SpagoBI How To 1.6

Authors Fiscato Luca
Zerbetto Davide

Index

1 VERSION....3

2 DOCUMENT GOAL....3

3 REFERENCES....3

4 SPAGOBI HOW TO....3

4.1 How to add a new datawarehouse connection ?....3
4.1.1 SpagoBI and Engines datasource use....3
4.1.2 Provide database driver....4
4.1.3 Two kind of datasources and connections definition....4
4.1.4 SpagoBI core datasource definition....5
4.1.5 SpagoBI engines connection definition....6
4.2 How to change SpagoBI metadata database ?....8
4.2.1 Hibernate metadata database configuration....8
4.3 How to change SpagoBI content repository system ?....9
4.4 How to config the initial functionalities structure ?....10
4.5 How to add a new language to SpagoBI ?....11
4.6 How to change security implementation ?....11
4.7 How to change user profile builder ?....12
4.8 How to change logging configuration ?....12
4.9 How to change query by example (QBE) page size ?....13
4.10 How to limit SQL query result on Qbe ?....13
4.11 How to change portal role filter ?....13
4.12 Who and how can change portlet preferences ?....14
4.13 How to define a datasource as an application server jndi resource ?....14
4.13.1 Tomcat jndi datasource....14
4.13.2 JOnAS jndi datasource....15
4.13.3 Jboss jndi datasource....16
4.14 How to configure SpagoBI for iReport plugin ?....17
4.15 How to add predefined Groovy script ?....18
4.16 How to use profile attributes in lov ?....18
4.17 How to configure data filtering using SpagoBI JpivotEngine ?....20
4.18 How to deploy a new Datamart into SpagoBIQbeEngine ?....22
4.19 How to define a new connection into SpagoBIQbeEngine ?....23
4.20 How to configure SpagoBIQbeEngine administrator users ?....24
4.21 How to define a new connection into SpagoBIGeoEngine ?....25
4.22 How to deploy a new Map into SpagoBIGeoEngine ?....25

1 Version

Version/Release n° :	1.3	**Data Version/Release :**	April, 11rd 2006
Update description:	SpagoBI How To		
Version/Release n° :	1.4	**Data Version/Release :**	July, 20th 2006
Update description:	SpagoBI How To		
Version/Release n° :	1.5	**Data Version/Release :**	October, 9th 2006
Update description:	SpagoBI How To		
Version/Release n° :	1.6	**Data Version/Release :**	January, 23th 2006
Update description:	SpagoBI How To		

2 Document goal

This document provides an explanation about the SpagBI configuration, answering to some questions that focus on some specific aspects.

3 References

Some of the concepts of this document refer to the following documentation:

- Spago framework (available at http://spago.eng.it)
- SpagoBI business intelligence platform framework (available at http://spagobi.eng.it/)

4 SpagoBI How To

4.1 How to add a new datawarehouse connection ?

SpagoBI platform uses datawarehouse connections to get data and fill BI documents. SpagoBI core and its engines are indipendent applications and so each connection has to be defined for both. SpagoBI, actually, doesn't manage only single connection but also pools of connections, so in the following we will refer to connection pool as 'datasource' (which is a component that allows to get a database connection) and to single connection as 'connection'.

4.1.1 SpagoBI and Engines datasource use

SpagoBI uses the defined connection for the definition of a list of value (lov) that retrieves its values with a query. The list of values will be used as the set of possible values for a parameter. The

graphical interface for the definition of the query allows to choose the database target of the query from a combo, which contains all the connection pool defined.

The SpagoBI Engines use their internal datasource definitions to get data and fill BI documents. Each engines could define more connections, each one identified by a unique logical name, but only one of them is the 'default' that will be used every time the engine doesn't receive a specific request for another connection.

To tell an engine to use a specific connection it's necessary to:

- for SpagoBIJasperReportEngine, SpagoBIJPivotEngine, SpagoBIBirtReportEngine and SpagoBIWekaEngine: define a document parameter with 'connectionName' as url name and with the connection logical name as value;
- for SpagoBIGeoEngine: the name of the connection must be set in the document template (see examples in the SpagoBI demo).

4.1.2Provide database driver

Every time you define a new connection or datasource towards a database server you must provide also a database jdbc driver (a compress file with jar extension that you can download from database vendor site). The driver library should be put under different folders for each application server:

- Tomcat: tomcat-home/common/lib
- JOnAS: jonas-home/lib/commons/jonas
- JBoss: jboss-home/server/default/lib

So before define a new connection check that inside the right folder there's the driver library. In case there's no driver library for your database server you must download it and put it into the right folder.

4.1.3Two kind of datasources and connections definition

For both engines and SpagoBI core there are two possible ways to define a database datasource or connection:

- Define a Spago Connection Pool passing all the necessary parameters, like database class driver, database connection string, user name, and so on.
- Retrive the connection pool defined into the application server as a jndi resource. (look at 'How to define a database datasource as an application server jndi resource' section to learn how to define a jndi datasource)

These two approaches have some advantages and disadvantages:

- Defining a Spago Connection pool is surely more easy but following this way it not possible to use the same database pool within others applications. Each application needs to define its own pool and this causes problems of redundant and distributed configuration.

- Defining a connection pool as an application server jndi object and retreive it from spagobi is more difficult and require some system knowledge but allows to define the pool only once and use it within each application deployed into the application server.

4.1.4SpagoBI core datasource definition

All the datasource definition for SpagoBI core must be defined into the spagobi-application-folder/WEB-INF/conf/data-access.xml file.

To define a directly jdbc connection pool add the following piece of xml:

```
<CONNECTION-POOL connectionPoolName="dwh"
     connectionPoolFactoryClass="it.eng.spago.dbaccess.pool.DBCPConnectionPoolFactory"
     connectionPoolType="native"
       connectionDescription="Foodmart Data Warehouse">
      <CONNECTION-POOL-PARAMETER parameterName="connectionString"
            parameterValue="jdbc:postgresql://localhost:5432/foodmart"
            parameterType=""/>
      <CONNECTION-POOL-PARAMETER parameterName="driverClass"
            parameterValue="org.postgresql.Driver"
            parameterType=""/>
      <CONNECTION-POOL-PARAMETER parameterName="driverVersion" parameterValue="2.1"
                              parameterType=""/>
      <CONNECTION-POOL-PARAMETER parameterName="user" parameterValue="postgres"
                              parameterType=""/>
      <CONNECTION-POOL-PARAMETER parameterName="userPassword" parameterValue="postgres"
                              parameterType=""/>
      <CONNECTION-POOL-PARAMETER parameterName="poolMinLimit" parameterValue="1"
                              parameterType=""/>
      <CONNECTION-POOL-PARAMETER parameterName="poolMaxLimit" parameterValue="10"
                              parameterType=""/>
      <CONNECTION-POOL-PARAMETER parameterName="sqlMapperClass"
                  parameterValue="it.eng.spago.dbaccess.sql.mappers.OracleSQLMapper"
                  parameterType=""/>
</CONNECTION-POOL>
```

Obviously you have to change at least the values of parameters: `connectionPoolName` / `connectionDescription` / `connectionString` / `driverClass` / `user` / `password`.

To define a connection pool retriving it as a jndi resource add the following piece of xml:

```
<CONNECTION-POOL connectionPoolName="dwh"
      connectionPoolFactoryClass="it.eng.spago.dbaccess.pool.JNDIConnectionPoolFactory"
      connectionPoolType="native"
      connectionDescription="Foodmart Data Warehouse">
        <CONNECTION-POOL-PARAMETER parameterName="initialContext"
                                   parameterValue="java:comp/env"/>
        <CONNECTION-POOL-PARAMETER parameterName="resourceName"
                                   parameterValue="jdbc/sbifoodmart"/>
        <CONNECTION-POOL-PARAMETER parameterName="driverVersion"
                                   parameterValue="2.1" parameterType=""/>
       <CONNECTION-POOL-PARAMETER parameterName="sqlMapperClass"
                 parameterValue="it.eng.spago.dbaccess.sql.mappers.OracleSQLMapper"
               parameterType=""/>
</CONNECTION-POOL>
```

Obviously you have to change at least the values of parameters: `connectionPoolName` / `connectionDescription` / `initialContext` / `resourceName` based on the name of your jndi datasource

In both case the connectionDescription parameter is used by SpagoBI into the lov query definition as value for the connection selection. (If the parameter is not present the system uses the connectionPoolName). At the end remember to register the new pool adding a xml envelope `<REGISTER-POOL registeredPoolName=" logical name"/>` into the `<CONNECTION-MANAGER>` tag.

4.1.5SpagoBI engines connection definition

This operation can be done editing the file

- **exo-home/webapps/SpagoBIJPivotEngine/WEB-INF/classes/connections-config.xml** for Jpivot engine;
- **exo-home/webapps/SpagoBIJasperReportEngine/WEB-INF/classes/engine-config.xml** for JasperReport Engine;
- **exo-home/webapps/SpagoBIBirtReportEngine/WEB-INF/classes/engine-config.xml** for BirtReportEngine;
- **exo-home/webapps/SpagoBIWekaEngine/WEB-INF/classes/engine-config.xml** for WekaEngine;
- **exo-home/webapps/SpagoBIGeoEngine/WEB-INF/conf/spago/data-access.xml** for GeoEngine.
- **exo-home/webapps/SpagoBIQbeEngine/WEB-INF/conf/data-access.xml** for SpagoBIQbeEngine

For GeoEngine and SpagoBIQbeEngine the datasource configuration is similar to the one used in SpagoBICore.

In the other engines the configuration for a connection is defined in the same way. Each connection is described by an xml tag and its attributes. It's possible to get the connection from a jndi datasource objects or to create directly with jdbc.

This is an example of jndi connection definition (remember to change parameter value based on your jndi resource name) for SpagoBIBirtReportEngine and SpagoBIJPivotEngine:

```
<CONNECTION name="defaultDWH"
                isDefault="true"
                isJNDI="true"
                initialContext="java:comp/env"
                resourceName="jdbc/sbifoodmart"/>
```

This is an example of jdbc direct connection (remenber to change parameter value based on your database server):

```
<CONNECTION name="postgresDWH"
                 isDefault="false"
                 isJNDI="false"
                 driver="org.postgresql.Driver"
                 user="foodmart"
                 password="foodmart"
                 jdbcUrl="jdbc:postgresql://localhost:5432/foodmart"/>
```

In the first case the attribute 'isJNDI' must be set to true (otherwise to false) and only one connection definition of the file can be set as default (isDefault='true'). Since the SpagoBI platform and the engines application will probably access to the same datawarehouse it strongly advised to define the database datasource as a server jndi resource.

For SpagoBIJasperReportEngine and SpagoBIWekaEngine the configuration is almost identical:

```
<CONNECTIONS default="hsqlfoodmart">
        <CONNECTION name="hsqlfoodmart"
                                isJNDI="true"
                                initialContext="java:comp/env"
                                resourceName="jdbc/sbifoodmart"/>
        <CONNECTION name="…
                … />
</CONNECTIONS>
```

The only difference is about the default connection definition: it has to be declared in the “default” attribute of the tag “CONNECTIONS”.

4.2How to change SpagoBI metadata database ?

SpagoBI store it's metadata into a database and it's possible to choose different database servers. The current SpagoBI release support Postgres, Oracle, Mysql and hsqldb but it's easy to add support for other databases. The connection pool for the database has to be defined only for SpagoBI core (not for the engines) and, since SpagoBI uses Hiberante to manage data insertion, deletion and modification, it's necessary to define an hibernate configuration file.

Remember that the metadata db contains all parameters, lovs, checks, functionalities (folders) hierarchy and documents information while the CMS contains documents templates. If you change your database and you don't make a porting of the data from the old database to the new one the system will not be able to show you any parameter, check, lov, functionality and document, even if documents templates are still inside CMS. On the other hand, if you change CMS storage destination without make a porting from the old CMS to the new one, the system will show you all previously saved parameters, lovs, checks, functionalities and documents, but documents templates will be not available, since the new CMS is empty.

4.2.1Hibernate metadata database configuration

You must provide an hibernate configuration file for your database. SpagoBI contains four hibernate configuration files, one for each database supported, into spagobi-application/WEB-INF/classes:

- hibernate.cfg.postgres.xml (Postgres sb)
- hibernate.cfg.ora.xml (Oracle db)
- hibernate.cfg.mysql.xml (Mysql db)
- hibernate.cfg.hsql.xml (Hsql db)

If you want to add support to a new database you must provide a new configuration files with the right parameter. You don't need to change the whole file but only the following part:

```
<property name="hibernate.connection.datasource">java:/comp/env/jdbc/spagobi</property>
<!--
<property name="hibernate.connection.url">jdbc:postgresql://localhost:5432/spagobi</property>
<property name="hibernate.cglib.use_reflection_optimizer">false</property>
<property name="hibernate.connection.password">spagobi</property>
<property name="hibernate.connection.username">spagobi</property>
<property name="hibernate.connection.driver_class">org.postgresql.Driver</property>
-->
<property name="hibernate.dialect">org.hibernate.dialect.PostgreSQLDialect</property>
<property name="hibernate.show_sql">false</property>
```

As you can see from the extract it's possible to get the database datasource from jndi or to define a direct connection pool. In the first case you have to comment the rows for jdbc connection, in the second one you have to erase comment for jdbc data part and add a comment to the line

```
<property name="hibernate.connection.datasource">java:/comp/env/jdbc/spagobi</property>
```

A very important thing is the Dialect parameter which tells hibernate which sql dialect it must use for the database. You can look at hibernate documentation to find out possible dialects and more information about configuration.

At the end remember to tell SpagoBI which configuration file it must use, so, edit the file spagobi-application/WEB-INF/conf/spagobi/spagobi.xml, search the tag <HIBERNATE-CFGFILE> and change its value putting the name of the right hiberante configuration file.

4.3How to change SpagoBI content repository system ?

SpagoBI uses a content management system to store and version the BI documents templates. The respository used must be compliant to the jsr 170 specification. The current SpagoBI release supports natively jackrabbit (which can be also configured as a jndi resource) implementation. Remember that the cms repository and the metadata db are synchronized so if you change you cms implementation you will no more be able to see the documents templates you uploaded before. When you change your cms system it's always better to clean data (data related with biobject and functionalities) into the metadata database starting with a clean situation, otherwise you should do a porting of the data form the old cms to the new one.

To change the cms system you must edit the file spagobi-application/WEB-INF/conf/cms.xml. and configure the <CONTENTREPOSITORY> tag:

- If the repository is not defined as a jndi resource set the tag as follows:

```
<CONTENTREPOSITORY  class="it.eng.spago.cms.repositoryproviders.JackrabbitRepositoryProvider"
name="jackcms">
<PARAMETERS>
       <PARAMETER name="repository_path" value="${SERVER_HOME}/sbidata/jcrRepositoryFS" />
       <PARAMETER name="conf_file_path" value="${SERVER_HOME}/sbidata/repository.xml" />
</PARAMETERS>
…
</CONTENTREPOSITORY>
```

  substituting `${SERVER_HOME}` with the actual path of the server home in the definition of the following parameters: 'repository_path', that is the folder that will be the file system root folder of you cms repository (the path should start with / and should contain only / as folder separator); 'conf_file_path', that is the path to the repository configuration file. If you know jackrabbit you can also change it's configuration file editing the file `${SERVER_HOME}`/sbidata/repository.xml and changing it's parameters (look at jackrabbit documentation for more information about). Obviously you must also retrieve the jackrabbit libraries and put them into the classpath of the spagobi application (you can simply add them into spagobi-application/WEB-INF/lib)

- If the repository is defined as a jndi resource set the tag as follows:

```
<CONTENTREPOSITORY        class="it.eng.spago.cms.repositoryproviders.JndiRepositoryProvider"
name="jndicms">
  <PARAMETERS>
         <JNDICONTEXTNAME name="java:comp/env"/>
         <JNDIOBJECTNAME name="cms/spagobicms"/>
  </PARAMETERS>
```

Obviously the parameter values depende on the name you gave to the jackrabbit repository jndi resource.

- To add support for a new cms repository you need first to write a class which implement a Spago framework interface and using the new cms system API build and return a jsr 170 repostiory object (Look at Spago framework documentation). Then you should put the compiled class into the classpath of your application and fill the 'class' attribute of the 'contentrepository' tag with it's name.. Remenber to put into the application classpath the libraries api of the new cms system.

All the other parameters are common to all cms repositories:

- Connection data: credential to access to the cms system
- Pool Configuration: Configuration for the session pool associated to the cms repository
- Initial Structure: the intial tree structure that the platform creates into the cms every time it starts
- Namespace: uri for the namespaces used by the system to store content into cms

At the end remember to check that your `<CONTENTREPOSITORY>` name attribute value is equals to the name of the default repository into the tag `<DEFAULTREPOSITORY name="jndicms" />`.

4.4How to config the initial functionalities structure ?

It is possible to config the initial functionalities (folders) tree structure: starting from this initial configuration, users will add more folders and save documents inside them.
The initial structure must be set in file spagobi-application/WEB-INF/conf/spagobi/spagobi.xml inside tag TREE_INITIAL_STRUCTURE: you can add how many nodes you want specifying for each node the following properties:

- code: the label of the folder (this field cannot contain "/" characters);
- name: the name of the folder;
- description: the description of the folder;
- parentPath: the path of the parent folder;
- codeType="LOW_FUNCT" (this property value specifies that the node is a folder; no other value are permissible in initialization).

Start defining the root (the folder with empty parentPath which is the only mandatory functionality) and then proceed towards leaves as the example below:

```
<TREE_INITIAL_STRUCTURE>
        <NODE code="Functionalities" name="Functionalities"
                description="Functionalities" parentPath="" codeType="LOW_FUNCT" />
        <NODE code="SystemFunctionalities" name="System Functionalities"
                description="System Functionalities" parentPath="/Functionalities" codeType="LOW_FUNCT" />
        <NODE code="Reports" name="Reports"
```

```
        description="Reports" parentPath="/Functionalities/SystemFunctionalities" codeType="LOW_FUNCT" />
    <NODE code="PersonalFolders" name="Personal Folders"
        description="Personal Folders" parentPath="/Functionalities " codeType="LOW_FUNCT" />
</TREE_INITIAL_STRUCTURE>
```

When SpagoBI is started, it checks if any functionality exists into the metadata db and if no functionalities are present, it inserts the functionalities defined in <TREE_INITIAL_STRUCTURE>.

Note that this initialization does no affect the CMS: the functionalities structure is stored only into metadata db, while documents templates are stored into CMS in a unique folder. The folder is not visible by the user and it is specified in file spagobi-application/WEB-INF/conf/spagobi/spagobi.xml inside tag BIOBJECTSPATH (this folder must be created by CMS initialization, so must be contained in INITIALSTRUCTURE tag in file spagobi-application/WEB-INF/conf/cms.xml).

4.5How to add a new language to SpagoBI ?

SpagoBI supports internazionalization. The current release supports italian an english language. If you want to add a new translation you have to:

1. create a new file called messages_<<language code>>_<<country code>>.properties into the folder spagobi-application/WEB-INF/classes (an example is messages_en_US.properties)
2. copy inside the new file the content of the spagobi-application/WEB-INF/classes/messages_en_US.properties file
3. translate into your language all the strings on the left side of the equal sign
4. repeat the points 1. ÷ 3. for all the component_…_messages_<<language code>>_<<country code>>.properties files you find in spagobi-application/WEB-INF/classes
5. edit the file spagobi-application/WEB-INF/conf/spagobi/spagobi.xml, search the tag <LANGUAGE_SUPPORTED> and then add into a registration tag for the new language <LANGUAGE default="false" language="<<language code>>" country="<<country code>>" />
6. Add the new language to you portal (see portal documentation)

4.6How to change security implementation ?

SpagoBI use the portal single sign on and imports the portal roles in order to use them to associate permission to BI documents. Since each portal has its own security environment and it's own set of api to access to it, SpagoBI delegates this task to an external component using the factory pattern. The default implementation is provided for exo portal and you can look at it as an example. If you install SpagoBI over a different portal server you need to change this implementation:

- write a java class that implements all the methods of the SpagoBI interface 'it.eng.spagobi.security.IPortalSecurityProvider' (look at SpagoBI code). This class will use the api of your portal to get the security information (roles and users)
- Add the compiled class into the spagobi classpath (you can simply create a jar file and add it into spagobi-application/WEB-INF/lib)
- edit the file spagobi-application/WEB-INF/conf/spagobi/spagobi.xml, search the tag <PORTAL-SECURITY-CLASS> and change it's value putting the complete name of the class you wrote.

4.7How to change user profile builder ?

SpagoBI, when a user executes the login operation, builds a user profile object, filling it with the user data, and put it into the session. The profile object will be used by SpagoBI to define lovs with profile attributes and to obtain the groups/roles of the user. The data to fill the user profile must be taken from the portal (user/groups) and from other security systems and repositories.

Every company has its own user information repository so it's not possible to write a generic service to construct and fill a profile. Due to this problem SpagoBi delegates this task to an external component which can be written for a specific company and requirement. SpagoBI contains a default profile builder component for demo and test purpose, this implementation reads user information from an xml file and you can look at it as an example to start a new one.

To configure your user profile builder implementation you need to:

- write a java class that implements all the methods of the SpagoBI interface 'it.eng.spagobi.security.IUserProfileFactory' (look at SpagoBI code). This class will access to your security repository and use the api of your portal to build a user profile. The user profile object built must be an implementation of the Spago framework interface 'it.eng.spago.security.IEngUserProfile' (look at Spago code and documentation), otherwise SpagoBI will not be able to manage it
- Fill the profile object putting the roles of the user (mandatory)
- Fill the profile object putting the attributes of the user (not mandatory)
- Add the compiled class into the spagobi classpath (you can simply create a jar file and add it into spagobi-application/WEB-INF/lib)
- edit the file spagobi-application/WEB-INF/conf/spagobi/spagobi.xml, search the tag <USER-PROFILE-FACTORY-CLASS> and change the value of the attribute “className” putting the complete name of the class you wrote.

4.8How to change logging configuration ?

SpagoBI uses the logging mechanism of the Spago framework and can be configured to logs different levels of information. To change the logging configuration edit the file spagobi-application/WEB-INF/conf/tracing.xml and change its parameter values:

- trace_min_log_severity: possible value are numbers from 0 to 5. The 0 level is the minimun logs level and the system will write all the logs (include the 'INFO' level). The 5 level is the maximun level and the system will logs only the critical errors.
- Debug: if set to true the system will log the debug information. This log feature is very useful during tests but can't be used in production environment because it generates very large log files
- trace_path: the path where the log files will be stored
- trace_name: the prefix of each log file

4.9How to change query by example (qbe) page size ?

The query by example can be used from users to define their own queries on datawarehouse data and then preview the result. Since its possible that the users define query with a very large resultset SpagoBI use a resultset pagination mechanism to avoid memory problem. The default rows number for each page is 10 but it can be changed:

- edit the file SpagoBIQbeEngine-application/WEB-INF/conf/qbe.xml
- search the tag <QBE-MODE>
- change the page-size attribute with the number of rows that you want

4.10How to limit SQL query result on Qbe ?

Using Qbe, while creating a sql query in expert mode query editor or in the exporting phase, in order to limit the result of the SQL query (to avoid memory problems), you have to write the max rows number inside the value attribute of the tag <QBE-SQL-RESULT-LIMIT> in file SpagoBIQbeEngine-application/WEB-INF/conf/qbe.xml.

4.11How to change portal role filter ?

SpagoBI imports the portal roles/groups in order to use them for associate permission on BI documents. As default SpagoBI will import all the roles of the portal (also if they have a gerarchical structure) but you can add a filter in order to import only some of them. The filter is based on the complete name of the role and on a regular expression. Only roles with a name that matches the regular expression will be imported. To define the filter:

- edit the file spagobi-application/WEB-INF/conf/spagobi/spagobi.xml
- search the tag <ROLE-NAME-PATTERN-FILTER>
- change it's value putting the right regular expression (see java documentation for more information about).

4.12Who and how can change portlet preferences ?

SpagoBI portlets have some preferences that can be set at runtime with the edit portlet mode. In order to let only some users to change these preferences, SpagoBI permits the definition of the roles that a user must have in order to make this modification; those roles must be defined inside <PORTLET_EDIT_MODE_ROLES> in file spagobi-application/WEB-INF/conf/spagobi/spagobi.xml. If the user roles collection does not contain any of them, the user cannot change portlet preferences.

For the time being, no roles can be deleted or added at runtime.

In order to change portlet preferences, click on the edit portlet mode button in the portlet info bar (if you do not see it, click on Edit Page Mode, then on Edit Portlet Properties and check "Show Info Bar"). A form with all the portlet preferences will appear.
You can find all possible values of the preferences for each SpagoBI portlet in file spagobi-application/WEB-INF/portlet.xml.

4.13How to define a datasource as an application server jndi resource ?

The advantage of defining a datasource as a jndi resource is that you can define it only one time but you can use it in every application that can be connected to it. The definition of the resource and it's application link it's different for each application server and you should look at their documentation to learn how to do. Anyway in the following we will give a brief explanation of the main steps for Tomcat, Jboss and JOnAS.

4.13.1Tomcat jndi datasource

- **Define a global jndi datasource**: edit the file exo-home/conf/server.xml, search the GlobalNamingResources tag and add into a definition of a global datasource. Below is reported an example, obvioulsy you need to change the parameters value.

```
<Resource name="jdbc/sbifoodmart" auth="Container" type="javax.sql.DataSource"/>
<ResourceParams name="jdbc/sbifoodmart">
   <parameter>
      <name>factory</name>
      <value>org.apache.commons.dbcp.BasicDataSourceFactory</value>
   </parameter>
   <parameter>
      <name>driverClassName</name>
      <value>org.hsqldb.jdbcDriver</value>
   </parameter>
   <parameter>
      <name>url</name>
      <value>jdbc:hsqldb:hsql://localhost/foodmart</value>
   </parameter>
   <parameter>
      <name>username</name>
      <value>sa</value>
   </parameter>
```

```
    <parameter>
       <name>password</name>
       <value></value>
    </parameter>
    <parameter>
       <name>maxActive</name>
       <value>20</value>
    </parameter>
    <parameter>
       <name>maxIdle</name>
       <value>10</value>
    </parameter>
    <parameter>
       <name>maxWait</name>
       <value>-1</value>
    </parameter>
</ResourceParams>
```

○**Associate the global jndi resources with the context of the applications that use it**: open the folder **exo-tomcat-home/conf/Catalina/localhost** (inside there is a set of files with the same names of the applications deployed), search the files of the applications that use the datasource (if they doesn't exist you need to define them), edit each one of them and add a link to the global jndi resource with the following sintax (change the values based on you configuration)

```
<ResourceLink name="jdbc/sbifoodmart" type="javax.sql.DataSource"
global="jdbc/sbifoodmart" />
```

○**Associate the context resource with the application**: for each application, that use the datasource, edit the file **exo-tomcat-home/webapps/<<name_app>>/WEB-INF/web.xml** and before the last tag add the reference to the application jndi resource like below (based on the configuration it's necessary to change values)

```
<resource-ref>
       <description>Foodmart db</description>
       <res-ref-name>jdbc/sbifoodmart</res-ref-name>
       <res-type>javax.sql.DataSource</res-type>
       <res-auth>Container</res-auth>
</resource-ref>
```

4.13.2JOnAS jndi datasource

○**Define a jndi datasource**: Create a file called **<<name_of_datasource>>.properties**, define inside all the necessary parameters for the connection (as the example below shows), and put it into the **jonas-home/conf** directory.

```
datasource.name jdbc/sbifoodmart
datasource.url jdbc:hsqldb:hsql://localhost:9002/foodmart
datasource.classname org.hsqldb.jdbcDriver
datasource.username sa
datasource.password
datasource.mapper rdb.hsql
jdbc.connchecklevel 0
jdbc.connmaxage 1440
jdbc.maxopentime 60
jdbc.connteststmt select 1
jdbc.minconpool 10
jdbc.maxconpool 30
jdbc.samplingperiod 30
```

```
jdbc.maxwaittime 5
jdbc.maxwaiters 100
```

○**Register a jndi datasource:** edit the file **jonas-home/conf/jonas.properties**, search the property **jonas.service.dbm.datasources** and at the end of the row add the names of yours datasource (separated with commas)

○**Associate the jndi resource with the context of the applications that use it**: For each application that use the datasource define a **jonas-web.xml** file into **webapps-home/WEB-INF** folder in order to map global datasource to the application context like the example below:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE jonas-web-app PUBLIC "-//ObjectWeb//DTD JOnAS Web App 2.6//EN"
"http://www.objectweb.org/jonas/dtds/jonas-web-app_2_6.dtd">
<jonas-web-app>
   <jonas-resource>
      <res-ref-name>jdbc/sbifoodmart</res-ref-name>
      <jndi-name>jdbc/sbifoodmart</jndi-name>
   </jonas-resource>
</jonas-web-app>
```

○**Associate the context jndi resource with the application:** for each application that use the datasource edit the **webapps-home/WEB-INF/web.xml** and add a reference to the context jndi datasource like the example below:

```
<resource-ref>
   <description>Foodmart db</description>
   <res-ref-name>jdbc/sbifoodmart</res-ref-name>
   <res-type>javax.sql.DataSource</res-type>
   <res-auth>Container</res-auth>
</resource-ref>
```

4.13.3Jboss jndi datasource

○**Define a jndi datasource**: Create a file called **<<name_of_datasource>>-ds.xml**, define inside all the necessary parameters for the connection (as the example below shows) and put it into the **jboss-home/server/default/deploy** directory.

```
<?xml version="1.0" encoding="UTF-8"?>
<datasources>
 <local-tx-datasource>
  <jndi-name>spagobi</jndi-name>
  <connection-url>jdbc:hsqldb:hsql://localhost:9002/spagobi</connection-url>
  <driver-class>org.hsqldb.jdbcDriver</driver-class>
  <user-name>sa</user-name>
  <password></password>
  <min-pool-size>5</min-pool-size>
```

```
    <max-pool-size>20</max-pool-size>
    <metadata>
      <type-mapping>Hypersonic SQL</type-mapping>
    </metadata>
  </local-tx-datasource>
</datasources>
```

- **Associate the jndi resource with the context of the applications that use it**: for each application that use the datasource define a **jboss-web.xml** file into **webapps-home/WEB-INF** folder in order to map global datasource to the application context like the example below:

```
<jboss-web>
  <resource-ref>
    <res-ref-name>jdbc/sbifoodmart</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <jndi-name>java:/sbifoodmart</jndi-name>
  </resource-ref>
</jboss-web>
```

- **Associate the context jndi resource with the application:** for each application that use the datasource edit the **webapps-home/WEB-INF/web.xml** and add a reference to the context jndi datasource like the example below:

```
<resource-ref>
   <description>Foodmart db</description>
   <res-ref-name>jdbc/sbifoodmart</res-ref-name>
   <res-type>javax.sql.DataSource</res-type>
   <res-auth>Container</res-auth>
</resource-ref>
```

4.14How to configure SpagoBI for iReport plugin ?

In order to let SpagoBI iReport plugin work with SpagoBI platform you have to set the name of the context of the portal application inside tag <NAME_PORTAL_APPLICATION> in file spagobi-application/WEB-INF/conf/spagobi/spagobi.xml.

This plugin will show you the collection of report type documents that the logged user can see in the SpagoBI functionalities tree; check-in or check-out operations are possible. Since report documents can be relevant to different report engines, there is an init parameter for the IReportServlet (in file spagobi-application/WEB-INF/web.xml), that is the servlet invoked by SpagoBI iReport plugin; in this parameter you have to write the driver names (comma-separated) of the engines of the reports that you want to be displayed on that tree in iReport plugin. For example: you want the reports executed by JasperReport to be visualized and the reports executed by Birt not to be visualized, so you have to configure the servlet as follows:

```
<servlet>
        <servlet-name>IReportServlet</servlet-name>
        <servlet-class>it.eng.spagobi.services.IReportPluginServlet</servlet-class>
        <init-param>
                <param-name>DRIVERS</param-name>
                <param-value>it.eng.spagobi.drivers.jasperreport.JasperReportDriver</param-value>
        </init-param>
</servlet>
```

4.15How to add predefined Groovy script ?

Before starting the server you can edit the file spagobi-application/WEB-INF/classes/predefinedGroovyScript.groovy: in this file you can add methods and classes as you want. This file will be loaded before any Groovy script execution so you can call those methods and classes in any script you write in the lov detail page.

4.16How to use profile attributes in lov ?

Profile attributes can be used inside a lov (query, script or fix lov), either they are single-value or multi-value.

- A single-value profile attribute should respect the form:
  profile attribute name = profile attribute value
- A multi-value profile attribute should respect the form:
  profile attribute name = {splitter character{list of values separated by the splitter character}}

Queries

In a query the profile attribute declaration must respect the sintax ${name_of_profile_attribute} if it is single-value and ${name_of_profile_attribute(prefix;splitter;suffix)} if it is multi-value: in the latter case the declaration will be substituted by *prefix + list of values separeted by the splitter + suffix*.

Examples:

Single value profile attribute:
Profile attribute name = “store_city”
Profile attribute value = “Padova”

The following query:
“select * from customers where city = ‘${store_city}’”

will become
“select * from customers where city = ‘Padova”

Multi value profile attribute:
Profile attribute name = “store_city”
Profile attribute values = “{,{Padova,Roma,Venezia}}”

The following query:
“select * from customers where city in ${store_city((‘;’,’;‘))}”
will become
“select * from customers where city in (‘Padova’,’Roma’,’Venezia‘)”

Fix Lov

In a fix lov the profile attribute declaration must respect the sintax ${name_of_profile_attribute} if it is single-value and ${name_of_profile_attribute(prefix;splitter;suffix)} if it is multi-value: in the latter case the declaration will be substituted by *prefix + list of values separeted by the splitter + suffix*.

Examples:

Single value profile attribute:
Profile attribute name = “store_city”
Profile attribute value = “Padova”

The following lov element:
Name: “Store city”
Value: “${store_city}”
will become
Name: “Store city”
Value: “Padova”

Multi value profile attribute:
Profile attribute name = “store_city”
Profile attribute values = “{,{Padova,Roma,Venezia}}”

The following lov element:
Name: “Store city”
Value: “${store_city((’;’,’;’))}”
will become
Name: “Store city”
Value: “(‘Padova’,’Roma’,’Venezia‘)”

Scripts

In a script you can use the predefined groovy methods:

- getSingleValueProfileAttribute(*attribute name*) for single-value profile attributes: this method will return the value of the profile attribute as a String;
- getMultiValueProfileAttribute(*attribute name*, *prefix*, *splitter*, *suffix*) for multi-value profile attributes: this method will return a String composed by *prefix* + *list values separeted by the splitter* + *suffix*.

Examples:

Single value profile attribute:
Profile attribute name = "store_city"
Profile attribute value = "Padova"

The following script:
"getSingleValueProfileAttribute(store_city)"
will return the string
"Padova"

Multi value profile attribute:
Profile attribute name = "store_city"
Profile attribute values = "{,{Padova,Roma,Venezia}}"

The following script:
"getMultiValueProfileAttribute(store_city, '(', ';', ')')"
will return the string
"(Padova;Roma;Venezia)"

4.17How to configure data filtering using SpagoBI JpivotEngine ?

Starting from the SpagoBI version 1.9.1 it's possible, using SpagoBI JpivotEngine as olap module and eXo as portal server, to configure a filter on olap enquire results, based on the roles/groups of the user. Once configured, SpagoBI platform will be able to interact with JpivotEngine giving the possibility to the users to access only the data allowed to them. This means that if the user, once entered the jpivot interface, change the mdx query, trying to access different data area, the system blocks him if he doesn't have the right permissions.

First of all, in order to configure the filter, you need to be sure that you have the following SpagoBI security configuration inside the spagobi.xml configuration file:

```
<PORTAL-SECURITY-CLASS className="it.eng.spagobi.security.ExoMembershipAsRoleSecurityProviderImpl">
        <CONFIG>
                <NAME_PORTAL_APPLICATION>name of your portal context</NAME_PORTAL_APPLICATION>
        </CONFIG>
</PORTAL-SECURITY-CLASS>
```

```
<USER-PROFILE-FACTORY-CLASS
       className=”it.eng.spagobi.security.ExoMembershipAsRoleUserProfileFactoryImpl” >
       <FUNCTIONALITIES-LOADER>
              <GROUP-TRANSFORMERS>
                     <TRANSFORMER startwith="/Products" prefix="data_access:"
                                  dimension="Products" />
              </GROUP-TRANSFORMERS>
       </FUNCTIONALITIES-LOADER>
</USER-PROFILE-FACTORY-CLASS>
```

These security classes transform the eXo user memberships into SpagoBI user roles and, if configured, eXo user groups into SpagoBI user functionalities (associated to the SpagoBI user roles). The user functionalities can be considered as grants of the user on particular areas or activities (what the user can do or see). In case of olap filtering each functionalities is a filter path where the first token is the name of the dimension to filter and the others are the names of level elements allowed.

The exo groups translation into functionalities is applied only if the transformation is configured. To configure a transformation you need to specify at least one 'transformer' element. Each transformer element is a rule for the functionalities assignment based on the user groups. The attributes of the transformer element are:

- startwith: the user group names which start with the value of the attribute are assigned as user functionalities
- prefix: the value of the attribute is put before the name of the user group
- dimension: the first token of the user group is substitute with the value of the attribute. This behaviour is useful because often the first token has not the same name as the datawarehouse dimension

Example:

```
eXo user configuration:

exo user name: biadmin
exo user groups: /Products/Food/Potatoes with membership consumer
                 /Products/Food/Tomatoes with membership producer

becomes a SpagoBI profile like this:

SpagoBI user name: biadmin
SpagoBI user roles: consumer, producer
SpagoBI user functionalities: data_access:/Products/Food/Potatoes (for SpagoBI role consumer)
                              data_access:/Products/Food/Tomatoes (for SpagoBI role producer)
```

When a new olap document is defined, if you want to use the filter function, its template must have the following sintax:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<OLAP>
    <CUBE reference='/WEB-INF/queries/LapamMart.xml' />
    <MDXquery>
        mdx query
    </MDXquery>
    <DATA-ACCESS>
       <GRANTED-DIMENSIONS>
          <DIMENSION name="Products" grantSource="ProfileFunctionalities" />
       </GRANTED-DIMENSIONS>
    </DATA-ACCESS>
```

```
</OLAP>
```

The entire element 'data-access' can be omitted but in that case no one filter is applied. The element 'granted-dimensions' can contain more 'dimension' elements. Each one of this elements means that you want to use the filter function for the specified dimension.

When you exec the olap document with the previous configuration SpagoBI

- recovers the list of granted dimensions from the template
- for each one granted dimension tag it recovers the dimension name
- recovers all the functionalities from the user profile
- discards all the functionalities that don't start with the string data_access:/<<dimension name>>
- sends to the engine the functionalities not discarded

When the request arrives to the SpagoBI JpivotEngine the engine recovers all the functionalities sent and uses them to set the grant for the dimension and relative level elements. All the other elements of the dimension will be deny.

Based on the previous example, if the user exec the olap document with the consumer role:

- JpivotEngine receives the functionality data_access:/Products/Food/Potatoes
- The engine set a grant on the 'Products' dimension only for element '[Food].[Potatoes]', all the other elements will be deny
- For all the other dimension the engine doesn't set access grants

As a result if the user changes the mdx trying to see the data relative to [Food].[Tomatoes] he will get an error because he doesn't have the permission.

4.18How to deploy a new Datamart into SpagoBIQbeEngine ?

SpagoBIQbeEngine is a web application usefull to build your own queries starting from an object analytical model of your source database. The application is based on Hibernate technology which allows to define a database model using java classes and xml mapping files. All the Hibernate files relative to an analytical model are grouped into a single archive file (file with jar extension call datamart).
Every time you want to add an analytical model to the SpagoBIQbeEngine application you need first to build the relative jar file. Read the 'HOW TO CREATE DATABASE MODEL JAR FILE' section of the SpagoBI quickstart document in order to learn how to do it. Once created the jar file it's necessary to deploy it into the SpagoBIQbeEngine application:

- connect to the url http://servername:port/SpagoBIQbeEngine
- click on the presentation image

- into the subsequent login form insert your SpagoBIQbeEngine administrator credential (Look at the question 'How to configure SpagoBIQbeEngine administrator users' to learn how to get or change SpagoBIQbeEngine credentials)
- Once authenticated, if you are authorized to add a new datamart, an upload form will appear.
- Fill the 'Jar File' field, using the 'File' button, with the jar file of the analytical model.
- Fill the 'Datamart Name' field with a logical name for the analytical model. (If the name already exists you will get an error after the upload).
- Press the upload button.
- If the deploy operation complete correctly the new model will appear in the list above the upload form
- Each row of the list has two buttons, one for the deletion of the datamart and one for the execution

4.19How to define a new connection into SpagoBIQbeEngine ?

Once you have deployed a new datamart file into the SpagoBIQbeEngine (see the 'How to deploy a new Datamart into SpagoBIQbeEngine' question) you can exec it. The datamart is only a description of the model, it doesn't contain data, so it must be associated to a datasource. For this reason, in order to exec a datamart, the engine needs a connection to the source database.
The database connections avaiable for your SpagoBIQbeEngine installation must be defined into a configuration file and they must have been previously defined as jndi datasource of the application server. So, in order to add a new database connection to your SpagoBIQbeEngine application, proceed as follow:

- be sure that the application server where the engine is installed is not running
- define the datasource into your application server as a jndi connection (Look to the question 'How to define a datasource as an application server jndi resource' to learn ho to do).
- Edit the file <<server application folder>>/SpagoBIQbeEngine/WEB-INF/conf/data_access.xml
- into the <DATA-ACCESS> tag place a child tag like this

```
<CONNECTION name="logical name of the connection"
                    jndiResourceName="jndi name of the connection"
                    jndiContext="name of the jndi initial context" />
```

Example:

```
<CONNECTION name="hsqlFoodmart"
                    jndiResourceName="jdbc/sbifoodmart"
                    jndiContext="java:comp/env" />
```

N.B. in a JOnAS server leave the jndiContext attribute value blank

- restart the server

4.20How to configure SpagoBIQbeEngine administrator users ?

Users can connect to the SpagoBIQbeEngine application directly (without passing from SpagoBI). This kind of access is restricted by authentication and it's usefull for SpagoBIQbeEngine administrators who have to deploy new anlytical model (see 'How to deploy a new Datamart into SpagoBIQbeEngine' question). To configure access credentials for administrator users proceed as follow:

- be sure that the application server where the engine is installed is not running
- Edit the file <<server application folder>>/SpagoBIQbeEngine/WEB-INF/conf/authorizations.xml
- search the tag <USERS>
- for each user you want to add insert a new child tag, like the one below, and change its values. (you can also change the existing one)

  <USER userID="spagobi" password="f3IXvwaGV9HL0MwWxG8g+QPdoSg=" />

  The userID attribute contains the UserName (is not case sensitive)
  The password attribute contains the encrypted password of the user.
  To generate a correct password you need to use the graphical interface released with the binary package of SpagoBIQbeEngine, so:

  - get the binary package of SpagobIQbeEngine (download it if you don't have it)
  - unzip it everywhere you want
  - exec the file QbeEnginePasswordGenerator.bat on a window system or QbeEnginePasswordGenerator.sh in a unix system
  - In the graphical interface (GUI) type the password you want to assign to the user and then click on the 'Generate' button
  - copy the encrypted password that appear in the 'Encrypted Password' field and paste it as value of the password attribute of the xml file.

- close the GUI using the abort button
- search the tag <BEHAVIOURS>
- for each user added insert three child tags like the ones below

  <BEHAVIOUR userID="username" roleName="QbeDev" />
  <BEHAVIOUR userID="username" roleName="QbeAdmin" />
  <BEHAVIOUR userID="username" roleName="QbeUser" />
  Obviously replace the username string with the name of the user inserted

- restart the server

4.21How to define a new connection into SpagoBIGeoEngine ?

Once you have deployed a new map file into the SpagoBIGeoEngine (see the 'How to deploy a new map into SpagoBIGeoEngine' question) you can exec it. During the execution process, the engine try to obtains data from a database and then it associates the data to the map. For this reason, in order to exec a map document, the engine needs a connection to at least one source database.
The database connections avaiable for your SpagoBIGeoEngine installation must be defined into a configuration file, so, in order to add a new database connection, proceed as follow:

- be sure that the application server where the engine is installed is not running
- Edit the file <<server application folder>>/SpagoBIGeoEngine/WEB-INF/conf/data_access.xml
- Configure the file in order to add one or more database connections. Since SpagoBIGeoEngine is developed using the Spago framework you can learn how to configure the file reading the section 'SpagoBI core datasource definition' of the question 'How to add a new datawarehouse connection'. (or you can look at the Spago framework documentation)
- restart the server

4.22How to deploy a new Map into SpagoBIGeoEngine ?

SpagoBIGeoEngine is a web application which transforms and shows a geographic map, based on the datawarehouse data. The base geographic map must be an svg file which contains all the boundaries and elements of the map. The web application manage a catalogue of available maps. Each map contained is identified by an unique name. When a SpagoBI developer defines a new Map Document he must configure the template of the document (look at maps section of the quickstart document) to refer to an existing map of the SpagoBIGeoEngine catalogue.
In order to add a map to the SpagoBIGeoEngine catalogue proceed as follow:

- be sure that the application server where the engine is installed is not running
- open the folder <<server application folder>>/SpagoBIGeoEngine/maps (let's call the folder MAPSROOT)
- decide the logical name of the map you want to add and then create a new folder, into MAPSROOT, with the same name of the map logical name. (let's call the new folder

NEWMAPFOLDER). The logical name will be used by SpagoBI developers to refer the map when they want to create a new SpagoBI map document.

- Copy your map svg file into NEWMAPFOLDER and rename it as 'map.svg'.
- restart the server

Example:

Let's suppose you have an svg map file named 'italymapfile.svg' and you want to add it into the Engine catalogue with the logical name 'Italy'. Open MAPSROOT folder, create a new folder name 'Italy', rename the file named 'italymapfile.svg' to 'map.svg' and then copy the file 'map.svg' inside MAPSROOT/Italy.