Table of Contents

JVx allows the development of database-independent applications. However, access to the database requires specific information, such as host name, database port, database name, username, password, etc.

This information is usually saved in a configuration file. We use the applications configuration config.xml that is available for our application on the server side:

config.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>

<application>
  <securitymanager>
<class>com.sibvisions.apps.packung.security.AnonymousDBSecurityManager<
/class>
    <database>
      <url2>jdbc:oracle:thin:@localhost:1521:packung</url2>
      <url>jdbc:derby://localhost:1527/packung</url>
      <username>pack_user</username>
      <password>pack_password</password>
    </database>
  </securitymanager>

  <!-- predefined life cycle object names -->
  <lifecycle>
    <mastersession>com.sibvisions.apps.packung.Session</mastersession>
  </lifecycle>
</application>
```

As this example shows, we use the url parameter to define access to the database. During the development of a database-independent application, the database has to be changed to test its functionality. For this purpose we replace the content of url with url2.

After the database has been defined, the application has to access it.

To access the database, an object of type IDBAccess is required. A number of different implementations exist, such as OracleDBAccess or DerbyDBAccess.

The developer could now use a specific implementation, such as OracleDBAccess, to access only Oracle databases. Although this allows the use of special features provided by OracleDBAccess, if the database is changed, the source code has to be altered as well.

In this case, we chose a generic approach to access the database:

```java
DBAccess dba = DBAccess.getDBAccess(cfgSession.getProperty
("/application/securitymanager/database/url"));

dba.setUsername(cfgSession.getProperty("/application/securitymanager/databas
e/username"));
dba.setPassword(cfgSession.getProperty("/application/securitymanager/databas
e/password"));
dba.open();
```

This way we always receive the correct object for access to the configured database.