Table of Contents

Test DBStorages Without Lifecycle Objects 1
A Custom Application Without Menu, Toolbar, etc. 1
Test Your Business Logic With JUnit 3
A Very Simple AbstractMemStorage Implementation 7
Change XML Files Quickly 8
EventHandler Without Listener Interface 9
EventHandler With Listener Interface 10
A Custom AccessController 12
Remote Calls to JVx Without UI 14
Use JVx Swing Controls Without JVx UI 15
Example for Automatic Link Cell Editors 17
Encrypt Passwords 34
Fetch Data With DBStorage 35
50% Width With FormLayout 36
Search Column in Condition 36
Connection Property Changed Listener 37
Test UI With Simple JUnit Test 38
Authentication/User Logging 46
Detect Production Mode 50

We have a list of useful code snippets for you. Simply use them for your application. All snippets are free to use and licensed under Apache 2.0.

Test DBStorages Without Lifecycle Objects

Access a DBStorage without JVx Server, Lifecycle Object, Security, e.g., for unit tests.

ContactsAutoFrame.java

```
//configure DB access
DBAccess dba = new DBAccess();
dba.setUrl("...");
dba.setUsername("user");
dba.setPassword("pwd");
dba.open();

//configure storage for Table USERS
DBStorage dbs = new DBStorage();
dbs.setFromClause("USERS");
dbs.setDBAccess(dba);
dbs.open();

//direct object connection for direct method calls
DirectObjectConnection con = new DirectObjectConnection();
con.put("users", dbs);

//client connection for RemoteDataBook access to the storage
MasterConnection macon = new MasterConnection(con);
macon.open();

RemoteDataSource rds = new RemoteDataSource(macon);
rds.open();

RemoteDataBook rdbApps = new RemoteDataBook();
rdbApps.setDataSource(rds);
rdbApps.setName("users");
rdbApps.open();
```

A Custom Application Without Menu, Toolbar, etc.

Sometimes we need an application without overhead, e.g., for Vaadin UI.

SimpleApplication.java

```
public class SimpleApplication extends Application
                               implements IExceptionListener
{
   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
// Class members
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

/** the main/content panel. */
private UIPanel panMain;

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
// Initialization
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

/**
 * Creates a new instance of <code>SimpleApplication</code>.
 *
 * @param pLauncher the launcher
 */
public SimpleApplication(UILauncher pLauncher)
{
   super(pLauncher);

   setName("Simple application");

   init();
}

/**
 * Initializes the application.
 */
private void init()
{
   ExceptionHandler.addExceptionListener(this);

   panMain = new UIPanel();
   panMain.setLayout(new UIBorderLayout());

   panMain.add(YOUR COMPONENT, UIBorderLayout.CENTER);

   setLayout(new UIBorderLayout());
   add(panMain);
}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
// Interface implementation
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

public IContainer getContentPane()
{
   return panMain;
}

public <OP> IContent showMessage(OP pOpener,
                                 int pIconType,
```

```
                                    int pButtonType,
                                    String pMessage,
                                    String pOkAction,
                                    String pCancelAction) throws
Throwable
   {
      System.out.println(pMessage);

      return null;
   }

   public void handleException(Throwable pThrowable)
   {
      System.out.println(CommonUtil.dump(pThrowable, false));
   }

}   // SimpleApplication
```

Test Your Business Logic With JUnit

Tests our business logic not with an application server, but with our Lifecycle objects. We test our server code without specific configuration or modifications for unit tests.

Business object:

UserRegistration.java

```
public class UserRegistration
{
   /**
    * Removes/Unregisters a user and all its content.
    *
    * @param pUserName the user name
    * @throws Exception if the user can not be deleted
    */
   public void delete(String pUserName) throws Exception
   {
      try
      {
         DBAccess dba =
(DBAccess)SessionContext.getCurrentSession().get("newDBAccess");

         DBStorage dbsUser = new DBStorage();
         dbsUser.setDBAccess(dba);
         dbsUser.setWritebackTable("USERS");
         dbsUser.open();

         List<IBean> liBeans = dbsUser.fetchBean(new Equals("USERNAME",
```

```
                                                                pUserName),
null, 0, -1);

         //the username is unique and it's not possible that a user
exists
         //more than once!
         if (liBeans.size() == 1)
         {
            //delete the user (db will cascade all user-data)
            dbsUser.delete(liBeans.get(0));
         }
         else
         {
            throw new SecurityException("User '" + pUserName + "' was
not found!");
         }
      }
      catch (Throwable th)
      {
         throw new SecurityException("It's not possible to delete the
user!", th);
      }
   }
}   // UserRegistration
```

A standard life cycle object:

Session.java

```
public class Session extends GenericBean
{
   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   // User-defined methods
   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   /**
    * Creates a new database access object.
    *
    * @return the new database access object
    * @throws Exception if the connection can not be opened
    */
   public DBAccess getNewDBAccess() throws Exception
   {
      DBAccess dba =
DBAccess.getDBAccess(DBSecurityManager.getCredentials(
SessionContext.getCurrentSessionConfig()));
      dba.open();

      return dba;
   }
```

```
   /**
    * Returns the access to the database.
    *
    * @return the access to the database
    * @throws Exception if the datasource can not be opened
    */
   public DBAccess getDBAccess() throws Exception
   {
      DBAccess dba = (DBAccess)get("dBAccess");

      if (dba == null)
      {
         dba = getNewDBAccess();

         put("dBAccess", dba);
      }

      return dba;
   }

   /**
    * Gets the user registration business object.
    *
    * @return the business object for user registrations
    */
   public UserRegistration getRegistration()
   {
      UserRegistration ureg = (UserRegistration)get("registration");

      if (ureg == null)
      {
         ureg = new UserRegistration();

         put("registration", ureg);
      }

      return ureg;
   }

}    // Session
```

Unit test:

```
/**
 * Tests the our delete method from the user registration object.
 *
 * @throws Throwable if the test fails
 */
@Test
```

```
public void testDelete() throws Throwable
{
   AbstractConnection con = createConnection(null);

   try
   {
      //delete user with the name "unknownuser"
      con.call("registration", "delete", "unknownuser");

      Assert.fail("User 'unknownuser' found!");
   }
   catch (SecurityException se)
   {
      Assert.assertEquals("User 'unknownuser' was not found!",
                          se.getMessage());
   }

   con.close();
}

/**
 * Creates a new connection.
 *
 * @param pConProps additional connection properties
 * @return the connection
 * @throws Throwable if the connection can not be opened
 */
private AbstractConnection createConnection(Hashtable<String, String>
pConProps) throws Throwable
{
   MasterConnection macon = new MasterConnection(new
DirectServerConnection());
   macon.setApplicationName("app");
   macon.setUserName("user");
   macon.setPassword("pwd");

   if (pConProps != null)
   {
      for (Map.Entry<String, String> entry : pConProps.entrySet())
      {
         macon.setProperty(entry.getKey(), entry.getValue());
      }
   }

   macon.open();

   return macon;
}
```

A Very Simple AbstractMemStorage Implementation

A server-side memory storage with the column names: ID, NAME, PATH. The column PATH is not visible on the client side, but is important for server side. If “error” is set as NAME, an exception is thrown!

SimpleMemStorage.java

```
public class SimpleMemStorage extends AbstractMemStorage
{
   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   // Abstract methods implementation
   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   /**
    * {@inheritDoc}
    */
   @Override
   public RowDefinition getRowDefinition() throws ModelException
   {
      RowDefinition rowdef = new RowDefinition();
      rowdef.addColumnDefinition(new ColumnDefinition("ID", new
BigDecimalDataType()));
      rowdef.addColumnDefinition(new ColumnDefinition("NAME", new
StringDataType()));
      rowdef.addColumnDefinition(new ColumnDefinition("PATH", new
StringDataType()));

      rowdef.setPrimaryKeyColumnNames(new String[] {"ID"});

      rowdef.setColumnView(null, new ColumnView("ID", "NAME"));

      return rowdef;
   }

   /**
    * {@inheritDoc}
    */
   @Override
   public void loadData(MemDataBook pBook, ICondition pFilter) throws
ModelException
   {
      pBook.deleteAllDataRows();

      pBook.insert(false);
      pBook.setValues(new String[] {"ID", "NAME", "PATH"},
                      new Object[] {BigDecimal.valueOf(0), "First",
"/home/first"});
      pBook.insert(false);
      pBook.setValues(new String[] {"ID", "NAME", "PATH"},
```

```
                        new Object[] {BigDecimal.valueOf(1), "Second",
"/home/second"});
      pBook.insert(false);
      pBook.setValues(new String[] {"ID", "NAME", "PATH"},
                        new Object[] {BigDecimal.valueOf(2), "Third",
"/home/third"});
   }

   /**
    * {@inheritDoc}
    */
   @Override
   public void insert(DataBookEvent pEvent) throws ModelException
   {
      if
("error".equals(pEvent.getChangedDataBook().getValueAsString("NAME")))
      {
         throw new ModelException("not allowed");
      }
   }

   @Override
   public void delete(DataBookEvent pEvent)
   {
   }

   @Override
   public void update(DataBookEvent pEvent) throws ModelException
   {
      if
("error".equals(pEvent.getChangedDataBook().getValueAsString("NAME")))
      {
         throw new ModelException("not allowed");
      }
   }

}   // SimpleMemStorage
```

Change XML Files Quickly

Our XML file:

```
<server>
  <!-- Test: STARTPORT -->
  <startport>2001</startport>

  <audio>off</audio>
  <serial>COM1</serial>
```

```
  <domain>JVx</domain>
</server>
```

Change it:

```
XmlNode xmnRead = readXml("simple.xml");

xmnRead.setNode("/server/audio", null);
xmnRead.setNode("/server/domain", "www.sibvisions.com");

writeXml(xmnRead, "simple.xml");
```

EventHandler Without Listener Interface

Event definition:

```
/** the event handler for captured (the event has one parameter: byte[]). */
private CallableHandler chCaptured = new CallableHandler(byte[].class);

/** the event handler for canceled (the event has no parameter). */
private CallableHandler chCanceled = new CallableHandler();
```

Event access:

```
/**
 * Gets the captured event handler.
 *
 * @return the event handler
 */
public CallableHandler eventCaptured()
{
   return chCaptured;
}

/**
 * Gets the canceled event handler.
 *
 * @return the event handler
 */
public CallableHandler eventCanceled()
{
   return chCanceled;
}
```

Dispatch events:

```
chCaptured.dispatchEvent(byData);
chCancel.dispatchEvent();
```

Listener registration:

```
object.eventCaptured().addListener(this, "doCapture");

public void doCapture(byte[] pImage) throws Exception
{
    //...
}
```

EventHandler With Listener Interface

The interface:

IRemoteApplicationListener.java

```
public interface IRemoteApplicationListener
{
   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   // Method definitions
   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    /**
     * Invoked when login was successful.
     *
     * @param pApplication the application
     */
    public void afterLogin(RemoteApplication pApplication);

    /**
     * Invoked when logout was successful.
     *
     * @param pApplication the application
     */
    public void afterLogout(RemoteApplication pApplication);

}   // IRemoteApplicationListener
```

The EventHandler:

RemoteApplicationHandler.java

```
public class RemoteApplicationHandler extends
RuntimeEventHandler<IRemoteApplicationListener>
{
   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   // Initialization
   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
    /**
     * Constructs a new RemoteApplicationHandler.
     *
     * @param pListenerMethodName the method to be called inside the
interface.
     */
    public RemoteApplicationHandler(String pListenerMethodName)
    {
        super(IRemoteApplicationListener.class, pListenerMethodName);
    }

}   // RemoteApplicationHandler
```

Event access:

```
/** the "after login" event. */
private RemoteApplicationHandler eventAfterLogin;

/** the "after logout" event. */
private RemoteApplicationHandler eventAfterLogout;


/**
 * Gets the event handler for the after login event.
 *
 * @return the event handler
 */
public RemoteApplicationHandler eventAfterLogin()
{
    if (eventAfterLogin == null)
    {
        eventAfterLogin = new RemoteApplicationHandler("afterLogin");
    }
    return eventAfterLogin;
}

/**
 * Gets the event handler for the after logout event.
 *
 * @return the event handler
 */
public RemoteApplicationHandler eventAfterLogout()
{
    if (eventAfterLogout == null)
    {
        eventAfterLogout = new RemoteApplicationHandler("afterLogout");
    }
    return eventAfterLogout;
}
```

Dispatch events:

```
/**
 * Fires the after logout event.
 */
protected void afterLogout()
{
   if (eventAfterLogout != null)
   {
      eventAfterLogout.dispatchEvent(this);
   }
}

/**
 * Fires the after login event.
 */
protected void afterLogin()
{
   if (eventAfterLogin != null)
   {
      eventAfterLogin.dispatchEvent(this);
   }
}
```

Listener registration:

```
app.eventAfterLogin().addListener(this, "doAfterLogin");
app.eventAfterLogout().addListener(this, "doAfterLogout");

//We do not need the parameter
public void doAfterLogin()
{
}

//We do not need the parameter
public void doAfterLogout()
{
}
```

A Custom AccessController

Create a custom AccessController implementation, e.g.:

AppAccessController.java

```
package com.sibvisions.apps.test;

import ...
```

```
public class AppAccessController implements IAccessController
{
   /** the allowed lifecycle objects. */
   private ArrayUtil<String> auAllowedLCO = null;

   /**
    * {@inheritDoc}
    */
   public boolean isAllowed(String pLifecycleName)
   {
      //all explicite allowed lifecycle objects are accessible
      if (auAllowedLCO != null)
      {
         return auAllowedLCO.contains(pLifecycleName);
      }

      return false;
   }

   /**
    * {@inheritDoc}
    */
   public void addAccess(String pLifecycleName)
   {
      if (pLifecycleName == null)
      {
         return;
      }

      if (auAllowedLCO == null)
      {
         auAllowedLCO = new ArrayUtil<String>();
      }

      if (!auAllowedLCO.contains(pLifecycleName))
      {
         auAllowedLCO.add(pLifecycleName);
      }
   }
}
```

Configure the application:

config.xml

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

<application>
  <securitymanager>
    <class>com.sibvisions.rad.server.security.DBSecurityManager</class>
```

```
<accesscontroller>com.sibvisions.apps.test.AppAccessController</accessc
ontroller>
  </securitymanager>

  <datasource>
    <db name="default">
      <driver>org.hsqldb.jdbcDriver</driver>
      <url>jdbc:hsqldb:hsql://localhost/demodb</url>
      <username>sa</username>
      <password></password>
    </db>
  </datasource>
</application>
```

Remote Calls to JVx Without UI

JVx is a full-stack application framework, but all parts are independent. It is no problem to use only the communication classes or the Swing controls without GenUI. This snippet shows how it is possible to use JVx only on server side. You have full session handling and the client is built with your preferred UI or if your client has no UI.

Use JVx on server side as usual. Integrate it into an application server like Tomcat or use it standalone.

Use the communication classes to access server objects and actions. A simple object call could be implemented like the following snippet.

Get the source code for a specific class via Server Object:

```
HttpConnection httpcon = new
HttpConnection("http://demo.sibvisions.org/showcase/services/Server");

MasterConnection macon = new MasterConnection(httpcon);
macon.setApplicationName("showcase");
macon.setUserName("admin");
macon.setPassword("admin");
macon.open();

macon.call("sourceCode", "get",
"com.sibvisions.apps.showcase.frames.ChartFrame");
```

Get data from the database:

```
SubConnection subcon =
macon.createSubConnection("com.sibvisions.apps.showcase.frames.Contacts");
subcon.open();

RemoteDataSource dataSource = new RemoteDataSource();
dataSource.setConnection(subcon);
```

```
dataSource.open();

RemoteDataBook rdbContacts = new RemoteDataBook();
rdbContacts.setDataSource(dataSource);
rdbContacts.setName("contacts");
rdbContacts.open();

rdbContacts.fetchAll();
```

Use rdbContacts to insert/update/delete records.

Use JVx Swing Controls Without JVx UI

JVx's Swing controls are independent of JVx UI. It is no problem to use a table that shows some database or in-memory records. It is also possible to use three-tier or two-tier architecture.

In-Memory Data

Use JVxTable and integrate it in your Swing application. Use it like a standard JTable.

JVxTableTest.java

```
public class JVxTableTest extends JFrame
{
   public static void main(String[] pArgs) throws Throwable
   {
      new JVxTableTest();
   }

   JVxTableTest() throws Throwable
   {
      //--------------------------------
      // Data
      //--------------------------------

      MemDataBook mdbData = new MemDataBook();
      mdbData.setName("person");
      mdbData.getRowDefinition().addColumnDefinition(new
ColumnDefinition("FIRSTNAME"));
      mdbData.getRowDefinition().addColumnDefinition(new
ColumnDefinition("LASTNAME"));
      mdbData.getRowDefinition().addColumnDefinition(new
ColumnDefinition("PHONE"));
      mdbData.open();

      mdbData.insert(true);
      mdbData.setValues(new String[] {"FIRSTNAME", "LASTNAME",
"PHONE"},
```

```
                        new String[] {"JVx", "rocks", "+43 000 / 1234
567"});

      mdbData.saveAllRows();

      //---------------------------------
      // Swing
      //---------------------------------

      JVxTable table = new JVxTable();
      table.setDataBook(mdbData);

      setLayout(new BorderLayout());

      add(table, BorderLayout.CENTER);

      setPreferredSize(new Dimension(500, 400));
      pack();
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setVisible(true);
   }

}   // JVxTableTest
```

You have all JVx features like link cell editors, image choice editors - in your Swing application!

Two-Tier Data (Database)

Replace the MemDataBook with following code:

```
//---------------------------------
// Database handling
//---------------------------------

//DB Connection
DBAccess dba = DBAccess.getDBAccess("jdbc:hsqldb:hsql://localhost/db");
dba.setUsername("sa");
dba.setPassword("");
dba.open();

//Table access
DBStorage dsFiles = dba.createStorage();
dsFiles.setWritebackTable("FILES");
dsFiles.setAutoLinkReference(false);
dsFiles.open();

//---------------------------------
// Communication
```

```
//--------------------------------

//Connection handling
DirectObjectConnection con = new DirectObjectConnection();
con.put("files", dsFiles);

MasterConnection macon = new MasterConnection(con);
macon.open();

//Connect to the "remote" table
RemoteDataSource rds = new RemoteDataSource(macon);
rds.open();

RemoteDataBook rdbData = new RemoteDataBook();
rdbData.setDataSource(rds);
rdbData.setName("files");
rdbData.open();
```

Three-Tier Data (Database)

Remote DBAccess, DBStorage and DirectObjectConnection and replace the MasterConnection with following code:

```
HttpConnection con = new
HttpConnection("http://server/app/services/Server");

MasterConnection macon = new MasterConnection(con);
macon.setApplicationName("showcase");
macon.setUserName("admin");
macon.setPassword("admin");
macon.open();
```

Example for Automatic Link Cell Editors

The client code:

ContactsAutoFrame.java

```
/*
 * Copyright 2009 SIB Visions GmbH
 *
 * Licensed under the Apache License, Version 2.0 (the "License"); you
may not
 * use this file except in compliance with the License. You may obtain
a copy of
 * the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
```

```
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT
 * WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See
the
 * License for the specific language governing permissions and
limitations under
 * the License.
 *
 *
 * History
 *
 * 27.11.2009 - [HM] - creation
 * 22.02.2010 - [JR] - changed column names for auto storages
 */
package com.sibvisions.apps.showcase.frames;

import java.io.ByteArrayOutputStream;
import java.io.IOException;

import jvx.rad.genui.UIDimension;
import jvx.rad.genui.UIInsets;
import jvx.rad.genui.celleditor.UIDateCellEditor;
import jvx.rad.genui.celleditor.UIImageViewer;
import jvx.rad.genui.celleditor.UINumberCellEditor;
import jvx.rad.genui.component.UIButton;
import jvx.rad.genui.component.UILabel;
import jvx.rad.genui.container.UIGroupPanel;
import jvx.rad.genui.container.UIPanel;
import jvx.rad.genui.container.UISplitPanel;
import jvx.rad.genui.control.UIEditor;
import jvx.rad.genui.layout.UIBorderLayout;
import jvx.rad.genui.layout.UIFormLayout;
import jvx.rad.io.IFileHandle;
import jvx.rad.model.ColumnDefinition;
import jvx.rad.model.ColumnView;
import jvx.rad.model.ModelException;
import jvx.rad.model.RowDefinition;
import jvx.rad.model.condition.ICondition;
import jvx.rad.model.condition.LikeIgnoreCase;
import jvx.rad.model.datatype.StringDataType;
import jvx.rad.model.reference.ReferenceDefinition;

import com.sibvisions.apps.showcase.Showcase;
import com.sibvisions.apps.showcase.components.NavigationTable;
import com.sibvisions.apps.showcase.components.SourceAccessFrame;
import com.sibvisions.rad.model.mem.DataRow;
import com.sibvisions.rad.model.remote.RemoteDataBook;
import com.sibvisions.rad.model.remote.RemoteDataSource;
import com.sibvisions.util.type.FileUtil;
```

```
import com.sibvisions.util.type.ImageUtil;

/**
 * The <code>ContactsAutoFrame</code> shows contacts, their detail
information
 * and all educations. The automatic link celleditors will be detected
automatically. It shows
 * how less code a developer has to write. You can compare the LoC with
the
 * {@link ContactsFrame} which handles automatic link celleditors
definition manually.
 *
 * @author Martin Handsteiner
 */
public class ContactsAutoFrame extends SourceAccessFrame
                               implements IDataScreen
{
   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   // Class members
   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   /** the default image when no image was found. */
   private static final String NO_IMAGE =
"/com/sibvisions/apps/showcase/images/nobody.gif";


   /** the DataSource for fetching table data. */
   private RemoteDataSource dataSource = new RemoteDataSource();

   /** storage for contacts. */
   private RemoteDataBook rdbContacts = new RemoteDataBook();
   /** storage for contacts educations. */
   private RemoteDataBook rdbContEduc = new RemoteDataBook();

   /** search row. */
   private DataRow drSearch = null;

   /** the frames layout. */
   private UIBorderLayout blThis = new UIBorderLayout();
   /** the split between contacts and details. */
   private UISplitPanel splitMain = new UISplitPanel();
   /** the navigator for contacts. */
   private NavigationTable navContacts;
   /** the navigagor for showing educations. */
   private NavigationTable navContEdu;

   /** the layout for details. */
   private UIFormLayout flDetails = new UIFormLayout();
   /** the details. */
   private UIPanel panDetails = new UIPanel();
   /** the details. */
```

```
    private UIGroupPanel gpanDedails = new UIGroupPanel();
    /** the details. */
    private UIGroupPanel gpanEducations = new UIGroupPanel();

    /** contacts layout. */
    private UIBorderLayout blContacts = new UIBorderLayout();
    /** contacts panel. */
    private UIPanel panContacts = new UIPanel();
    /** search panel. */
    private UIPanel panSearch = new UIPanel();

    /** Label. */
    private UILabel lblSalutation = new UILabel();
    /** Label. */
    private UILabel lblAcademicTitle = new UILabel();
    /** Label. */
    private UILabel lblFirstName = new UILabel();
    /** Label. */
    private UILabel lblLastName = new UILabel();
    /** Label. */
    private UILabel lblStreet = new UILabel();
    /** Label. */
    private UILabel lblNr = new UILabel();
    /** Label. */
    private UILabel lblZip = new UILabel();
    /** Label. */
    private UILabel lblTown = new UILabel();
    /** Label. */
    private UILabel lblCountry = new UILabel();
    /** Label. */
    private UILabel lblBirthday = new UILabel();
    /** Label. */
    private UILabel lblSocialSecurityNr = new UILabel();
    /** Label. */
    private UILabel lblHealthInsurance = new UILabel();
    /** Label. */
    private UILabel lblFilename = new UILabel();
    /** labelSuchen. */
    private UILabel lblSearch = new UILabel();

    /** Editor. */
    private UIEditor edtSalutation = new UIEditor();
    /** Editor. */
    private UIEditor edtAcademicTitle = new UIEditor();
    /** Editor. */
    private UIEditor edtFirstName = new UIEditor();
    /** Editor. */
    private UIEditor edtLastName = new UIEditor();
    /** Editor. */
    private UIEditor edtStreet = new UIEditor();
    /** Editor. */
```

```
   private UIEditor editNr = new UIEditor();
   /** Editor. */
   private UIEditor edtZip = new UIEditor();
   /** Editor. */
   private UIEditor edtTown = new UIEditor();
   /** Editor. */
   private UIEditor edtCountry = new UIEditor();
   /** Editor. */
   private UIEditor edtBirthday = new UIEditor();
   /** Editor. */
   private UIEditor edtSocialSecurityNr = new UIEditor();
   /** Editor. */
   private UIEditor edtHealthInsurance = new UIEditor();
   /** Editor. */
   private UIEditor edtFilename = new UIEditor();
   /** editSuchen. */
   private UIEditor edtSearch = new UIEditor();

   /** contact image. */
   private UIEditor icoImage = new UIEditor();
   /** load image button. */
   private UIButton butLoadImage = new UIButton();

   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   // Initialization
   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   /**
    * Constructs a new instance of <code>ContactsFrame</code>.
    *
    * @param pApplication the application.
    * @throws Throwable if the initialization throws an error
    */
   public ContactsAutoFrame(Showcase pApplication) throws Throwable
   {
      super(pApplication,
"com.sibvisions.apps.showcase.frames.ContactsAuto");

      initializeModel();
      initializeUI();
   }

   /**
    * Initializes the model.
    *
    * @throws Throwable if the initialization throws an error
    */
   private void initializeModel() throws Throwable
   {
      dataSource.setConnection(getConnection());
      dataSource.open();
```

```
      rdbContacts.setDataSource(dataSource);
      rdbContacts.setName("contacts");
      rdbContacts.open();

      //set same labels as in details panel
rdbContacts.getRowDefinition().getColumnDefinition("ACTI_ACADEMIC_TITLE
").setLabel("Academic title");
rdbContacts.getRowDefinition().getColumnDefinition("FIRSTNAME").setLabe
l("First name");
rdbContacts.getRowDefinition().getColumnDefinition("LASTNAME").setLabel
("Last name");
rdbContacts.getRowDefinition().getColumnDefinition("ZIP").setLabel("ZIP
");
rdbContacts.getRowDefinition().getColumnDefinition("BIRTHDAY").setLabel
("DoB");
rdbContacts.getRowDefinition().getColumnDefinition("SOCIALSECNR").setLa
bel("Social security nr");
rdbContacts.getRowDefinition().getColumnDefinition("HEIN_HEALTH_INSURAN
CE").setLabel("Health insurance");

      rdbContEduc.setDataSource(dataSource);
      rdbContEduc.setName("contEduc");
      rdbContEduc.setMasterReference(new ReferenceDefinition(new
String[] {"CONT_ID"}, rdbContacts, new String[] {"ID"}));
      rdbContEduc.open();

      rdbContEduc.getRowDefinition().setColumnView(null, new
ColumnView("EDUC_EDUCATION"));

      UIImageViewer imageViewer = new UIImageViewer();
      imageViewer.setDefaultImageName(NO_IMAGE);

rdbContacts.getRowDefinition().getColumnDefinition("FILENAME").setReadO
nly(true);
rdbContacts.getRowDefinition().getColumnDefinition("IMAGE").getDataType
().setCellEditor(imageViewer);

rdbContacts.getRowDefinition().getColumnDefinition("SOCIALSECNR").getDa
taType().setCellEditor(new UINumberCellEditor("0000"));
rdbContacts.getRowDefinition().getColumnDefinition("BIRTHDAY").getDataT
ype().setCellEditor(new UIDateCellEditor("dd.MM.yyyy"));

      RowDefinition definition = new RowDefinition();
      definition.addColumnDefinition(new ColumnDefinition("SEARCH", new
StringDataType()));

      drSearch = new DataRow(definition);
      drSearch.eventValuesChanged().addListener(this, "doFilter");
   }
```

```
   /**
    * Initializes the UI.
    *
    * @throws Throwable if the initialization throws an error
    */
   private void initializeUI() throws Throwable
   {
      lblSearch.setText("Search");
      edtSearch.setDataRow(drSearch);
      edtSearch.setColumnName("SEARCH");

      UIFormLayout layoutSearch = new UIFormLayout();

      panSearch.setLayout(layoutSearch);
      panSearch.add(lblSearch, layoutSearch.getConstraints(0, 0));
      panSearch.add(edtSearch, layoutSearch.getConstraints(1, 0, -1,
0));

      navContacts = new NavigationTable(getApplication().getLauncher(),
getConnection(), rdbContacts);
      navContacts.getTable().setAutoResize(false);

      panContacts.setLayout(blContacts);
      panContacts.add(panSearch, UIBorderLayout.NORTH);
      panContacts.add(navContacts, UIBorderLayout.CENTER);

      navContEdu = new NavigationTable(getApplication().getLauncher(),
getConnection(), rdbContEduc);
      navContEdu.getTable().setPreferredSize(new UIDimension(150,
150));
      navContEdu.eventNewDetail().addListener(this, "doNewEducations");

      icoImage.setPreferredSize(new UIDimension(75, 75));
      icoImage.setDataRow(rdbContacts);
      icoImage.setColumnName("IMAGE");

      lblSalutation.setText("Salutation");
      lblAcademicTitle.setText("Academic title");
      lblFirstName.setText("First name");
      lblLastName.setText("Last name");
      lblStreet.setText("Street");
      lblNr.setText("Nr");
      lblZip.setText("ZIP");
      lblTown.setText("Town");
      lblCountry.setText("Country");
      lblBirthday.setText("DoB");
      lblSocialSecurityNr.setText("Social security nr");
      lblHealthInsurance.setText("Health insurance");
      lblFilename.setText("Filename");

      edtSalutation.setDataRow(rdbContacts);
```

```
      edtSalutation.setColumnName("SALU_SALUTATION");
      edtSalutation.setPreferredSize(new UIDimension(75, 21));
      edtAcademicTitle.setDataRow(rdbContacts);
      edtAcademicTitle.setColumnName("ACTI_ACADEMIC_TITLE");
      edtAcademicTitle.setPreferredSize(new UIDimension(75, 21));
      edtFirstName.setDataRow(rdbContacts);
      edtFirstName.setColumnName("FIRSTNAME");
      edtLastName.setDataRow(rdbContacts);
      edtLastName.setColumnName("LASTNAME");
      edtStreet.setDataRow(rdbContacts);
      edtStreet.setColumnName("STREET");
      editNr.setDataRow(rdbContacts);
      editNr.setColumnName("NR");
      edtZip.setDataRow(rdbContacts);
      edtZip.setColumnName("ZIP");
      edtTown.setDataRow(rdbContacts);
      edtTown.setColumnName("TOWN");
      edtCountry.setDataRow(rdbContacts);
      edtCountry.setColumnName("CTRY_COUNTRY");
      edtBirthday.setDataRow(rdbContacts);
      edtBirthday.setColumnName("BIRTHDAY");
      edtSocialSecurityNr.setDataRow(rdbContacts);
      edtSocialSecurityNr.setColumnName("SOCIALSECNR");
      edtHealthInsurance.setDataRow(rdbContacts);
      edtHealthInsurance.setColumnName("HEIN_HEALTH_INSURANCE");
      edtFilename.setDataRow(rdbContacts);
      edtFilename.setColumnName("FILENAME");

      butLoadImage.setText("Upload");
      butLoadImage.eventAction().addListener(this, "doUpload");
      butLoadImage.setFocusable(false);

      flDetails.setMargins(new UIInsets(20, 20, 20, 20));

      gpanDedails.setText("Contact");

      gpanDedails.setLayout(flDetails);
      gpanDedails.add(icoImage, flDetails.getConstraints(0, 0, 1, 7));
      gpanDedails.add(butLoadImage, flDetails.getConstraints(0, 8));
      gpanDedails.add(edtFilename, flDetails.getConstraints(1, 8));

      flDetails.setHorizontalGap(15);
      gpanDedails.add(lblSalutation, flDetails.getConstraints(2, 0));
      flDetails.setHorizontalGap(5);
      gpanDedails.add(edtSalutation, flDetails.getConstraints(3, 0));
      gpanDedails.add(lblAcademicTitle, flDetails.getConstraints(2,
1));
      gpanDedails.add(edtAcademicTitle, flDetails.getConstraints(3,
1));
      gpanDedails.add(lblFirstName, flDetails.getConstraints(2, 2));
      gpanDedails.add(edtFirstName, flDetails.getConstraints(3, 2, -1,
```

```
2));
      gpanDedails.add(lblLastName, flDetails.getConstraints(2, 3));
      gpanDedails.add(edtLastName, flDetails.getConstraints(3, 3, -1,
3));

      gpanDedails.add(lblSocialSecurityNr, flDetails.getConstraints(2,
4));
      gpanDedails.add(edtSocialSecurityNr, flDetails.getConstraints(3,
4));
      gpanDedails.add(lblBirthday, flDetails.getConstraints(4, 4));
      gpanDedails.add(edtBirthday, flDetails.getConstraints(5, 4, -1,
4));

      gpanDedails.add(lblHealthInsurance, flDetails.getConstraints(2,
5));
      gpanDedails.add(edtHealthInsurance, flDetails.getConstraints(3,
5, -1, 5));

      gpanDedails.add(lblStreet, flDetails.getConstraints(2, 6));
      gpanDedails.add(edtStreet, flDetails.getConstraints(3, 6, -3,
6));
      gpanDedails.add(lblNr, flDetails.getConstraints(-2, 6));
      gpanDedails.add(editNr, flDetails.getConstraints(-1, 6));
      gpanDedails.add(lblZip, flDetails.getConstraints(2, 7));
      gpanDedails.add(edtZip, flDetails.getConstraints(3, 7));
      gpanDedails.add(lblTown, flDetails.getConstraints(4, 7));
      gpanDedails.add(edtTown, flDetails.getConstraints(5, 7, -1, 7));

      UIFormLayout layoutSchulung = new UIFormLayout();

      gpanEducations.setText("Schooling");
      gpanEducations.setLayout(layoutSchulung);
      gpanEducations.add(navContEdu, layoutSchulung.getConstraints(0,
0, -1, -1));

      UIFormLayout layout = new UIFormLayout();

      panDetails.setLayout(layout);
      panDetails.add(gpanDedails, layout.getConstraints(0, 0, -1, 0));
      panDetails.add(gpanEducations, layout.getConstraints(0, 1, -1,
-1));

      splitMain.setDividerPosition(250);
      splitMain.setDividerAlignment(UISplitPanel.DIVIDER_TOP_LEFT);
      splitMain.setFirstComponent(panContacts);
      splitMain.setSecondComponent(panDetails);

      setTitle("Automatic Link Editors");
      setLayout(blThis);
      add(splitMain, UIBorderLayout.CENTER);
   }
```

```
   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   // Interface Implementation
   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   /**
    * {@inheritDoc}
    */
   public void save() throws ModelException
   {
      dataSource.saveAllDataBooks();
   }

   /**
    * {@inheritDoc}
    */
   public void reload() throws ModelException
   {
      dataSource.reloadAllDataBooks();
   }

   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   // User-Defined Methods
   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   /**
    * Saves the image to the contact.
    *
    * @param pFileHandle the file.
    * @throws Throwable if an error occures.
    */
   public void storeFile(IFileHandle pFileHandle) throws Throwable
   {
      String sFormat =
FileUtil.getExtension(pFileHandle.getFileName().toLowerCase());

      if ("png".equals(sFormat)
         || "jpg".equals(sFormat)
         || "gif".equals(sFormat))
      {
         ByteArrayOutputStream stream = new ByteArrayOutputStream();

         ImageUtil.createScaledImage(pFileHandle.getInputStream(),
                                     140,
                                     185,
                                          true,
                                          stream,
                                          sFormat);

         stream.close();
```

```
         rdbContacts.setValue("FILENAME", pFileHandle.getFileName());
         rdbContacts.setValue("IMAGE", stream.toByteArray());

         try
         {
            rdbContacts.saveSelectedRow();
         }
         catch (Exception pException)
         {
            // Silent Save of current row.
         }
      }
      else
      {
         throw new IOException("Image format '" + sFormat + "' not
supported. Use 'png', 'jpg' or 'gif'!");
      }

   }

   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   // Actions
   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   /**
    * Starts the image upload.
    *
    * @throws Throwable if an error occures.
    */
   public void doUpload() throws Throwable
   {
      if (rdbContacts.getSelectedRow() >= 0)
      {
         getApplication().getLauncher().getFileHandle(this,
"storeFile");
      }
   }

   /**
    * Opens the educations frame.
    *
    * @throws Throwable if the educations frame can not be opened
    */
   public void doNewEducations() throws Throwable
   {
      getApplication().openFrame(EducationsFrame.class);
   }

   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   // Interface implementation
   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
   /**
    * Searches the contacts with the search text.
    *
    * @throws ModelException if the search fails
    */
   public void doFilter() throws ModelException
   {
      String suche = (String)drSearch.getValue("SEARCH");
      if (suche == null)
      {
         rdbContacts.setFilter(null);
      }
      else
      {
         ICondition filter = new LikeIgnoreCase("FIRSTNAME", "*" +
suche + "*").or(
                        new LikeIgnoreCase("LASTNAME", "*" + suche +
"*").or(
                        new LikeIgnoreCase("STREET", "*" + suche +
"*").or(
                        new LikeIgnoreCase("TOWN", "*" + suche +
"*"))));
         rdbContacts.setFilter(filter);
      }
   }


}   // ContactsAutoFrame
```

The server code:

ContactsAuto.java

```
/*
 * Copyright 2009 SIB Visions GmbH
 *
 * Licensed under the Apache License, Version 2.0 (the "License"); you
may not
 * use this file except in compliance with the License. You may obtain
a copy of
 * the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT
 * WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See
the
```

```
 * License for the specific language governing permissions and
limitations under
 * the License.
 *
 *
 * History
 *
 * 27.11.2009 - [HM] - creation
 */
package com.sibvisions.apps.showcase.frames;

import com.sibvisions.apps.showcase.Session;
import com.sibvisions.rad.persist.jdbc.DBStorage;

/**
 * The <code>ContactsAuto</code> class is the life-cycle object for
<code>ContactsAutoFrame</code>.
 *
 * @author Martin Handsteiner
 */
public class ContactsAuto extends Session
{
   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   // Members
   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   /** storage for contacts. */
   private DBStorage dbsContacts = null;

   /** Storage for contacts educations. */
   private DBStorage dbsContEduc = null;

   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   // User-defined methods
   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   /**
    * Returns the contacts storage.
    *
    * @return the Contacts storage
    * @throws Exception if the initialization throws an error
    */
   public DBStorage getContacts() throws Exception
   {
      if (dbsContacts == null)
      {
         dbsContacts = new DBStorage();
         dbsContacts.setDBAccess(getDBAccess());
         dbsContacts.setWritebackTable("CONTACTS");
         dbsContacts.setAutoLinkReference(true);
         dbsContacts.open();
```

```
      }

      return dbsContacts;
   }

   /**
    * Returns the contacts educations storage.
    *
    * @return the contacts storage
    * @throws Exception if the initialization throws an error
    */
   public DBStorage getContEduc() throws Exception
   {
      if (dbsContEduc == null)
      {
         dbsContEduc = new DBStorage();
         dbsContEduc.setDBAccess(getDBAccess());
         dbsContEduc.setWritebackTable("CONT_EDUC");
         dbsContEduc.setAutoLinkReference(true);
         dbsContEduc.open();
      }

      return dbsContEduc;
   }

}   // ContactsAuto
```

The database script:

create.sql

```
--------------------------------------------------------------------------------
--------
-- Tables
--------------------------------------------------------------------------------
--------

CREATE TABLE SALUTATIONS
(
  ID         INTEGER IDENTITY,
  SALUTATION VARCHAR(200),
  CONSTRAINT SALU_UK UNIQUE (SALUTATION)
)

CREATE TABLE ACADEMICTITLES
(
  ID             INTEGER IDENTITY,
  ACADEMIC_TITLE VARCHAR(200) NOT NULL,
  CONSTRAINT ACTI_UK UNIQUE (ACADEMIC_TITLE)
)
```

```
CREATE TABLE COUNTRIES
(
  ID      INTEGER IDENTITY,
  COUNTRY VARCHAR(200) NOT NULL,
  EU      CHAR(1) DEFAULT 'N' NOT NULL,
  CONSTRAINT CTRY_UK UNIQUE (COUNTRY)
)

CREATE TABLE STATES
(
  ID      INTEGER IDENTITY,
  CTRY_ID INTEGER NOT NULL,
  STATE   VARCHAR(200) NOT NULL,
  CONSTRAINT STAT_UK UNIQUE (STATE),
  CONSTRAINT STAT_CTRY_ID_FK FOREIGN KEY (CTRY_ID) REFERENCES COUNTRIES
(ID)
)

CREATE TABLE DISTRICTS
(
  ID       INTEGER IDENTITY,
  STAT_ID  INTEGER NOT NULL,
  DISTRICT VARCHAR(200),
  CONSTRAINT DIST_UK UNIQUE (DISTRICT),
  CONSTRAINT DIST_STAT_ID_FK FOREIGN KEY (STAT_ID) REFERENCES STATES
(ID)
)

CREATE TABLE HEALTHINSURANCES
(
  ID               INTEGER IDENTITY,
  HEALTH_INSURANCE VARCHAR(200) NOT NULL,
  CONSTRAINT HEIN_UK UNIQUE (HEALTH_INSURANCE)
)

CREATE TABLE EDUCATIONS
(
  ID        INTEGER IDENTITY,
  EDUCATION VARCHAR(200),
  CONSTRAINT EDUC_UK UNIQUE (EDUCATION)
)

CREATE TABLE CONTACTS
(
  ID            INTEGER IDENTITY,
  SALU_ID       INTEGER,
  ACTI_ID       INTEGER,
  FIRSTNAME     VARCHAR(200) NOT NULL,
  LASTNAME      VARCHAR(200) NOT NULL,
  STREET        VARCHAR(200),
```

```
  NR            VARCHAR(200),
  ZIP           VARCHAR(4),
  TOWN          VARCHAR(200),
  CTRY_ID       INTEGER,
  BIRTHDAY      DATE,
  SOCIALSECNR   DECIMAL(4),
  HEIN_ID       INTEGER,
  FILENAME      VARCHAR(200),
  IMAGE         BINARY,
  CONSTRAINT CONT_SALU_ID_FK FOREIGN KEY (SALU_ID) REFERENCES
SALUTATIONS (ID),
  CONSTRAINT CONT_CTRY_ID_FK FOREIGN KEY (CTRY_ID) REFERENCES COUNTRIES
(ID),
  CONSTRAINT CONT_ACTI_ID_FK FOREIGN KEY (ACTI_ID) REFERENCES
ACADEMICTITLES (ID),
  CONSTRAINT CONT_HEIN_ID_FK FOREIGN KEY (HEIN_ID) REFERENCES
HEALTHINSURANCES (ID)
)

CREATE TABLE CONT_EDUC
(
  ID      INTEGER IDENTITY,
  CONT_ID INTEGER NOT NULL,
  EDUC_ID INTEGER NOT NULL,
  CONSTRAINT COED_UK UNIQUE (CONT_ID, EDUC_ID),
  CONSTRAINT COED_CONT_ID_FK FOREIGN KEY (CONT_ID) REFERENCES CONTACTS
(ID),
  CONSTRAINT COED_EDUC_ID_FK FOREIGN KEY (EDUC_ID) REFERENCES
EDUCATIONS (ID)
)

CREATE TABLE FOLDERS
(
  ID        INTEGER IDENTITY,
  FOLDER    VARCHAR(256) NOT NULL,
  FOLD_ID   INTEGER,
  CONSTRAINT FOLD_UK UNIQUE (FOLD_ID, FOLDER),
  CONSTRAINT FOLD_FOLD_ID_FK FOREIGN KEY (FOLD_ID) REFERENCES FOLDERS
(ID) ON DELETE CASCADE
)

CREATE TABLE FILES
(
  ID         INTEGER IDENTITY,
  TYPE       VARCHAR(50) NOT NULL,
  FILENAME   VARCHAR(256) NOT NULL,
  FILESIZE   INTEGER NOT NULL,
  CREATED    DATE NOT NULL,
  CREATED_BY VARCHAR(64),
  CHANGED    DATE NOT NULL,
  CHANGED_BY VARCHAR(64),
```

```
  FOLD_ID    INTEGER,
  CONSTRAINT FILE_UK UNIQUE (FOLD_ID, FILENAME),
  CONSTRAINT FILE_FOLD_ID_FK FOREIGN KEY (FOLD_ID) REFERENCES FOLDERS
(ID) ON DELETE CASCADE
)

--------------------------------------------------------------------------
--------
-- Views
--------------------------------------------------------------------------
--------

CREATE VIEW V_COUNTRIES AS
SELECT c.ID
      ,c.COUNTRY
      ,c.EU
  FROM COUNTRIES c
 ORDER BY c.COUNTRY

CREATE VIEW V_STATES AS
SELECT s.ID
      ,s.CTRY_ID
      ,s.STATE
  FROM STATES s
 ORDER BY s.STATE

CREATE VIEW V_DISTRICTS AS
SELECT d.ID
      ,d.STAT_ID
      ,d.DISTRICT
  FROM DISTRICTS d
 ORDER BY d.DISTRICT

CREATE VIEW V_EDUCATIONS AS
SELECT e.ID
      ,e.EDUCATION
  FROM EDUCATIONS e
 ORDER BY e.EDUCATION

CREATE VIEW V_CONTACTS AS
SELECT C.ID
      ,C.SALU_ID
      ,(SELECT S.SALUTATION FROM SALUTATIONS S WHERE S.ID = C.SALU_ID)
SALUTATION
      ,C.ACTI_ID
      ,(SELECT T.ACADEMIC_TITLE FROM ACADEMICTITLES T WHERE T.ID =
C.ACTI_ID) ACADEMIC_TITLE
      ,C.FIRSTNAME
      ,C.LASTNAME
      ,C.STREET
      ,C.NR
```

```
      ,C.ZIP
      ,C.TOWN
      ,C.CTRY_ID
      ,(SELECT CO.COUNTRY FROM COUNTRIES CO WHERE CO.ID = C.CTRY_ID)
COUNTRY
      ,C.BIRTHDAY
      ,C.SOCIALSECNR
      ,C.HEIN_ID
      ,(SELECT I.HEALTH_INSURANCE FROM HEALTHINSURANCES I WHERE I.ID =
C.HEIN_ID) HEALTH_INSURANCE
      ,C.FILENAME
      ,C.IMAGE
  FROM CONTACTS C

CREATE VIEW V_CONT_EDUC AS
SELECT CE.ID
      ,CE.CONT_ID
      ,CE.EDUC_ID
      ,(SELECT E.EDUCATION FROM EDUCATIONS E WHERE E.ID = CE.EDUC_ID)
EDUCATION
  FROM CONT_EDUC CE
```

Encrypt Passwords

We don't encrypt passwords on client-side nor in the database. We use our middleware for that. It's super easy with server-side triggers/events:

Add the method:

```
/**
 * Encrypts a password, if password is changed.
 *
 * @param pEvent the storage event
 * @throws Exception if encryption or data change fails
 */
public void doEncryptPwd(StorageEvent pEvent) throws Exception
{
   IBean bn = pEvent.getNew();

   String sNew = (String)bn.get("PASSWORD");
   String sOld;

   IBean bnOld = pEvent.getOld();

   if (bnOld != null)
   {
      sOld = (String)bnOld.get("PASSWORD");
   }
```

```
   else
   {
      sOld = null;
   }

   if (!CommonUtil.equals(sOld, sNew))
   {
      //use the configuration of the selected application!
      bn.put("PASSWORD", AbstractSecurityManager.getEncryptedPassword(
                           SessionContext.getCurrentSession().getConfig(),
sNew));
   }
}
```

to your lifecycle object, e.g., Session.java.

Add an event to your storage

```
//example storage
dbsUser = new DBStorage();
dbsUser.setDBAccess(getDBAccess());
dbsUser.setFromClause("USER");
dbsUser.open();

dbsUser.eventBeforeInsert().addListener(this, "doEncryptPwd");
dbsUser.eventBeforeUpdate().addListener(this, "doEncryptPwd");
```

As last step, you need the password algorithm in your config.xml:

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

<application>
  <securitymanager>
    <passwordalgorithm>SHA</passwordalgorithm>
  </securitymanager>
  ...
</application>
```

Choose one of the following algorithm: MD2, MD4, MD5, SHA, SHA-256, SHA-384, SHA-512.

Fetch Data With DBStorage

Sometimes you want to use DBAccess and DBStorage without JVx server and lifecycle objects. This is very easy because JVx was designed as library. Simply use DBStorage and DBAccess in your Servlets, applications, etc.

The following example uses DBAccess and DBStorage to check if a user exists in the database and if an email address doesn't exist.

```
dba = DBAccess.getDBAccess("jdbc:oracle:thin:@localhost:1521:xe");
```

```
dba.setUsername("user");
dba.setPassword("password");
//also possible, but needs rad/apps/myapp/config.xml
//DBAccess dba = DBAccess.getDBAccess(DBSecurityManager.getCredentials(
//
Configuration.getApplicationZone("myapp").getConfig()));
dba.open();
dba.getConnection().setAutoCommit(false);

DBStorage dbsUser = new DBStorage();
dbsUser.setDBAccess(dba);
dbsUser.setWritebackTable("USERS");
dbsUser.open();
```

If you prefer POJOs:

```
User user = dbsUser.createPOJO(User.class, bnUser);
```

50% Width With FormLayout

If you have two components, e.g., GroupPanels, and if you want that each group is 50% of the parent width, you could use following code:

```
UIFormLayout layout = new UIFormLayout();

IConstraints center = layout.getHCenterConstraints(0, 5, -1, 5);

UILabel label = new UILabel();
label.setPreferredSize(5, 0);

panel.add(label, center);
panel.add(gpanLeft, layout.getConstraints(center.getTopAnchor(),
layout.createAnchor(layout.getLeftMarginAnchor(), 0),
                                  center.getBottomAnchor(),
                                  layout.createAnchor(center.getLeftAnchor(),
0)));
panel.add(gpanRight, layout.getConstraints(center.getTopAnchor(),
layout.createAnchor(center.getRightAnchor(), 0),
                                  center.getBottomAnchor(),
layout.createAnchor(layout.getRightMarginAnchor(), 0)));
```

Search Column in Condition

If you create your own IStorage impelementation, it could be useful to know how to find the value for a specific column:

```
public static Object getEqualsValue(ICondition pFilter, String pColumn)
{
```

```
        if (pFilter instanceof OperatorCondition)
        {
                for (ICondition cond :
((OperatorCondition)pFilter).getConditions())
                {
                        if (cond instanceof Equals)
                        {
                                if
(pColumn.equals(((Equals)cond).getColumnName()))
                                {
                                        return ((Equals)cond).getValue();
                                }
                        }
                }
        }
        else if (pFilter instanceof Equals)
        {
                if (pColumn.equals(((Equals)pFilter).getColumnName()))
                {
                        return ((Equals)pFilter).getValue();
                }
        }

        return null;
}
```

The above method tries to find the first equals condition with the given column name.

The method doesn't work with recursion because usually this is not necessary. If you want to know the different condition types, simply check `toString()` of `jvx.rad.model.condition.BaseCondition`.

Connection Property Changed Listener

It's easy to listen on connection property changed events:

```
MasterConnection appcon = new MasterConnection(createConnection());
appcon.addPropertyChangedListener("Application.mode", new
IConnectionPropertyChangedListener()
{
    public void propertyChanged(PropertyEvent pEvent)
    {
        liProps.add(pEvent.getPropertyName() + " = " +
pEvent.getNewValue());
    }
});
```

The above listener will be invoked whenever the connection property **Application.mode** will be changed.

It's also possible to listen on all properties, with following snippet:

```
MasterConnection appcon = new MasterConnection(createConnection());
appcon.addPropertyChangedListener(null, new
IConnectionPropertyChangedListener()
{
    public void propertyChanged(PropertyEvent pEvent)
    {
        liProps.add(pEvent.getPropertyName() + " = " +
pEvent.getNewValue());
    }
});
```

Test UI With Simple JUnit Test

Sometimes you want a JUnit test case for a specific UI feature. If you need an application frame for your test, you could use following implementation:

A simple Test Launcher:

SimpleTestLauncher.java

```
/*
 * Copyright 2016 SIB Visions GmbH
 *
 * Licensed under the Apache License, Version 2.0 (the "License"); you
may not
 * use this file except in compliance with the License. You may obtain
a copy of
 * the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT
 * WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See
the
 * License for the specific language governing permissions and
limitations under
 * the License.
 *
 *
 * History
 *
 * 11.08.2016 - [JR] - creation
 */
package jvx.rad.application;

import jvx.rad.ui.IComponent;
```

```
import jvx.rad.ui.container.IDesktopPanel;
import jvx.rad.util.IRunnable;
import javax.swing.JComponent;
import javax.swing.SwingUtilities;

import com.sibvisions.rad.application.Application;
import com.sibvisions.rad.ui.swing.impl.SwingApplication;

/**
 * The <code>SimpleTestLauncher</code> starts a {@link
SwingApplication} and shows
 * a specific content in the application pane.
 *
 * @author René Jahn
 */
public class SimpleTestLauncher
{
    //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    // Initialization
    //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    /**
     * Creates a new instance of <code>SimpleTestLauncher</code>.
     *
     * @param pComponent the component to show
     */
    public SimpleTestLauncher(final IComponent pComponent)
    {
        final SwingApplication app = new SwingApplication();

        app.startup(SimpleTestApplication.class.getName(), null, null);
        app.setSystemExitOnDispose(false);
        app.eventWindowClosed().addListener(new IRunnable()
        {
            public void run()
            {
                synchronized (SimpleTestLauncher.this)
                {
                    SimpleTestLauncher.this.notify();
                }
            }
        });

        SwingUtilities.invokeLater(new Runnable()
        {
            public void run()
            {
                IDesktopPanel desktop =
((Application)app.getApplication()).getDesktopPane();
                desktop.removeAll();
                desktop.add(pComponent);
```

```
                //because test was created before LaF was set!
SwingUtilities.updateComponentTreeUI((JComponent)pComponent.getResource
());
            }
        });

        synchronized(this)
        {
            try
            {
                wait();
            }
            catch (InterruptedException ie)
            {
                //ignore
            }
        }
    }

}   // SimpleTestLauncher
```

This class wrapps application creation, because we need to set the UI component after the application was started!

The second class is our Application:

SimpleTestApplication.java

```
/*
 * Copyright 2016 SIB Visions GmbH
 *
 * Licensed under the Apache License, Version 2.0 (the "License"); you
may not
 * use this file except in compliance with the License. You may obtain
a copy of
 * the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT
 * WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See
the
 * License for the specific language governing permissions and
limitations under
 * the License.
 *
 *
```

```
 * History
 *
 * 11.08.2016 - [JR] - creation
 */
package jvx.rad.application;

import jvx.rad.application.genui.UILauncher;
import jvx.rad.remote.IConnection;

import com.sibvisions.rad.application.Application;

/**
 * The <code>SimpleTestApplication</code> is an always connected
application
 * implementation, for test cases.
 *
 * @author René Jahn
 */
public class SimpleTestApplication extends Application
{
    //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    // Initialization
    //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    /**
     * Creates a new instance of <code>SimpleTestApplication</code>.
     *
     * @param pLauncher the launcher
     */
    public SimpleTestApplication(UILauncher pLauncher)
    {
        super(pLauncher);
    }

    //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    // Overwritten methods
    //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    @Override
    protected String getApplicationName()
    {
        return "Test Application";
    }

    @Override
    protected IConnection createConnection() throws Exception
    {
        return null;
    }

    @Override
```

```
    public void notifyVisible()
    {
    }

    @Override
    protected void afterLogin()
    {
    }

}   // SimpleTestApplication
```

The application removes login screen and you can use it without manual tasks.

Finally, our JUnit test case:

TestActivity.java

```
/*
 * Copyright 2016 SIB Visions GmbH
 *
 * Licensed under the Apache License, Version 2.0 (the "License"); you
may not
 * use this file except in compliance with the License. You may obtain
a copy of
 * the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT
 * WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See
the
 * License for the specific language governing permissions and
limitations under
 * the License.
 *
 *
 * History
 *
 * 11.08.2016 - [JR] - creation
 */
package com.sibvisions.forum;

import jvx.rad.application.SimpleTestLauncher;
import jvx.rad.genui.UIFactoryManager;
import jvx.rad.genui.container.UIPanel;
import jvx.rad.genui.layout.UIBorderLayout;
import jvx.rad.model.reference.ReferenceDefinition;
```

```
import org.junit.BeforeClass;
import org.junit.Test;

import com.sibvisions.apps.components.NavigationTable;
import com.sibvisions.rad.persist.StorageDataBook;
import com.sibvisions.rad.persist.jdbc.DBAccess;
import com.sibvisions.rad.persist.jdbc.DBStorage;
import com.sibvisions.rad.ui.swing.impl.SwingFactory;

public class TestActivity extends UIPanel
{
    //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    // Initialization
    //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    /**
     * Initializes the unit test.
     *
     * @throws Exception if initialization fails
     */
    @BeforeClass
    public static void beforeClass()
    {
        UIFactoryManager.getFactoryInstance(SwingFactory.class);
    }

    //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    // Test methods
    //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    /**
     * Tests application start with a custom UI.
     *
     * @throws Exception if app start fails
     */
    @Test
    public void testStartApplication() throws Exception
    {
        DBAccess dba =
DBAccess.getDBAccess("jdbc:oracle:thin:@localhost:1521:XE",
                                             "username", "password");
        dba.open();

        DBStorage dbsContract = new DBStorage();
        dbsContract.setWritebackTable("CONTRACT");
        dbsContract.setDBAccess(dba);
        dbsContract.open();

        DBStorage dbsActivity = new DBStorage();
        dbsActivity.setWritebackTable("ACTIVITY");
        dbsActivity.setDBAccess(dba);
```

```
        dbsActivity.open();

        StorageDataBook sdbContract = new StorageDataBook(dbsContract);
        sdbContract.open();
        sdbContract.fetchAll();
        sdbContract.setSelectedRow(0);

        StorageDataBook sdbActivity = new StorageDataBook(sdbContract,
dbsActivity);
        sdbActivity.setMasterReference(new ReferenceDefinition(new
String[] {"CONT_ID"},
                                       sdbContract,
                                       new String[] {"ID"}));
        sdbActivity.open();

        setLayout(new UIBorderLayout());

        NavigationTable nav = new NavigationTable();
        nav.setDataBook(sdbActivity);

        add(nav);

        new SimpleTestLauncher(this);
    }

}   // TestActivity
```

The result:



The database model (Oracle):



The database objects (Oracle):

create.sql

```
CREATE TABLE CONTRACT
(
  id    NUMBER(16) NOT NULL,
  title VARCHAR2(1000),
  state NUMBER(1)
)
;
comment ON COLUMN CONTRACT.id
  IS 'PK';
comment ON COLUMN CONTRACT.title
  IS 'Contract title';
comment ON COLUMN CONTRACT.state
```

```
  IS 'Contract state';
ALTER TABLE CONTRACT
  ADD CONSTRAINT CONT_PK PRIMARY KEY (ID);

CREATE TABLE ACTIVITY
(
  id                  NUMBER(16) NOT NULL,
  cont_id             NUMBER(16),
  name                VARCHAR2(1000),
  cost                NUMBER(10,2),
  valid_from          DATE,
  valid_to            DATE,
  responsible_user_id NUMBER(16),
  validated           CHAR(1) DEFAULT 'Y'
)
;
comment ON COLUMN ACTIVITY.id
  IS 'PK';
comment ON COLUMN ACTIVITY.cont_id
  IS 'Contract FK';
comment ON COLUMN ACTIVITY.name
  IS 'Activity name';
comment ON COLUMN ACTIVITY.cost
  IS 'Activity costs';
comment ON COLUMN ACTIVITY.valid_from
  IS 'Valid from';
comment ON COLUMN ACTIVITY.valid_to
  IS 'Valid to';
comment ON COLUMN ACTIVITY.responsible_user_id
  IS 'Responsible User FK';
comment ON COLUMN ACTIVITY.validated
  IS 'Whether the activity is valid';
ALTER TABLE ACTIVITY
  ADD CONSTRAINT ACTI_PK PRIMARY KEY (ID);
ALTER TABLE ACTIVITY
  ADD CONSTRAINT ACTI_CONT_FK FOREIGN KEY (CONT_ID)
  REFERENCES CONTRACT (ID);
ALTER TABLE ACTIVITY
  ADD CONSTRAINT ACTI_RESP_USER_FK FOREIGN KEY (RESPONSIBLE_USER_ID)
  REFERENCES USERS (ID);


CREATE OR REPLACE TRIGGER TR_ACTIVITY_BR_IU
  BEFORE INSERT OR UPDATE ON activity
  FOR each ROW
BEGIN

  IF (:NEW.validated = 'N') THEN

    IF (:NEW.valid_from IS NULL) THEN
      raise_application_error(-20000, 'Valid from can''t be null!');
```

```
    END IF;

  END IF;

END TR_ACTIVITY_BR_IU;
/

CREATE OR REPLACE TRIGGER TR_CONTRACT_BR_IU
  BEFORE INSERT OR UPDATE ON contract
  FOR each ROW
BEGIN

  IF (:NEW.state = 4) THEN

    UPDATE activity
       SET validated = 'N'
     WHERE cont_id = id
       AND validated = 'Y';

  END IF;

END TR_CONTRACT_BR_IU;
/


ALTER TABLE CONTRACT disable ALL triggers;
ALTER TABLE ACTIVITY disable ALL triggers;
ALTER TABLE ACTIVITY disable CONSTRAINT ACTI_CONT_FK;
ALTER TABLE ACTIVITY disable CONSTRAINT ACTI_RESP_USER_FK;
INSERT INTO CONTRACT (id, title, state)
VALUES (1, 'Air', 1);
commit;
INSERT INTO ACTIVITY (id, cont_id, name, cost, valid_from, valid_to,
responsible_user_id, validated)
VALUES (1, 1, 'Part 1', .2, NULL, NULL, NULL, 'Y');
commit;
ALTER TABLE ACTIVITY enable CONSTRAINT ACTI_CONT_FK;
ALTER TABLE ACTIVITY enable CONSTRAINT ACTI_RESP_USER_FK;
ALTER TABLE CONTRACT enable ALL triggers;
ALTER TABLE ACTIVITY enable ALL triggers;
```

Authentication/User Logging

Some applications need detailed information about successful or failed user logins. If you have such requirement, you could create a custom security manager to solve the problem. The security manager will be used for user authentication, so it's the right place to start.

We use the database security manager as base class:

LoggingSecurityManager.java

```
package com.sibvisions.apps.vaadin.web;

public class LoggingSecurityManager extends DBSecurityManager
{
    private PreparedStatement psLockedUsers;

    @Override
    protected boolean isPasswordValid(ISession pSession, String
pPassword) throws Exception
    {
        // additional check: locked user

        ResultSet resultSet = null;

        try
        {
            psLockedUsers.clearParameters();
            psLockedUsers.setString(1, pSession.getUserName());
            psLockedUsers.execute();

            resultSet = psLockedUsers.getResultSet();

            if (resultSet.next())
            {
                throw new SilentAbortException();
            }
        }
        finally
        {
            CommonUtil.close(resultSet);
        }

        return super.isPasswordValid(pSession, pPassword);
    }

    @Override
    public synchronized void validateAuthentication(ISession pSession)
throws Exception
    {
        String result = null;

        Throwable error = null;

        try
        {
            super.validateAuthentication(pSession);

            result = Constants.RESULT_OK;
        }
```

```
        catch (SilentAbortException sae)
        {
            result = Constants.RESULT_IGNORE;
            error = sae;

            throw sae;
        }
        catch (SecurityException se)
        {
            result = Constants.RESULT_DENIED;
            error = se;

            throw se;
        }
        catch (Exception ex)
        {
            result = Constants.RESULT_ERROR;
            error = ex;

            error(ex);

            throw ex;
        }
        finally
        {
            try
            {
                pSession.callAction("dbLog", Constants.TYPE_LOGIN,
error, result);
            }
            catch (Throwable thr)
            {
                error(thr);
            }
        }
    }

    @Override
    public synchronized void logout(ISession pSession)
    {
        String result = null;
        Throwable error = null;

        try
        {
            super.logout(pSession);

            if
(Boolean.parseBoolean((String)pSession.getProperty("userlogout")))
            {
                result = Constants.RESULT_OK;
```

```
            }
            else
            {
                result = Constants.RESULT_EXPIRED;
            }
        }
        catch (Exception ex)
        {
            result = Constants.RESULT_ERROR;
            error = ex;

            error(ex);
        }
        finally
        {
            try
            {
                pSession.callAction("dbLog", Constants.TYPE_LOGOUT,
error, result);
            }
            catch (Throwable eLog)
            {
                error(eLog);
            }
        }
    }

    @Override
    protected void initStatements(Connection pConnection) throws
Exception
    {
        super.initStatements(pConnection);

        psLockedUsers = prepareStatement(pConnection, "select * from
LOCKS where USERNAME = ?");
    }

}     // LoggingSecurityManager
```

The clue is that the log method **dbLog** was implemented in the session LCO of our application because we re-use the business logic in our security manager. Here's the missing method:

```
public void dbLog(int pAction, Throwable pError, String pStatus)
{
    ILogger log = LoggerFactory.getInstance(getClass());

    log.info("Log action ", pAction, pError, pStatus);

    try
    {
```

```
        getDBAccess().executeProcedure("log",
                                       pAction,
SessionContext.getCurrentSession().getUserName(),
                                       new
Timestamp(System.currentTimeMillis()));
    }
    catch (Exception ex)
    {
        log.error(ex);
    }
}
```

Detect Production Mode

If you use the DBSecurityManager, it's possible to set the environment. The environment is a marker for the database connection, e.g.:

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

<application>
  <securitymanager>
    <class>com.sibvisions.rad.server.security.DBSecurityManager</class>
    <environment>prod</environment>
  </securitymanager>
  <datasource>
    <db name="default">
      <url>jdbc:oracle:thin:@dbserver:1521:XE</url>
      <username>test</username>
      <username_prod>prod</username_prod>
      <password>test</password>
      <password_prod>prod</password_prod>
    </db>
  </datasource>
</application>
```

The marker will be used as postfix for the detection of datasource credentials. In the example, the postfix is **prod** and the **username_prod** will be used instead of **username**. If not tag with environment postfix is available, the standard tag will be used.

If you want to use the environment in your source code, simply call:

```
String env =
SessionContext.getCurrentSessionConfig().getProperty("/application/securitym
anager/environment");
```

From:
http://doc.sibvisions.com/ - **Documentation**

Permanent link:
http://doc.sibvisions.com/jvx/code_snippets

Last update: **2024/11/18 10:16**