SpagoBI How To

Authors Luca Fiscato

Review Grazia Cazzin

Index

1 Version

Version/Release n° :	1.3	Data Version/Release :	May, 18th 2006
Update description:	SpagoBI How To		

2 Document goal

This document provides an explanation about the SpagoBI 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, really, 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 retrives 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 call it passing a parameter 'connectionName' with a value equals to the connection logical name. So, using SpagoBI interface, for a report or olap that needs a specific engine connection, it's necessary to define a biparameter with url_name equal to 'connectionName' and assign it, as value, the engine connection logical name.

4.1.2 Provide 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/
- 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.

4.1.3 Two 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 retrive 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.

4.1.4 SpagoBI 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="spagobi"
     connectionPoolFactoryClass="it.eng.spago.dbaccess.pool.DBCPConnectionPoolFactory"
     connectionPoolType="native"
     connectionDescription="Foodmart Data Warehouse">
      <CONNECTION-POOL-PARAMETER parameterName="connectionString"
            parameterValue="jdbc:postgresql://localhost:5432/spagobi"
            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="spagobi"
                                   parameterType=""/>
        <CONNECTION-POOL-PARAMETER parameterName="userPassword" parameterValue="spagobi"
                                   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.

4.1.5 SpagoBI engines connection definition

This operation can be done editing the file **exo-home/webapps/SpagoBIJPivotEngine/WEB-INF/classes/connections-config.xml** for Jpivot engine and **exo-home/webapps/SpagoBIJasperReportEngine\WEB-INF\classes/engine-config.xml** for JasperReport Engine. In both files 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):

```
<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.

4.2 How 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) but has to be configured two times because:

- SpagoBI uses the Spago framework list service so the metadata connection pool must be defined into data-access.xml
- SpagoBI uses Hiberante to manage data insertion, deletion and modification so it's necessary to define an hibernate configuration file

In the future SpagoBI will use only hibernate and so the first configuration will be no more necessary but for the moment it must be done. Anyway since the two configuration use the same database it's better to define the datasource as a server jndi resource and after retrive it from hibernate and Spago configuration.

Remember that the metadata db is sincronized with the SpagoBI cms system so 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 stop to work.

4.2.1 Spago framework metadata database configuration

Edit the file spagobi-application/WEB-INF/conf/data-access.xml and search the xml envelope with 'connectionPoolName' equals to 'spagobi' (spagobi it's the mandatory name for the metadata connection) and comment it (the whole tag with also the content). Define another connection tag (with the same logical name) which define the connection to your metadata database in a jndi or jdbc manner. To learn how to do look at sections:

- 'How to add a new datawarehouse connection' to learn the difference between jndi a jdbc connection and define one of them into the data-access.xml
- 'How to define a database datasource as an application server jndi resource' to learn how to configure a jndi datasource

Remember that the metadata connection must have the 'connectionPoolName' equals to 'spagobi'

4.2.2 Hibernate 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.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 from 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.

4.3 How 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) and exo v2 jcr (which works only on a exo v2 ecm version) implementations. Remember that the cms repository and the metadata db are sincronized so if you change you cms implementation you will no more be able to see the documents you configured 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:

- To use the jackrabbit cms repository start the tag like below

  ```
  <CONTENTREPOSITORY class="it.eng.spago.cms.JackrabbitRepositoryFactoryImpl" name="jackcms">
  ```

  Then edit the spagobi-application/WEB-INF/classes/jackrabbitSessionFactory.properties file and change the value of property 'repository_path' with a folder path that will be the file system root folder of you cms repository (the path should start with / and should contain only / as folder separator). If you know jackrabbit you can also change it's configuration editing the file spagobi-application/WEB-INF/repository.xml and changing it's parameters (look at jackrabbit documentation for more information about). Obviously you must also retrive the jackrabbit libraries and put them into the classpath of the spagobi application (you can simply add them into spagobi-application/WEB-INF/lib)

- To use the exo ecm v2 jcr repository start the tag like below

  ```
  <CONTENTREPOSITORY class="it.eng.spago.cms.ExoRepositoryFactoryImpl" name="exocms">
  <PARAMETERS>
          <CONTEXTNAME name="ecm"/>
  </PARAMETERS>
  ```

  Only if you add an application portal (similar to the ecm) remember to change the contextname based on you portal application name

- To retrive the cms repository as a jndi resource start the tag like below (you must had configured a jackrabbit repository as a jndi resource)

```
<CONTENTREPOSITORY class="it.eng.spago.cms.JndiRepositoryFactoryImpl" 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" />`

4.4 How 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:

- 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)
- copy inside the new file the content of the spagobi-application/WEB-INF/classes/messages_en_US.properties file
- translate in you language all the strings on the left side of the equal sign
- 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" />

- Add the new language to you portal (see portal documentation)

4.5 How 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 '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.

4.6 How to change user profile builder ?

SpagoBI, when a user execs the login operation, build 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 retrive 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 write 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 '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 class you write should return a user profile object which implement a Spago framework 'IEngUserProfile' interface,which define a generic profile object (look at Spago code and documentation), otherwise SpagoBI will not be able to manage it
- It's mandatry to fill the user profile with the roles of the user and you can also add an attribute called 'PROFILE_ATTRIBUTES' which contains an hashmap of all the user attributes retrived from the security repostiory.
- 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 it's value putting the complete name of the class you wrote.

4.7 How 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

4.8 How 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 15 but it can be changed:

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

4.9 How 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).

4.10 How to define a database 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.

4.10.1 Tomcat 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 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>
```

4.10.2 JOnAS 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.conncheckleve l 0
jdbc.connmaxage 1440
jdbc.maxopentime 60
jdbc.conntesteststmt 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>
```

4.10.3 Jboss 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>
```