Geophysical Research Abstracts,
Vol. 11, EGU2009-10298-2, 2009
EGU General Assembly 2009
© Author(s) 2009

Enabling the development of Community Extensions to GI-cat - the SIB-ESS-C case study

L. Bigagli (1,2), N. Meier (3), E. Boldrini (1), and R. Gerlach (3)

(1) Institute of Methodologies for Environmental Analysis of the Italian National Research Council (IMAA-CNR), Contrada Santa Loja - Zona Industriale, 85050, Tito Scalo (PZ), Italy ({bigagli, boldrini}@imaa.cnr.it), (2) University of Florence, Piazza Ciardi 25, 59100, Prato, Italy (lorenzo.bigagli@pin.unifi.it), (3) Friedrich-Schiller-University, Institute for Geography, Earth Observation, Grietgasse 6, 07743 Jena, Germany ({n.meier, roman.gerlach}@uni-jena.de)

GI-cat is a Java software package that implements discovery and access services for disparate geospatial resources. An instance of GI-cat provides a single point of service for querying and accessing remote, as well as local, heterogeneous sources of geospatial information, either through standard interfaces, or taking advantage of GI-cat advanced features, such as incremental responses, query feedback, etc.
GI-cat supports a number of de-iure and de-facto standards, but can also be extended to additional community catalog/inventory services, by defining appropriate mediation components.
The GI-cat and the SIB-ESS-C development teams collaborated in the development of a mediator to the Siberian Earth Science System Cluster (SIB-ESS-C), a web-based infrastructure to support the communities of environmental and Earth System research in Siberia.
This activity resulted in the identification of appropriate technologies and internal mechanisms supporting the development of GI-cat extensions, that are the object of this work.
GI-cat is actually built up of a modular framework of SOA components, that can be variously arranged to fit the needs of a community of users. For example, a particular GI-cat instance may be configured to provide discovery functionalities onto an OGC WMS; or to adapt a THREDDS catalog to the standard OGC CSW interface; or to merge a number of CDI repositories into a single, more efficient catalog.
The flexibility of GI-cat framework is achieved thanks to its design, that follows the Tree of Responsibility (ToR) pattern and the Uniform Pipe and Filter architectural style.
This approach allows the building of software blocks that can be [U+FB02]exibly reused and composed in multiple ways. In fact, the components that make up any GI-cat configuration all implement two common interfaces (i.e. IChainNode and ICatalogService), that support chaining one component to another . Hence, it would suffice to implement those interfaces (plus an appropriate factory class: the mechanism used to create GI-cat components) to support a custom community catalog/inventory service in GI-cat.
In general, all the terminal nodes of a GI-cat configuration chain are in charge of mediating between the GI-cat common interfaces and a backend, so we implemented a default behavior in an abstract class, termed Accessor, to be more easily subclassed.
Moreover, we identified several typical backend scenarios and provided specialized Accessor subclasses, even simpler to implement. For example, in case of a coarse-grained backend service, that responds its data all at once, a specialized Accessor can retrieve the whole content the first time, and subsequently browse/query the local copy of the data.
This was the approach followed for the development of SibesscAccessor.
The SIB-ESS-C case study is also noticeable because it requires mediating between the relational and the semi-structured data models. In fact, SIB-ESS-C data are stored in a relational database, to provide performant access even to huge amounts of data. The SibesscAccessor is in charge of establishing a JDBC connection to the database, reading the data by means of SQL statements, creating Java objects according to the ISO 19115 data model, and marshalling the resulting information to an XML document.
During the implementation of the SibesscAccessor, the mix of technologies and deployment environments and the geographical distribution of the development teams turned out to be important issues. To solve them, we relied on technologies and tools for collaborative software development: the Maven build system, the SVN version control

system, the XPlanner project planning and tracking tool, and of course VOIP tools.
Moreover, we shipped the Accessor Development Kit (ADK) Java library, containing the classes needed for extending GI-cat to custom community catalog/inventory services and other supporting material (documentation, best-practices, examples).
The ADK is distributed as a Maven artifact, to simplify dependency management and ease the common tasks of testing, packaging, etc.
The SibesscAccessor was the first custom addition to the set of GI-cat accessors. Later, also the so-called Standard Accessors library has been refactored onto the ADK.
The SIB-ESS-C case study also gave us the opportunity to refine our policies for collaborative software development. Besides, several improvements were made to the overall GI-cat data model and framework.
Finally, the SIB-ESS-C development team developed a GI-cat web client by means of Web 2.0 technologies (JavaScript, XML, HTML, CSS, etc.)
The client can easily be integrated in any HTML context on any web page.
The web GUI allows the user to define requests to GI-cat by entering parameter strings and/or selecting an area of interest on a map. The client sends its request to GI-cat via SOAP through HTTP-POST, parses GI-cat SOAP responses and presents user-friendly information on a web page.