Improving open source GIS-SDI integration: the web service publishing extension for gvSIG

José Vicente Higón[1] and Salvador Bayarri[2]

[1]IVER Technologies, Valencia, Spain josevicente.higon@iver.es
[2]IVER Technologies, Valencia, Spain salvador.bayarri@iver.es

Abstract

The publishing extension for gvSIG, an open source desktop GIS currently in OSGeo incubation, aims at easing the task of publishing OGC/ISO standard web geoservices through different open source technologies. Together with the metadata editing and publishing extension, the overarching goal is to help users and SDI developers to better integrate their workflow from desktop GIS to server technologies, bridging the current gap between OSS geospatial packages, and thus promoting widespread use of these technologies.

1. Introduction

In the open source domain, desktop GIS tools have usually been clients –consumers– of standard web services (Web Map Services –WMS–, Web Feature Services –WFS-, Web Coverage Services –WCS-, Catalog Services for the Web –CSW– and others) which are provided by commercial or open source server-side software like MapServer, GeoServer o GeoNetwork. Each OSS server-side technology has developed its own mechanism for configuring these services, typically by using text files or simple web pages (e.g. see Mapfile), but these mechanisms lack the interactive capabilities of a GIS-like graphical user interface to combine layers and define the filters and symbology applying to them. Users feel that the work they have already performed in their desktop GIS while creating workspaces, projects and maps, should be reused instead of duplicated when publishing web services.

It is probably for the reasons mentioned above that publishing standard geospatial web services is still considered by many users as an off-line task for specialists rather than something they can do themselves. Currently, this difficulty is also a specific disadvantage of an open-source GIS/SDI software stack compared to a single-provider commercial solution. At this moment, when the INSPIRE EU Directive (INSPIRE 2007) mandates the publication of OGC-compliant services at all levels of public government in Europe, and SDI technology is spreading elsewhere, it seems very important to streamline the process of service publishing.

The immediate need to develop the publishing extension arose in the Regional Council of Infrastructure and Transport (CIT) of the Valencia Region in Spain, the governmental agency which also funds most of gvSIG's development (gvSIG, 2004), and had the need to reuse existing GIS document to set up an INSPIRE-compliant server. The gvSIG publishing extension was thus born as a way to ease the process of producing geospatial services equivalent to the data and maps managed by GIS users at the CIT. A related, but separate, project developing a gvSIG extension for metadata creation and publishing (through Geonetwork opensource) is carried out by the University Jaume I of Castellón (Díaz, 2008).

2. Architecture

The architecture of the publishing extension is designed following the plugin-based extensibility model which pervades gvSIG (gvSIG 2007). Common core components provide access to the GIS objects and a GUI framework, while extensible components implement specific exporting functionality for different platforms (e.g. MapServer) and services (e.g. WMS).

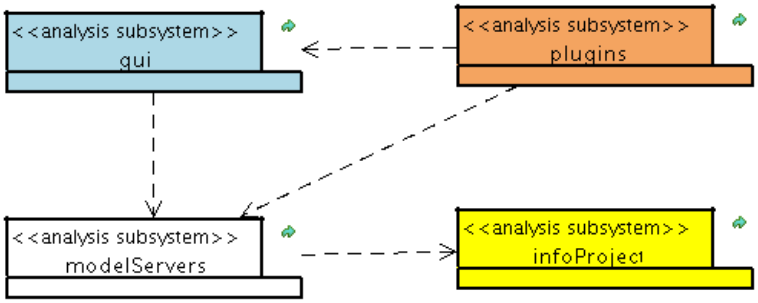

Figure 1. Top-level components of the publishing extension for gvSIG

Starting at the bottom of the dependency graph, the main components of the extension are (see Figure 1):

- infoProject: this API provides convenient access to the resources to be published (layers, views) and their properties (legends, spatial info, data sources) as they are defined in the core of gvSIG. The access to this objects is required to generate the publishing configuration.
- modelServers: these libraries define the essential functionality of the extension: the API and base objects which must be implemented for each target service and server technology.
- gui: provides a graphical interface framework for users to enter parameters and to control the publication process.
- plugins: these are deployable packages providing implementation of gvSIG's extension interfaces, using the above three components to build specific publishing modules for a combination of server and service(s). For instance, in the first release of the publishing extension, two plugins are included; one contains the GUI and library elements to publish WMS, WFS and WCS services via MapServer, while the other plugin allows WFS publishing via GeoServer.

Both libraries (infoProject and modelServers) can be used outside the context of the publishing extension, providing a mechanism for other extensions or applications to develop their own high-level publishing mechanisms.

2.1. The modelServers library

The modelServers library proposes a simple object model (see Figure 2): each Publication task is linked to a specific Server class implementation (e.g. GeoServer) and one implementation of the Service class (e.g. WFS). The services can, in turn, use other RemoteResources as their source (e.g.

a published WMS might use several WMS services as sources). The implementations for these classes must be registered (by the plugin providing them) to be available in the user interface,

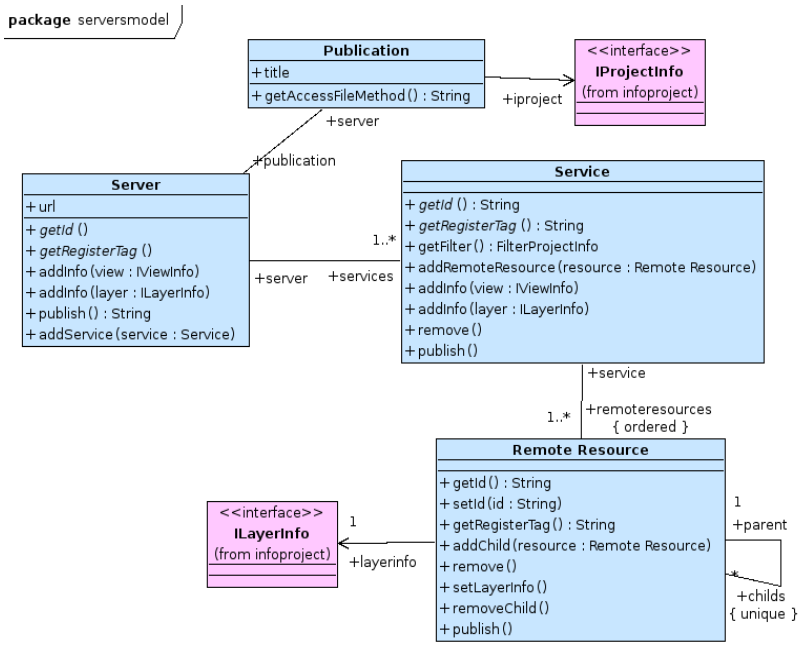

Figure 2. Object model for the modelServers library

2.2. The GUI object model

The GUI framework (see Figure 3) provides some common elements shared by all plugins; a MainWindow, and a MainController which drives the publication process. In addition, it defines the programming interfaces that each plugin must implement in order to allow the input of specific publication parameters and options (IPublishView) and the connection of these specific options to the overall publishing process led by the MainController, via the IPublishController implementations.

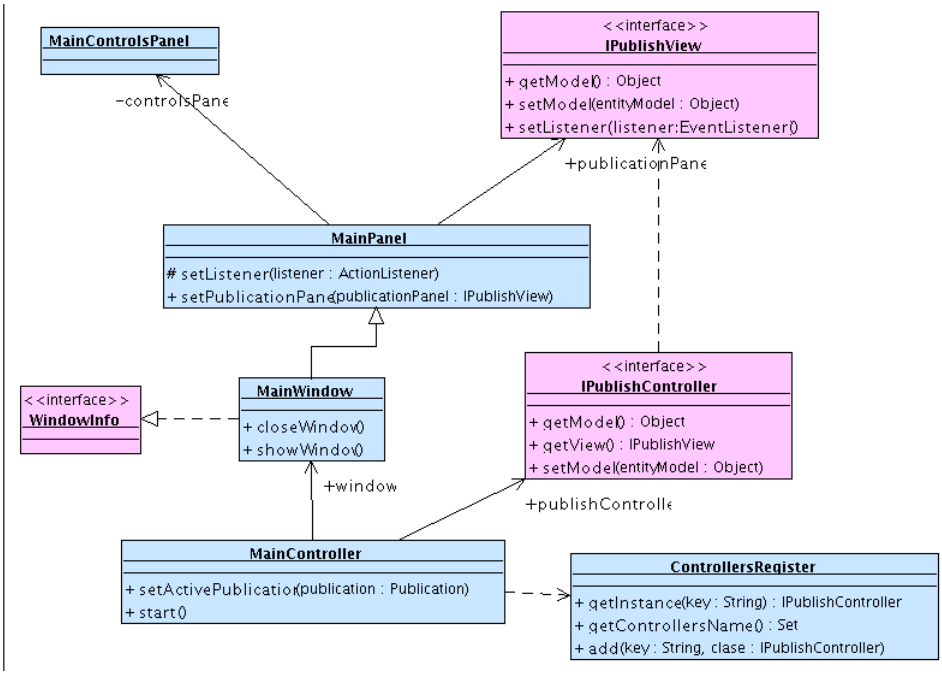

Figure 3: object model for the GUI library

2.3. The contents of a plugin

The role of a plugin is to provide the implementation of some components specific to a certain server technology, and registers those components to be recognized by the GUI framework, so they can be accessed by the user. Figure 4 shows an example the components that the MapServer plugin will provide to the GUI component (the ones whose names start with MapServer).

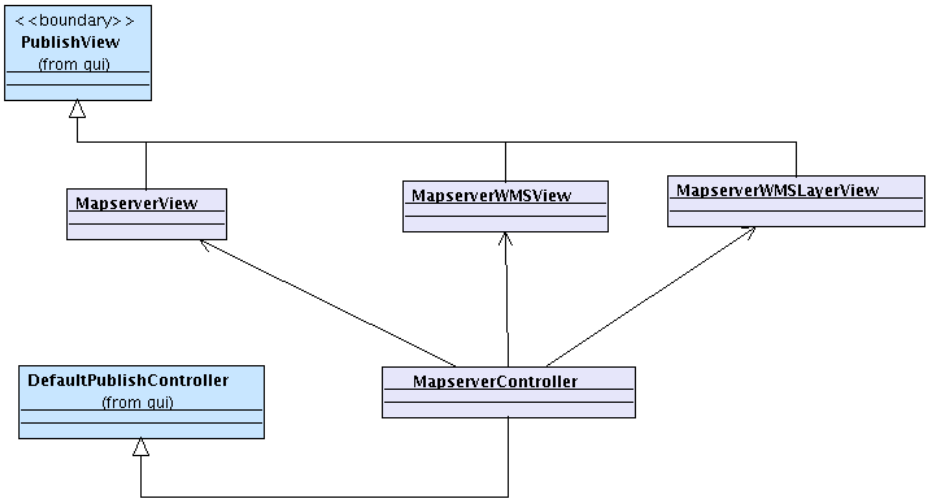

Figure 4. Diagram of objects provided by the MapServer plugin to the GUI component

The plugin also contains implementations of the Server, Service and RemoteResource objects, specific to a certain technology, and will register these components using the methods provided by the modelServers library. Figure 5 shows, as an example, the classes implemented by the MapServer plugin.

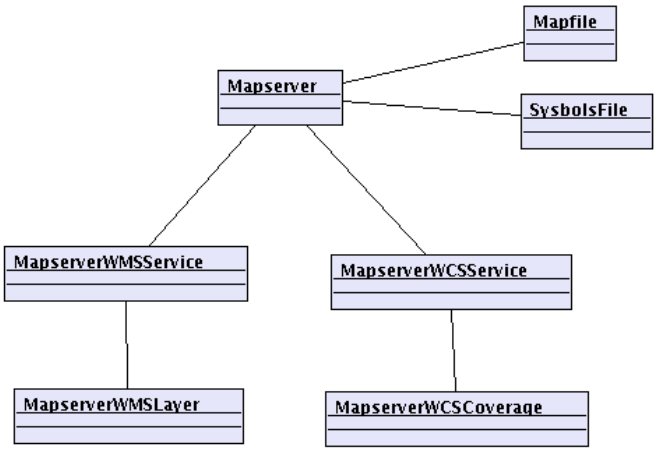

Figure 5. Diagram of modelServers library objects provided by the MapServer plugin

3. Use of the extension

The publishing extension works on a gvSIG View, a set of layers that is displayed together and has an associated legend with the symbols used by its layers. Once the user is satisfied with the contents of a view, a new “Publication document” can be created as part of the current gvSIG project (see Figure 6) and opened in order to launch the publishing wizard.

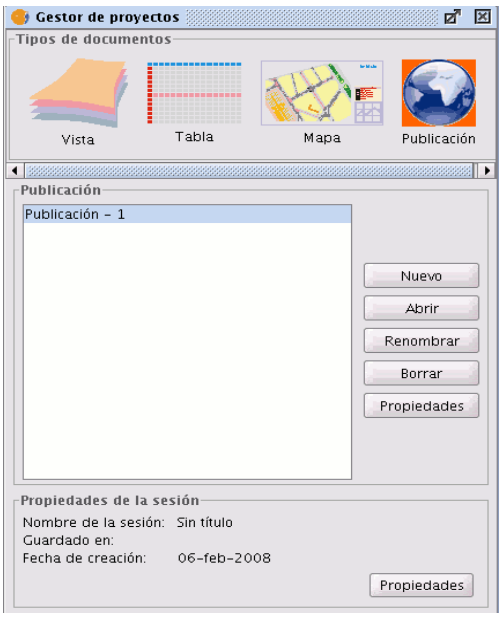

Figure 6. Creation of a Publication document inside the gvSIG project

The wizard runs through a number of pages that result from the GUI configuration as defined by the components registered by the plugins. Specifically, the user will first see the common Main Window (see Figure 7), where the type of Server can be selected (among those registered), as well as one of the Services (among those registered for the selected Server). Other parameters in this initial window are the URL of the Server deployment (it must be accessible by the extension user) and the path, relative to the Server, where this will be able to find the source data.

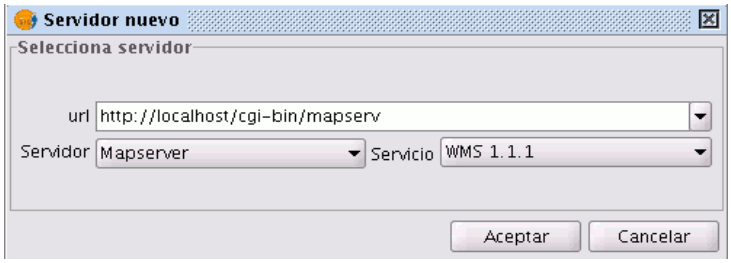

Figure 7. Initial window of the publishing wizard

Note that when a WMS service is selected, the configuration for the service is built to include all layers in the view in a single WMS, while when WFS or WCS options are selected, only vector layers (WFS) or raster layers (WCS) will be included in the published service.

The next window to be displayed in the wizard will be specific to the selected Server, and registered as such by the associated plugin. It will typically include inputs for the parameters needed by the selected Server's technology. For instance, to publish in GeoServer it will be required to enter the user, password, GML namespace and local path for the configuration file (see Figure 8). Instead, when publishing to MapServer, it will be enough to specify the location for the configuration file ('mapfile') and a temporal folder where the user has write access.

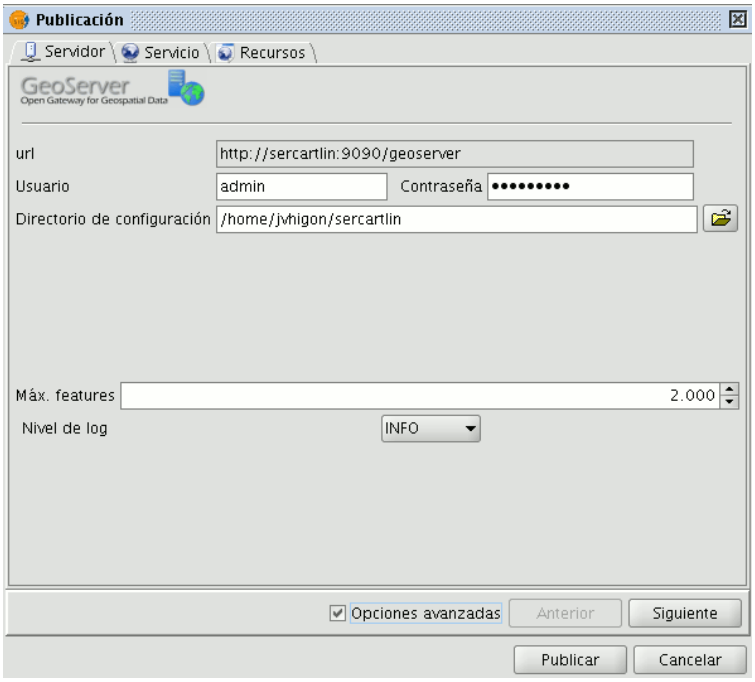

Figure 8. Example of Server-specific parameters page (for GeoServer)

Finally, a last page will be shown with parameters specific to the selected Service, which should also have been registered with this purpose. For instance (see Figure 9), in the case of WMS services published for MapServer the window will have input fields to change the default title, abstract, extent and spatial reference system for each layer.

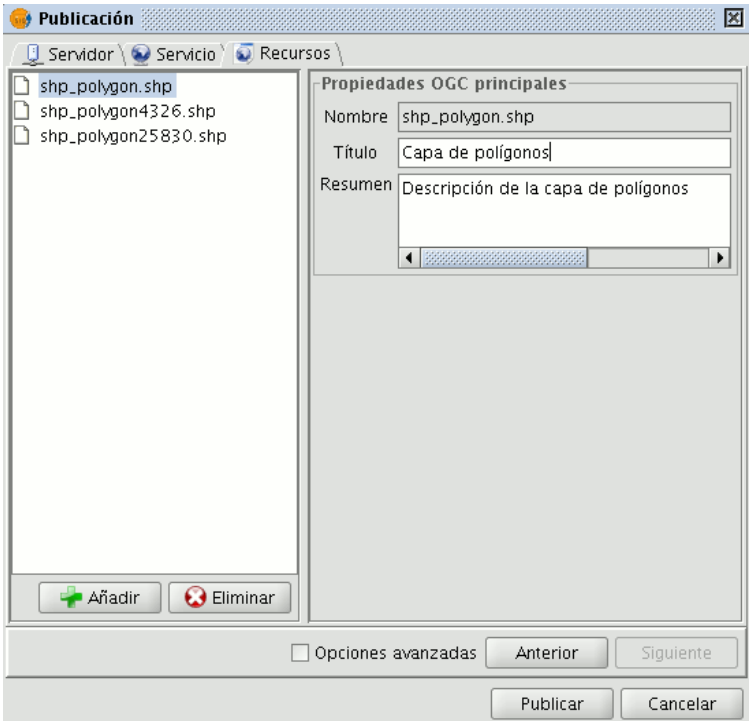

Figure 9. Example of Service-specific parameters page (for MapServer's WMS)

4. Output examples

To test the validity of the publication, the services configured with the publishing wizard can be displayed in gvSIG or other SDI clients. The actual output of the publishing process is the set of configuration files generated at the server location. For instance, Figure 10 shows a fragment of a MapServer mapfile generated from a gvSIG view with two layers, the first one using a PostGIS database, and the second a JPEG image.

```
LAYER
      NAME "limites"
      STATUS ON
      TRANSPARENCY 50
      TYPE POLYGON
      DATA "the_geom from limites"
      CONNECTIONTYPE POSTGIS
      CONNECTION "user=knoppix dbname=postgis host=localhost port=5432"
      METADATA
         "wms_name" "limites"
         "wms_title" "limites"
         "wms_extent" "89332.3125 3957958.75 631363.0 4300894.5"
      END
      CLASS
         NAME "limites"
         STYLE
            COLOR -1 -1 -1
            OUTLINECOLOR 0 0 0
         END
         SIZE 1
      END
   END # Layer
LAYER
      NAME "relieve"
      STATUS ON
      TRANSPARENCY 0
      TYPE RASTER
      DATA "rl6-1500.jpg"
      METADATA
         "wms_name" "relieve"
         "wms_title" "relieve"
         "wms_extent" "89332.3125 3957958.75 631363.0 4300894.5"
      END
   END # Layer
END # Map File
```

Figure 10. Example of mapfile fragment generated for a MapServer WMS

In the case of Geoserver, the extension generates two files with the global configuration (*catalog.xml* and *services.xml*) and one file (*info.xml*) for each 'Geoserver feature', that is, for each remote resource (see Figure 11). It is worth mentioning that all these parameters are automatically extracted from the view and layers properties in gvSIG through the infoProject API, saving users metadata documentation effort.

```
<featureType datastore = "postgis" >
<name>carreteras</name>
<!--
native wich EPGS code for the FeatureTypeInfoDTO
-->
<SRS>23030</SRS>
<title>carreteras</title>
<abstract>Generated from postgis</abstract>
<numDecimals value = "8" />
<keywords>postgis carreteras</keywords>
<latLonBoundingBox dynamic = "false",
                   miny = "35.96331549176131",
                   maxy = "38.74314706366716",
                   maxx = "-1.6056846589838207",
                   minx = "-7.5355498464821355"  />
<!--
the default style this FeatureTypeInfoDTO can be represented by.
at least must contain the "default" attribute.
-->
<styles default = "capitals"  />
<cacheinfo enabled = "false" maxage = ""  />
</featureType>
```

Figure 11. Example of ‘feature’ spec file generated for GeoServer

5. Conclusions and additional work

The publishing extension for gvSIG provides a convenient way to quickly generate the configuration of standard map and data services, and has helped already many users in real workflows involving the publication of data already managed in a desktop GIS. For instance, it has helped institutions participating in the Andean Information System for Disaster Prevention and Relief (Molina, 2008) to easily generate WMS configurations without the need to go through a steep learning curve.

The extension's architecture has been designed with extensibility in mind, with the goal of attracting developers, specially those familiar with the different server technologies and service standards, to implement new plugins for those servers and services not covered in the initial release (which are WMS, WFS and WCS for MapServer, and WFS for GeoServer).

Another aspect in which future work is necessary is the capability to read existing configuration files, so the extension works not only as a one-way generator, but as a full server/service configuration manager.

The publishing extension has been recently released as part of the gvSIG 1.x extension set in http://www.gvsig.gva.es/index.php?id=2009&L=2&K=1%2C. Its source code is also accessible via the public gvSIG SVN repository at

http://subversion.gvsig.org/gvSIG/branches/v10/extensions/extPublish*.

References

Díaz L. et al 2008. 'Multipurpose metadata management in gvSIG'. *FOSS4G 2008*, Cape Town, South Africa.
<http://conference.osgeo.org/index.php/foss4g/2008>

gvSIG 2004. gvSIG Project web site. Consellería de Infraestructuras y Transportes. Generalitat Valenciana.
<http://www.gvsig.gva.es/>

gvSIG 2007. gvSIG Project, developers web site.
<http://www.gvsig.org/web/docdev>

INSPIRE 2007, 'Directive 2007/2/EC of the European Parliament and of the Council of 14 March 2007 establishing an Infrastructure for Spatial Information in the European Community (INSPIRE)'.
<http://www.ec-gis.org/inspire/>

Mapfile reference. *MapServer Project* web site.
<http://mapserver.gis.umn.edu/docs/reference/mapfile>

Molina M. et al 2008. 'The Andean Information System for Disaster Prevention and Relief: a case study of multi-national open-source SDI'. *FOSS4G 2008*, Cape Town, South Africa.
<http://conference.osgeo.org/index.php/foss4g/2008>