets the coverage directly into it.

Web Processing Service (WPS)

OGC Web Processing Service specification: http://www.opengeospatial.org/standards/wps

The two previous discussed standards are focusing on data accessibility: WFS allows a client to access vector data while WCS allows a client to retrieve raster data.

Now we need to extend our capabilities in order to process data available using the recently introduced Web Processing Service (OGC, 2007) that provides access to processing and calculations on geospatial data. A WPS service can offer, through a network access, a vast variety of GIS functionalities ranging from a simple calculation to complex models. It acts as a sort of middleware between the client and the process that runs the calculations. It allows users to know which processes are available, to select the required input data and their formats, to create a model and run it, to manage processes (status, storage for the output, …) and to return the output once the computation is completed.

Like the others OWS, the WPS specification includes a set of operations:

- *GetCapabilities:* answer to a client describing its capabilities. It tells the client which kind of process are available.
- *DescribeProcess:* describe the parameters a selected process.
- *Execute:* execute a selected process.

The WPS differs a bit from the others OWS because these operations can be invoked either by SOAP or the traditional http-get and http-post.

- Example of a WPS URL with a GetCapabilites request:

http://localhost/wps/wps.py?service=WPS&request=GetCapabilities

The GetCapabilities operation returns to a client an XML document describing the service and the processes available for execution.

To invoke the operation, the user has only to define the service and the request parameters.

- Example of a WPS URL with a DescribeProcess request:

http://localhost/wps/wps.py?service=WPS&request=DescribeProcess&version=1.0.0&identifier=soil_process

The DescribeProcess operation returns an XML document describing what are the mandatory, optional and default parameters needed for a selected process, as well as the data formats for inputs and outputs.

Mandatory parameters for this operation:

- SERVICE: value “WPS”, this is the name of the invoked service.
- REQUEST: value “DescribeProcess”, this is the request to invoke the specific operation.
- VERSION: the version of the specification.
- IDENTIFIER: the name of the selected process to run.

- Example of a WPS URL with an Execute request:

http://localhost/wps/wps.py?version=1.0.0&service=WPS&request=Execute&identifier=soil_process&datainputs=http://localhost/wps/soil_param.xml

The Execute operation allows a client to run a selected process using values entered by the client for the required parameters (if needed) and reference the datasets location. Once the process is completed, the result is returned to the client as a new dataset.

Sensor Observation Service (SOS)

OGC Sensor Observation Service specification: http://www.opengeospatial.org/standards/sos

The OGC Sensor Web Service provides an interface for managing sensors and retrieving data from them. This standard is part of the suite of standards called Sensor Web Enablement (SWE) that are presented in details in the deliverable D2.3.

ISO 19115/19139

ISO 19115, Geographic information – Metadata: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=26020

ISO 19139, Geographic information – Metdata-XML schema implementation: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=32557

The ISO 19115 standard defines (with more than 400 metadata elements, 20 core mandatory elements) how to describe a georeferenced information and to provide information about the content, the identification, the quality, the spatial and temporal extent, the access and rights and the spatial reference.

This standard is applicable to:

- The cataloguing of datasets, clearinghouse activities, and the full description of datasets,
- Geographic datasets, dataset series, and individual geographic features and features properties.

The main applicability of ISO19115 is for digital data but its principles can be extended and applied to other forms of geospatial data such as maps, charts and textual documents as well as non-geographic data (ISO, 2003).

The ISO 19139 standard complements ISO19115 by defining an XML encoding schema implementation specifying the metadata record format and may be used to describe, validate, and exchange geospatial metadata prepared in XML.

ISO 19119

ISO 19119, Geographic information – Services:

http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=39890

The ISO 19119 identifies and defines the architecture patterns for service interfaces used for geographic information, defines its relationship to the Open Systems Environment model, presents a geographic services taxonomy and a list of example geographic services placed in the services taxonomy. It also prescribes how to create a platform-neutral service specification, how to derive conform platform-specific service specifications, and provides guidelines for the selection and specification of geographic services from both platform-neutral and platform-specific perspectives. In other words, ISO 19119 specifies the form and content of the XML document that describes the capabilities of a geospatial web service (eg, the answer to the GetCapabilities request).

Keyhole Markup Language (KML)

OGC Keyhole Markup Language specification: http://www.opengeospatial.org/standards/kml

The KML format is an XML based language schema for describing geographical objects in web-based viewer (the so-called geobrowser). It has been developed and popularized by the Google Earth application and due to its success was turned into an OGC standard.

Geographic Markup Language (GML)

OGC Geographic Markup Language specification: http://www.opengeospatial.org/standards/kml

The GML is a very complete XML based language used to describe all geographical features and objects and provides a standard mean for representing geographical features (properties, relationships, geometries, …). GML differs from KML as it is not only used for data visualization (which is the main focus of the KML specification) but serves also as a modeling language as well as an open and interoperable exchange format over the Internet. It is mostly used in the Web Feature Service to send geographical features between servers and clients.

The GML core schema does not contain definitions of features (because it is impossible to describe all features) but could be rather seen as a grammar providing means to define concrete features through GML Application Schemas that are created by users. Although GML is easily readable, it will be mostly used and generated by software or web services answering a specific request and then receiving the result as a GML dataset.

The real advantage of using GML is that XML technologies are available meaning that information stored in a GML file could be easily shared with other information and then specialized application domains could reuse, extend and/or refine GML components in an application schema in order to develop a specific data model.

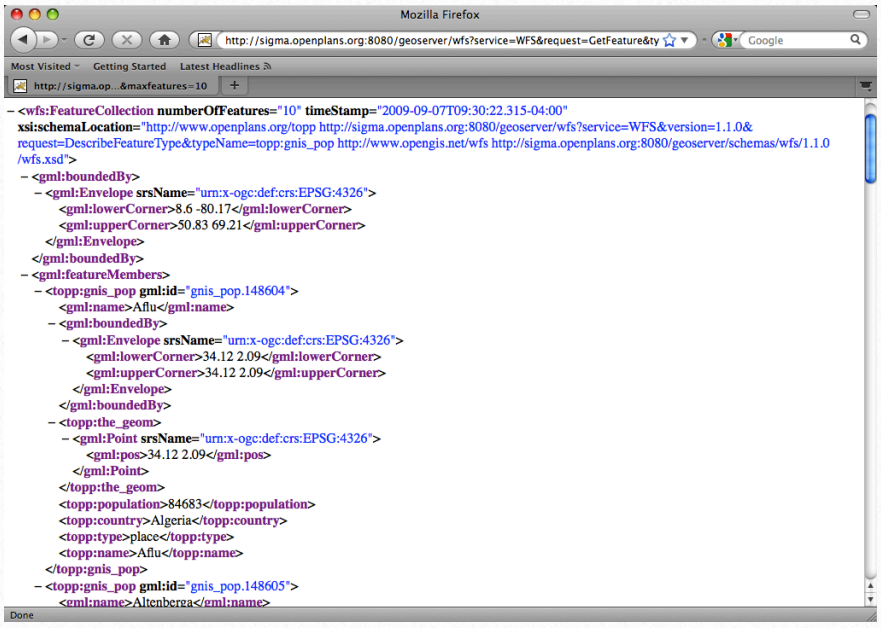


Fig.14: Example of GML returned after a WFS request.

More information available at:
http://www.w3.org/Mobile/posdep/GMLIntroduction.html

Acronyms

BMP: BitMaP
BSD: Berkeley Software Distribution
CoP: Community of Practice
CRS: Coordinate Reference System
CSR: Components and Services Registry
CSW: Catalog Service for the Web
DAB: Discovery and Access Broker
DD: Decimal Degree
ECW: ERMapper Compress Wavelets
EGEE: Enabling Grids for E-scienceE
EO: Earth Observation
EPSG: European Petroleum Survey Group
GEO: Group on Earth Observations
GEOSS: Global Earth Observation System of Systems
GBIF: Global Biodiversity Information Facility
GCI: GEOSS Common Infrastructure
GDAL: Geospatial Data Abstraction Library
GIF: Graphics Interchange Format
GIS: Geographic Information System
GMES: Global Monitoring for Environment and Security
GML: Geographic Markup Language
GPL: General Public License
GPS: Global Positioning System
GRASS: Geographic Resources Analysis Support System
GRID: Global Resource Information Database
GSDI: Global Spatial Data Infrastructure
GUI: Graphical User Interface
IDW: Inverse Distance Weighting
INSPIRE: Infrastructure for Spatial Information in the European Community
ISO: International Organization for Standardization
IR: Implementing Rules
IT: Information Technology
JPEG: Joint Photographic Experts Group
JSON: JavaScript Object Notation
KML: Keyhole Markup Language
NCEP: National Centers for Environmental Predictions
OAI: Open Archive Initiative
OASIS: Organization for the Advancement of Structured Information Standards
OGC: Open Geospatial Consortium
OWS: OGC Web Services
PDF: Portable Document Format
PNG: Portable Network Graphics
PyWPS: Python Web Processing Service
QGIS: Quantum GIS
RDBMS: Relational DataBase Management System
REST: Representational State Transfer
RPC: Remote Procedure Call
RSS: Really Simple Syndication
SDI: Spatial Data Infrastructure
SDK: Software Development Kit
SHP: Shape File
SLD: Styled Layer Descriptor

SOA: Service Oriented Architecture
SOAP: Simple Object Access Protocol
SOS: Sensor Observation Service
SQL: Structured Query Language
SRS: Spatial Reference System
SVG: Scalable Vector Graphics
SWE: Sensor Web Enablement
THREDDS: Thematic Realtime Environmental Distributed Dat Services
TIFF: Tagged Image File Format
UN: United Nations
UNEP: United Nations Environment Program
UNECA: United Nations Economic Commission for Africa
UNGIWG: United Nations Geographical Information Working Group
URM: Uniform Resource Management
VGI: Volunteered Geographic Information
W3C: World Wide Web Consortium
WAS: Web Service Authentication
WCS: Web Coverage Service
WFS: Web Feature Service
WGS 84: World Geodetic System 1984
WIS: WMO Information System
WMO: World Meteorological Organization
WMS: Web Map Service
WSDL: Web Service Description Language
WSS: Web Security Service
WPS: Web Processing Service
WPVS: Web Perspective and View Service
WTS: Web Terrain Service
XML: eXtended Markup Language

References

Almirall G., Bergada M.M, Ros P.Q., Craglia M. (2007) The Socio-Economic Impact of the Spatial Data Infrastructure of Catalonia, JRC Scientific and Technical Reports, 61p.
http://inspire.jrc.ec.europa.eu/reports/Study_reports/catalonia_impact_study_report.pdf

Arzberger P., Schroeder P., Beaulieu A., Bowker G., Casey K., Laaksonen L., Moorman D., Uhlir P., Wouters P. (2004) Promoting Access to Public Research Data for Scientific, Economic and Social Development, Data Science Journal, Vol.3, pp.135-152

Bejar R., Latre, M.A., Nogueras-Iso J., Muro-Medrano P.R., Zarazaga-Soria F.J. (2009) Systems of Systems as a Conceptual Framework for Spatial Data Infrastructures, International Journal of Spatial Data Infrastructure Research, Vol.4, 15p.
http://ijsdir.jrc.ec.europa.eu/index.php/ijsdir/article/view/124/105

Bernard L., Craglia M. (2005) SDI - From Spatial Data Infrastructure to Service Driven Infrastructure, Workshop on Cross-learning on Spatial Data Infrastructures (SDI) and Information Infrastructures (II), 8p.
http://www.ec-gis.org/sdi/ws/crosslearning/papers/PP%20Lars%20Bernard%20-%20Max%20Craglia.pdf

Boes U., Pavlova R. (2007) The Web 2.0 - A New Challenge For GIS And Cartography, Conference on recent problems in Geodesy and related fields with international importance, Sofia, Bulgaria, 10p.
http://www.rgz.sr.gov.yu/DocF/Files/intergeo-east-2007/n11.pdf

Boes U., Pavlova R. (2008) Is there a Future for Spatial Data Infrastructures?, GI Days 2008: Interoperability and spatial processing in GI applications, 10p.
http://www.gi-tage.de/archive/2008/downloads/acceptedPapers/Papers/Boes,Pavlova.pdf

Booz Allen Hamiltion (2005) Geospatial Interoperability Return on Investment Study, NASA, Geospatial Interoperability Office, 47p.
http://www.egy.org/files/ROI_Study.pdf

Coleman, D.J. and J.D. McLaughlin and S.E. Nichols (1997) Building a Spatial Data Infrastructure, Proceedings of the 64th Permananent Congress Meeting of the Fédération Internationale des Géometres (FIG), pp. 89-104, Singapore, May.

Coleman D.J., Georgiadou Y., Labonte J. (2009) Volunteered Geographic Information: the nature and motivation of produsers, GSDI-11 Conference Proceedings, 20p.
http://www.gsdi.org/gsdi11/papers/pdf/279.pdf

Cömert C. (2004) Web Services and National Data Infrastructure (NSDI), ISPRS Conference, 6p.
http://cartesia.org/geodoc/isprs2004/comm4/papers/365.pdf

Craglia M., Novak J. (2006) Assessing the impacts of Spatial Data Infrastructures, Report of International Workshop on Spatial Data Infrastructures' Cost-Benefit / Return on Investment, Ispra, Italy, 12-13 January, 57p.

Craglia M. (2007) Volunteered Geographic Information and Spatial Data Infrastructures: when do parallel lines converge?, VGI Specialist Meeting, Santa Barbara, USA, 3p.
http://www.ncgia.ucsb.edu/projects/vgi/docs/position/Craglia_paper.pdf

Craig W.J. (2005) White Knights of Spatial Data Infrastructure: The Role and Motivation of Key Individuals, URISA Journal, Vol.16, pp.5-13

Crompvoets J., Bregt A. (2003) World Status of National Spatial Data Clearinghouses, URISA Journal, Vol.15, pp.43-50

De Longueville B., Ostländer N, Keskitalo C. (2009) Addressing vagueness in Volunteered Geographic Information (VGI) – A case study, GSDI-11 Conference Proceedings, 21p.
http://www.gsdi.org/gsdi11/papers/pdf/221.pdf

European Commission (2007) Directive 2007/2/EC of the European Parliament and the Council of 14 March 2007 establishing an Infrastructure for Spatial Information in the European Community (INSPIRE), 14p
http://eur-lex.europa.eu/JOHtml.do?uri=OJ:L:2007:108:SOM:EN:HTML

European Commission (2008) Commission regulation (EC) No 1205/2008 of 3 December 2008 implementing Directive 2007/2/EC of the European Parliament and of the Council as regard metadata, 19p.
http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2008:326:0012:0030:EN:PDF

European Commission (2008) INSPIRE Generic Conceptual Model, 109p.
http://inspire.jrc.ec.europa.eu/reports/ImplementingRules/DataSpecifications/D2.5_v3.1.pdf

European Commission (2008) INSPIRE Network Services Architecture, 29p.
http://inspire.jrc.ec.europa.eu/reports/ImplementingRules/network/D3_5_INSPIRE_NS_Architecture_v3-0.pdf

European Commission (2007) INSPIRE Technical Architecture - Overview, 12p.
http://inspire.jrc.ec.europa.eu/reports/ImplementingRules/network/INSPIRETechnicalArchitectureOverview_v1.2.pdf

GEO (2005) Global Earth Observation System of Systems 10-Year Implementation Plan Reference Document, 209p
http://www.earthobservations.org/documents/10-Year%20Plan%20Reference%20Document.pdf

GEO (2005) The Global Earth Observation System of Systems (GEOSS) 10-Year Implementation Plan, 11p
http://www.earthobservations.org/documents/10-Year%20Implementation%20Plan.pdf

GEO (2006) GEO Capacity Building Strategy, 13p.
http://www.earthobservations.org/documents/committees/cbc/capacity_building_strategy.pdf

GEO (2007) The Full Picture, 143p.
http://www.earthobservations.org/documents/the_full_picture.pdf

GEO (2008) White Paper on the GEOSS Data Sharing Principles, 93p.
http://www.earthobservations.org/documents/dsp/Draft%20White%20Paper%20for%20GEOSS%20Data%20Sharing%20Policies_27Sept08.pdf

Goodchild M.F. (2007) Citizens as sensors: The World of Volunteered Geography, Workshop on Volunteered Geography Information, Santa Barbara, USA, 15p.
http://www.ncgia.ucsb.edu/projects/vgi/docs/position/Goodchild_VGI2007.pdf

Gore A. (1998) The Digital Earth: Understanding our planet in the 21st Century, 5th international Symposium on Digital Earth, 5p.
http://www.isde5.org/al_gore_speech.htm

Lehmann A. (2009) EnviroGRIDS, Document of Work, 131p.

Liping D., Chen A., Yang W., Zhao P. (2003) The Integration of Grid Technology with the OGC Web Services (OWS) in NWGISS for NASA EOS Data, , 10p.

Masser I. (2006) What's Special about SDI Related Research, International Journal of Spatial Data Infrastructures Research, Vol.1, pp.14-23

Masser I. (2005) The Future of Spatial Data Infrastructures, ISPRS Workshop on Service and Application of Spatial Data Infrastructure, Oct.14-16, Hangzhou, China, 9p.
http://www.commission4.isprs.org/workshop_hangzhou/papers/7-16%20Ian%20Masser-A001.pdf

Masser I. (2007), Building European Spatia Data Infrastructures, ESRI Press, 91p.

Mohammadi H., Rajabifard A. Williamson I.P. (2007) Multisource Spatial Data Integration within the Context of SDI Initiatives, 18p, Internation Journal of Spatial Data Infrastructures Research, 18p.
http://ijsdir.jrc.ec.europa.eu/index.php/ijsdir/article/view/74/34

Muresan O., Pop F., Gorgan D., Cristea V. (2006) Satellite Image Processing Applications in MedioGRID, Satellite image processing application in mediogrid, pp.253-262

Open Geospatial Consortium (2005) The Importance of Going “Open”, OGC White Paper, 8p.
http://portal.opengeospatial.org/files/?artifact_id=6211&version=2&format=pdf

Open Geospatial Consortium (2004) The Havoc of Non-Interoperability, OGC White Paper, 7p.
http://portal.opengeospatial.org/files/?artifact_id=5097&version=3&format=pdf

Open Geospatial Consortium (2004) Integrating Geospatial Standards and Standards Strategies Into Business Process, OGC White Paper, 7p.
http://portal.opengeospatial.org/files/?artifact_id=5098&version=2&format=pdf

Open Geospatial Consortium (2004) The Spatial Web, OGC White Paper, 8p.
http://portal.opengeospatial.org/files/?artifact_id=3859&version=2&format=pdf

Open Geospatial Consortium (2007) OpenGIS Web Service Common Implementation Specification, 153p.
http://www.opengeospatial.org/standards/common

Open Geospatial Consortium (2007) OpenGIS Geography Markup Language (GML) Encoding
Standard, 437p.
http://www.opengeospatial.org/standards/gml

Open Geospatial Consortium (2009) The OpenGIS Abstract Specification Topic 5: Features, 50p.
http://www.opengeospatial.org/standards/as

Open Geospatial Consortium (2008) OpenGIS Reference Model, 35p.
http://www.opengeospatial.org/standards/orm

Open Geospatial Consortium (2004) Geospatial Portal Reference Architecture, 23p.
http://portal.opengeospatial.org/files/?artifact_id=6669

Ostensen O. (2001) Global geomatics standards supporting sustainable geospatial data infrastructures, GSDI 5 Conference Proceedings, 6p.
http://gsdidocs.org/gsdiconf/GSDI-5/papers/Olaf%20Ostensen-paper.pdf

Padberg A., Greve K. (2009) Gridification of the OGC Web Processing Service: Challenges and Portential, AGILE Workshop, 5p.

Paluszynski W., Iwaniak A., Sliwinski A., Kubik T. (2007) Building Spatial Data Infrastructure At County Level Using Open Software, SDI-East Africa, 11p.
http://dewa03.unep.org/sdi-ea/system/files/BUILDING_SPATIAL_DATA_INFRASTRUCTURE_AT_C.doc

Phillips A., Williamson I., Ezigbalike C. (1999) Spatial Data Infrastructure Concepts, The Australian Surveyor, Vol.44 No1, p20-28
http://www.sli.unimelb.edu.au/research/publications/IPW/SDIDefinitionsAusSurv.html

Rajabifard A. and Williamson I.P. (2001) Spatial Data Infrastructures: Concept, SDI Hierarchy and Future directions, in Proceedings, of GEOMATICS'80 Conference, Tehran, Iran., 10p.
http://repository.unimelb.edu.au/10187/1247

Rajabifard A. (2002) Diffusion of Regional Spatial Data Infrastructures: with particular reference to Asia and the Pacific, Ph.D thesis, Department of Geomatics, Faculty of Engineering, The University of Melbourne, 213p.

Rajabifard A. and Williamson I.P. (2004) SDI Development and Capacity Building, GSDI-7 Conference on Spatial Data Infrastrcture for a Sustainable Futures, 12p.
http://www.geoinfo.ait.ac.th/download/SCOSA2007/1_DrAbbas/4-SDIDevelopmentCapacityBuilding.pdf

Ryttersgaard J. (2001) Spatial Data Infrastructure, Developing Trends and Challenges, International Conference on Spatial Information for Sustainable Development Proceedings, Nairobi, 8p.
http://www.fig.net/pub/proceedings/nairobi/ryttersgaard-TS1-1.pdf

Sahin K., Gumusay M.U. (2008) Service Oriented Architecture (SOA) based Web Services for Geographic Information Systems, ISPRES Congress 2008, pp.625-630
http://www.isprs.org/congresses/beijing2008/proceedings/2_pdf/5_WG-II-5/03.pdf

Sayar A., Pierce M., Fox G. (2005) OGC Compatible Geographical Information Systems Web Services, Indiana University , Departement of Computer Science, Technical Report, 64p.
http://www.cs.indiana.edu/pub/techreports/TR610.pdf

Stevens A.R, Onsrud J.H, Rao M.(2005) Global Spatial Data Infrastructure (GSDI): Encouraging SDI Development Internationally, ISPRS Conference on Service and Application of Spatial Data Infrastructure, 14-16 October, Hangzhou, China, 10p.
http://www.commission4.isprs.org/workshop_hangzhou/papers/309-312%20Abs&Papr%20GSDI%20China%2014Oct05.pdf

UNECA (2005) SDI Africa: An Implementation Guide, 120p.
http://geoinfo.uneca.org/sdiafrica/default1.htm

UNGIWG (2007) UNSDI Compendium - A UNSDI Vision, Implementation Strategy and Reference Architecture, 150p.
http://www.ungiwg.org/docs/unsdi/UNSDI_Compendium_13_02_2007.pdf

Vaccari L., Shvaiko P., Marchese M. (2008) An emergent semantics approach to semantic intergration of geo-services and geo-metadata in Spatial Data Infrastructures, IJSDIR, 23p.
http://ijsdir.jrc.ec.europa.eu/index.php/ijsdir/article/view/107/113

Williamson I.P, Rajabifard A., Enemark S. (2003) Capacity Building for SDIs, Proceedings 16th UN Regional Cartographic Conference., 14p.
http://repository.unimelb.edu.au/10187/1171

Williamson I., Rajabifard A., Binns A. (2006) Challenged and Issues for SDI Development, International Journal of Spatial Data Infrastructures Research, Vol.1 pp.24-35

Wilson M., Von Hagen C., Howard C. (2009) SDI in East Africa - Leveraging the UN presence, International Journal of Spatial Data Infrastructure Research, Vol.4, pp.1-23
http://ijsdir.jrc.ec.europa.eu/index.php/ijsdir/article/view/109/112

Acknowledgements

The authors duly thank the following people for revising this material: Pierric de Laborie (UNEP-DEWA-GRID/Geneva), Mick Wilson (UNEP-DEWA/Nairobi), Laura Lopez (UNEP-DEWA-GRID/Geneva).

Copyright

This material has been developed by researchers from University of Geneva (http://www.unige.ch/envirospace), Laboratory of Earth and Space Science Informatics of the Institute of Atmospheric Pollution Research (IIA) of the National Research Council of Italy (CNR) (http://www.essi-lab.eu) and United Nations Environment Programme GRID-Geneva (http://www.grid.unep.ch).

This material is largely based on GeoNetwork, OpenGeo, PostGIS, PostgreSQL, Python, PyWPS, QGIS documentation and has been adapted by the authors.

This material has been supported by EU FP7 EnviroGRIDS, IASON & EOPOWER projects.