OpenMIS Servlet & Services

Open Mobile IS Servlet

- Init parameters

org.openmobileis.services.servlet.OpenMISServlet is the servlet class, to be declared to the web server. It can take two Init Parameters, one of them being required :

org.openmis.services.installpath=<install-path> *(optional)*
This parameter is optional. It defines an extra-path to the application directory (user.dir). If defined, the application directory will be initialized to : user.dir = OpenMISServlet.getServletContext().getRealPath(extra-path).

Otherwise, it will be defined to :
user.dir = OpenMISServlet.getServletContext().getRealPath().

org.openmis.services.conffile=<conf-file> *(required)*
This parameter is required. It gives the path to OpenMIS' configuration file. The application will try to find it :
- first, relatively to the application directory (user.dir).
- then, if not found : as an absolute path.

- Application directory (user.dir)

Let's talk a bit more about the application directory (user.dir). As exposed before, this property is initialized by the servlet to its context path, (+ extra-path, if defined by the init parameter 'org.openmis.services.installpath').
The use of this variable isn't an obligation for developpers. They absolutely can ignore it, and refer to all files and directories by their absolute path (these paths may not have anything to do with the user.dir).

Nevertheless, it is highly recommended to put all datas used by the application into that directory (or sub-directories). If all files and directories are present in the user.dir (and its sub-directories), they can all be referred to as : $user.dir/path_to_file_or_direcotry. Acting this way, makes it really easy to move the application from a machine to an other. Indeed, even if the context path are different, the application will find all its files and directories, with no need for the developper to modify the path referring them.

- Main configuration file

Once started, the servlet parses it's property file (org.openmis.services.conffile) where it finds informations required for it's execution. We'll now call this file the **MainPropertyFile**. It contains a list of properties respecting the following syntax :

<option1>=<value1>
<option2>=<value2>

It mostly looks as follows :

```
# OpenMobileIS properties.
# server version
serve.version=1.0
# force pass management if declared
serve.service.forcepass=false
# default local to use if no local are defined.
serve.intl.defaultinitlocal=fr
# properties path
propsDir=$user.dir/WEB-INF/conf/properties/
# logs path
LOGFILE=$user.dir/WEB-INF/logs/openmis.log
#templates path
templatesDir=$user.dir/WEB-INF/templates/
# List Services
listServicesFile=$user.dir/WEB-INF/conf/properties/listServices.properties
```

Its content will be detailed later, but we can notice the last property called listServicesFile. This property defines a file where the application will find a list of services (we'll now call this file **ListServicesFile**). Indeed, OpenMobileIS servlet, works as a service manager. We talk about it in the comming part.

Open Mobile IS Services

OpenMobileIS servlet, works as a service manager, redirecting all requests to the appropriate service. All these services are identified by a unique service path.
The **ListServicesFile** must have the following syntax :

<service-path-1>=<service-class-1>
<service-path-2>=<service-class-2>

It can look like : /HelloWorld=com.ecare.group.testwebserver.embedded.services.HelloWorld /service/path=com.ecare.group.testwebserver.embedded.services.test /service/path2=com.ecare.group.testwebserver.embedded.services.test2

In this case, the request URI are tokenized as follows :

Request URI	Service path
/services/HelloWorld	/HelloWorld
/services/service/path	/service/path
/services/service/path2	/service/path2

All Services extend the abstract class org.openmobileis.embedded.services.Service and must implement the method 'public void run (HttpServletRequest req, HttpServletResponse res)'.

In our exemple, when the URI /openmis/HelloWorld is called, the run method of the class HelloWorld is called. It receives as parameters, the HttpServletRequest and the HttpServletResponse as defined in the Servlet API. The developper can retreive parameters from the request and generate an answer, just as he would with a normal servlet.

To make it easier, OpenMobileIS provides a Special service that uses a template engine. The template engine used is freemarker (http://freemarker.sourceforge.net/).
This service is called TemplateService : org.openmobileis.embedded.services.common.TemplateService.
All services extending this class must implement the method :
'public String runTemplate(HttpServletRequest req, HttpServletResponse res, TemplateModelRoot templateData)'.

As in a normal Service, the TemplateService receives a HttpServletRequest and a HttpServletResponse, but it receives a third argument :
a **TemplateModelRoot**. The TemplateModelRoot is part of freemarker's API. It is the structure used to store the variables that will be replaced in the template. We can notice that the runTemplate method returns a String. This string indicates the path of the template file, in which variables stored in the TemplateModelRoot will be replaced.
The way freemarker replaces variables in a template is very simple : freemarker recognises variables because of there special syntax : ${variable}. Thus, when the TemplateModelRoot stores a variable called 'toto', freemarkers looks for all occurences of ${toto} in the template, and replaces them with its value, stored in the TemplateModelRoot.