Web Development with Java EE

JavaBeans and JDBC

Federico Pecora

AASS Mobile Robotics Lab, Teknik

Room T2222

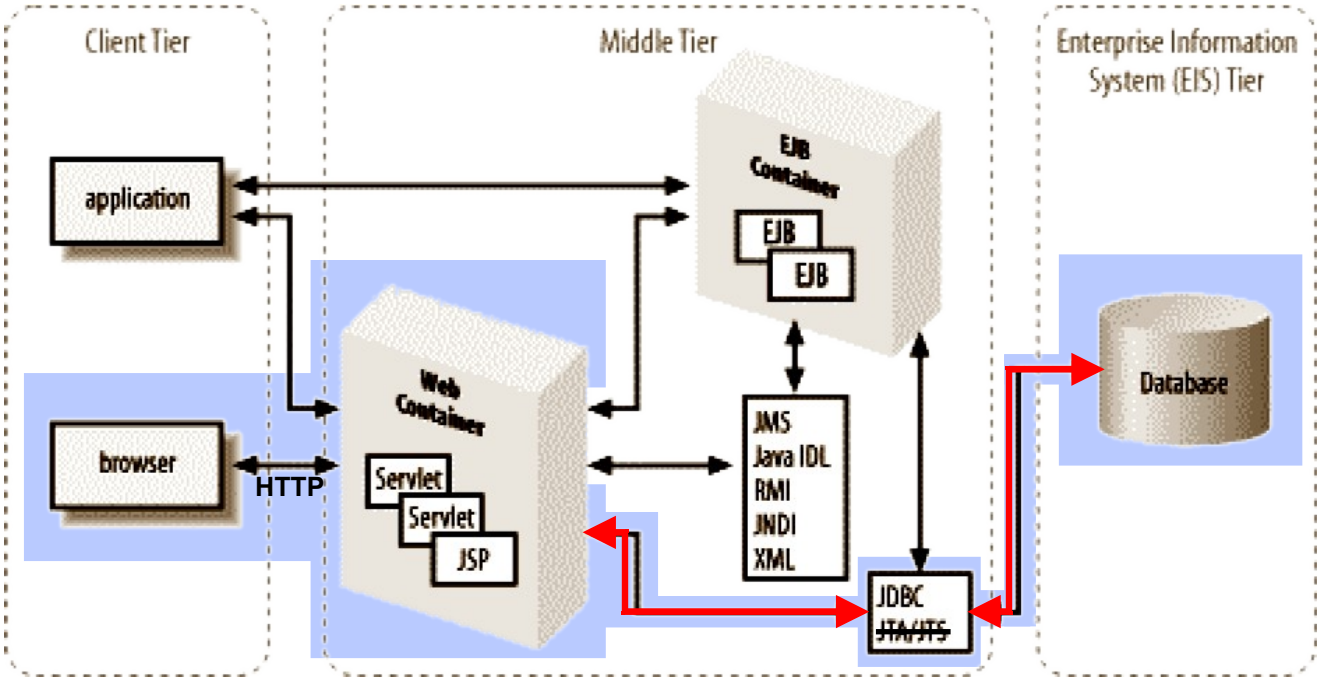
fpa@aass.oru.se

Contents

Introduction

1 Introduction

- Most web applications access databases
  - E.g., to store information that needs to survive a server restart, data that is too complex for plain text files, ...

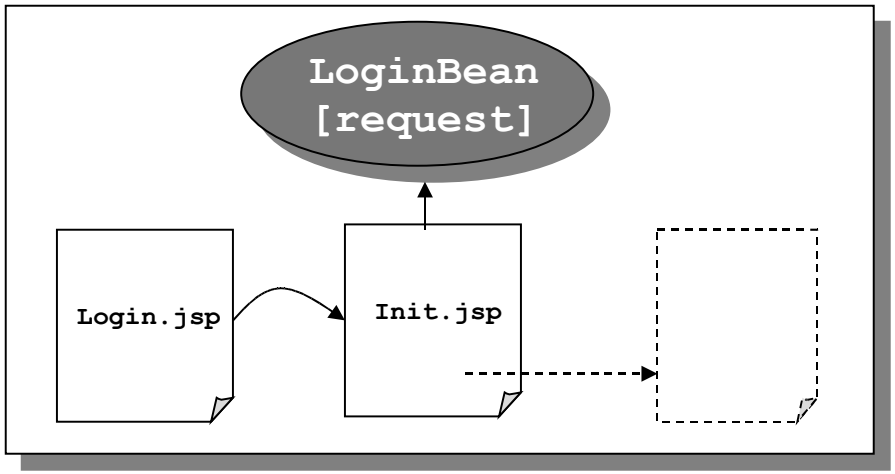


→ Contents

JavaBeans

2 JavaBeans

- What are JavaBeans?
  - classes that follow the JavaBeans programming conventions
    - other programs can easily discover information about JavaBeans
    - other programs can easily add JavaBean compliant code
- How are JavaBeans typically employed?
  - as a container for dynamic content
  - data validation
  - application logic
  - encapsulation (one place)



2 JavaBeans

- What makes a Java class a JavaBean?
  - it must have an empty constructor (default constructor)
  - it does *not* have public instance variables
  - it offers get/set methods to access values
    - `setProperty` / `isProperty` for boolean data types
    - `setProperty` / `getProperty` for all other data types
    - only `isProperty` / `getProperty` for read-only data-types
    - ! property names always start with a lowercase letter
    - ! public accessor methods with a standardised naming
  - typically a JavaBean is serializable

2 JavaBeans

```
package beanexamples;

import java.beans.*;
import java.io.Serializable;

public class LoginBean implements java.io.Serializable {

  public LoginBean() { }

  public String getUsername() { return username; }
  public void setUsername(String username) {
    if(username != null) this.username = username;
  }

  public String getPassword() { return password; }
  public void setPassword(String password) {
    if(password != null) this.password = password;
  }

  public boolean isMale() { return male; }
  public void setMale(boolean male) { this.male = male; }

  private String username;
  private String password;
  private boolean male;

}
```

beanexamples/LoginBean.java

2 JavaBeans

- JSP Integration – Loading a Bean
  - `<jsp:useBean`
    `id="loginInfo"`
    `class="beanexamples.LoginBean" />`
  - specify the JavaBean class (**`class`**)
  - specify an ID that refers to the JavaBean (**`id`**)
  - similar to
    `<%` **`beanexamples.LoginBean`** **`loginInfo`** `=`
    `new beanexamples.LoginBean()` `%>`

2 JavaBeans

- JSP Integration – Loading a Bean
  - `<jsp:useBean`

    `id="loginInfo"`

    `class="beanexamples.LoginBean"`

    `scope="session" />`
  - specify a scope for the JavaBean (**`scope`**)
  - **`application`**, **`session`**, **`request`**, **`page`** (default)
    - associates bean to **`ServletContext`**, **`HttpSession`**, …
  - uses an existing bean with the same **`id`** and **`scope`**

2 JavaBeans

- JSP Integration – Loading a Bean
  -
```
<jsp:useBean
   id="loginInfo"
   class="beanexamples.LoginBean"
   scope="session"
   type="Object" />
```
  - `type` specifies the superclass
  -
```
<% Object loginInfo =
     new beanexamples.LoginBean() %>
```

2 JavaBeans

JSP Integration – Loading a Bean

```
<jsp:useBean
   id="loginInfo"
   beanName="beanexamples.LoginBean"
   scope="session" />
```

- **beanName** can refer to
  - a class
  - a file with the serialized bean object

2 JavaBeans

JSP Integration – Accessing Bean Properties

```
<jsp:useBean
    id="loginInfo"
    beanName="beanexamples.LoginBean"
    scope="session" />
<jsp:getProperty
    name="loginInfo"
    property="username" />
<%= loginInfo.getUsername() %>
```

2 JavaBeans

JSP Integration – Setting Bean Properties (1)

```
<jsp:useBean
    id="loginInfo"
    beanName="beanexamples.LoginBean"
    scope="session" />
<jsp:setProperty
    name="loginInfo"
    property="username"
    value="Its Me" />
<%= loginInfo.setUsername("Its Me") %>
```

constant property value

2 JavaBeans

JSP Integration – Setting Bean Properties (2)

```
<jsp:setProperty
    name="loginInfo"
    property="username"
    value="<%=request.getParameter("name")
%>" />
```

variable String parameter

- **value** and **name** can be request-time expressions

2 JavaBeans

JSP Integration – Setting Bean Properties (2a)

```
<jsp:useBean id="loginInfo" class="beanexamples.LoginBean" scope="request" />
<jsp:setProperty name="loginInfo" property="username"
                 value="<%=request.getParameter("name")%>" />
<jsp:setProperty name="loginInfo" property="password"
                 value="<%=request.getParameter("password")%>" />

<html>
  <head><title>JSP Example 10 - First Bean</title></head>
<body>

  <h1>JSP Example 10 - First Bean</h1>

  <p>Username: <jsp:getProperty name="loginInfo" property="username"/></p>
  <p>Password: <jsp:getProperty name="loginInfo" property="password"/></p>
  <p>
   If you want to change name or password to "My Name" or "My Password", call
   <a href="<%=request.getRequestURI()%>?name=My+Name&password=My+Password">
     <%=request.getRequestURI()%>?name=My+Name&password=My+Password
   </a>.
  </p>

</body></html>
```

2 JavaBeans

JSP Integration – Setting Bean Properties (2b)

```
<jsp:useBean id="loginInfo" class="beanexamples.LoginBean" scope="request" />
<%
  loginInfo.setUsername((String) request.getParameter("name"));
  loginInfo.setPassword((String) request.getParameter("password"));
%>
<html>
  <head><title>JSP Example 10 - First Bean</title></head>
<body>
  <h1>JSP Example 10 - First Bean</h1>
  <p>Username: <jsp:getProperty name="loginInfo" property="username"/></p>
  <p>Password: <jsp:getProperty name="loginInfo" property="password"/></p>
  <p>
   …
  </p>
</body></html>
```

2 JavaBeans

JSP Integration – Setting Bean Properties (2c)

```
<%
  beanexamples.LoginBean loginInfo = new beanexamples.LoginBean();
  loginInfo.setUsername((String) request.getParameter("name"));
  loginInfo.setPassword((String) request.getParameter("password"));
  request.setAttribute("loginInfo", loginInfo);
%>

<html>
  <head><title>JSP Example 10 - First Bean</title></head>
<body>

  <h1>JSP Example 10 - First Bean</h1>

  <p>Username: <jsp:getProperty name="loginInfo" property="username"/></p>
  <p>Password: <jsp:getProperty name="loginInfo" property="password"/></p>
  <p>
    …
  </p>

</body></html>
```

2 JavaBeans

JSP Integration – Setting Bean Properties (2)

```
<jsp:useBean id="loginInfo" class="beanexamples.LoginBean" scope="request" />
<jsp:setProperty name="loginInfo" property="username"
                 value="<%=request.getParameter("name")%>" />
<jsp:setProperty name="loginInfo" property="password"
                 value="<%=request.getParameter("password")%>" />
```

```
<jsp:useBean id="loginInfo" class="beanexamples.LoginBean" scope="request" />
<%
  loginInfo.setUsername((String) request.getParameter("name"));
  loginInfo.setPassword((String) request.getParameter("password"));
%>
```

```
<%
  beanexamples.LoginBean loginInfo = new beanexamples.LoginBean();
  loginInfo.setUsername((String) request.getParameter("name"));
  loginInfo.setPassword((String) request.getParameter("password"));
  request.setAttribute("loginInfo", loginInfo);
%>
```

2 JavaBeans

JSP Integration – Setting Bean Properties (3)

```
package beanexamples;

import=java.beans.*;
import=java.io.Serializable;

public class LoginBean implements java.io.Serializable {

  public LoginBean() { }

  ...

  public String getPassword() { return password; }
  public void setPassword(String password) {
    if(password != null) this.password = password;
  }

  public boolean isMale() { return male; }
  public void setMale(boolean male) { this.male = male; }

  private String username;
  private String password;
  private boolean male;

}
```

non String property

beanexamples/LoginBean.java

2 JavaBeans

JSP Integration – Setting Bean Properties (3)

```
<%
  beanexamples.LoginBean loginInfo = new beanexamples.LoginBean();
  loginInfo.setUsername((String) request.getParameter("name"));
  loginInfo.setPassword((String) request.getParameter("password"));
  String isMaleParameter = (String) request.getParameter("male"));
  if(isMaleParameter != null) {
    loginInfo.setMale(Boolean.getBoolean(isMaleParameter));
  }
  request.setAttribute("loginInfo", loginInfo);
%>

...
  <p>Username: <jsp:getProperty name="loginInfo" property="username"/></p>
  <p>Password: <jsp:getProperty name="loginInfo" property="password"/></p>
  <p>Male?     <jsp:getProperty name="loginInfo" property="male"/></p>
...
```

type conversion

2 JavaBeans

JSP Integration – Setting Bean Properties (4)

```
<jsp:useBean id="loginInfo" class="beanexamples.LoginBean" scope="request" />
<jsp:setProperty name="loginInfo" property="username" param="name" />
<jsp:setProperty name="loginInfo" property="password" param="password" />
<jsp:setProperty name="loginInfo" property="male"     param="male" />
...
  <p>Username: <jsp:getProperty name="loginInfo" property="username"/></p>
  <p>Password: <jsp:getProperty name="loginInfo" property="password"/></p>
  <p>Male?     <jsp:getProperty name="loginInfo" property="male"/></p>
...
```

automatic type conversion

! no action if the input parameter is missing from the request

2 JavaBeans

JSP Integration – Setting Bean Properties (5)

```
<jsp:useBean id="loginInfo" class="beanexamples.LoginBean" scope="request" />
<jsp:setProperty name="loginInfo" property="*" />
...
  <p>Username: <jsp:getProperty name="loginInfo" property="username"/></p>
  <p>Password: <jsp:getProperty name="loginInfo" property="password"/></p>
  <p>Male?     <jsp:getProperty name="loginInfo" property="male"/></p>
...
```

- all properties ↔ identically named input parameters

2 JavaBeans

! JavaBean objects bound to local variables in _jspService()

Bean Declaration

- global declaration
  - all users can access the bean (class variable)
  - ! potentially insecure

```
<%!
  beanexamples.LoginBean loginInfo = new beanexamples.LoginBean();
/>
```

- local declaration
  - every user refers to an own *local* variable (in **`_jspService()`**)

```
<%
  beanexamples.LoginBean loginInfo = new beanexamples.LoginBean();
/>
```

2 JavaBeans

Bean Declaration

- "useBean declaration"
  - every user refers to an own *local* variable

```
<jsp:useBean id="loginInfo" class="beanexamples.LoginBean" scope="session"/>
```

- scope (optional parameter, default: "page")
  - **`page`** → **`PageContext`** (this page)
  - **`request`** → **`ServletRequest`** (same request)
  - **`session`** → **`HttpSession`** (during the session)
  - **`application`** → **`ServletContext`** (during the servlet lifetime)

2 JavaBeans

- Conditional Bean Creation with **`<jsp:useBean />`**
  - ⇒ instantiation of a new bean
  - ⇒ binding the "`id` variable" to an existing bean
    - with the same `scope` and `id`
    - typecast if the bean is more specific than the declared bean
    - might result in a `ClassCastException`
  - conditional execution of "init code"
    - code that is only executed if a new bean is created

```
<jsp:useBean id="loginInfo" class="beanexamples.LoginBean" scope="session">
  <jsp:setProperty name="loginInfo" property="username" param="name"/>
</jsp:useBean>
```

2 JavaBeans

A look at the resulting code...

```
<jsp:useBean id="loginInfo" class="beanexamples.LoginBean" scope="session"/>
<jsp:setProperty name="loginInfo" property="username" param="name"/>
...
<jsp:getProperty name="loginInfo" property="password"/>
```

Receiver.jsp

Receiver_jsp.java

```
beanexamples.LoginBean loginInfo = null;
synchronized(session) {
  loginInfo = (beanexamples.LoginBean)
    _jspx_page_context.getAttribute("loginInfo", PageContext.SESSION_SCOPE);
  if (loginInfo == null) {
    loginInfo = new beanexamples.LoginBean();
    _jspx_page_context.setAttribute("loginInfo", loginInfo, PageContext.SESSION_SCOPE);
  }
}
org.apache.jasper.runtime.JspRuntimeLibrary.introspecthelper(
  _jspx_page_context.findAttribute("loginInfo"),
  "username", request.getParameter("name"), request, "name", false);
...
out.write(
  org.apache.jasper.runtime.JspRuntimeLibrary.toString(
    (((beanexamples.LoginBean)_jspx_page_context.findAttribute("loginInfo")
  ).getPassword())));
```

2 JavaBeans

How Does the Code Look Like?

```
<jsp:useBean id="loginInfo" class="beanexamples.LoginBean" scope="session"/>
<jsp:setProperty name="loginInfo" property="username" param="name"/>
...
<jsp:getProperty name="loginInfo" property="password"/>
```

Receiver.jsp

```
beanexamples.LoginBean loginInfo = null;
synchronized(session) {
  loginInfo = (beanexamples.LoginBean)
    _jspx_page_context.getAttribute("loginInfo", PageContext.SESSION_SCOPE);
  if (loginInfo == null) {
    loginInfo = new beanexamples.LoginBean();
    _jspx_page_context.setAttribute("loginInfo", loginInfo, PageContext.SESSION_SCOPE);
  }
}
org.apache.jasper.runtime.JspRuntimeLibrary.introspecthelper(
  _jspx_page_context.findAttribute("loginInfo"),
  "username", request.getParameter("name"), request, "name", false);
...
out.write(
  org.apache.jasper.runtime.JspRuntimeLibrary.toString(
    (((beanexamples.LoginBean)_jspx_page_context.findAttribute("loginInfo")
  ).getPassword())));
```

Receiver_jsp.java

2 JavaBeans

How Does the Code Look Like?

```
<jsp:useBean id="loginInfo" class="beanexamples.LoginBean" scope="session"/>
<jsp:setProperty name="loginInfo" property="username" param="name"/>
...
<jsp:getProperty name="loginInfo" property="password"/>
```

Receiver.jsp

```
beanexamples.LoginBean loginInfo = null;
synchronized(session) {
  loginInfo = (beanexamples.LoginBean)
    _jspx_page_context.getAttribute("loginInfo", PageContext.SESSION_SCOPE);
  if (loginInfo == null) {
    loginInfo = new beanexamples.LoginBean();
    _jspx_page_context.setAttribute("loginInfo", loginInfo, PageContext.SESSION_SCOPE);
  }
}
org.apache.jasper.runtime.JspRuntimeLibrary.introspecthelper(
  _jspx_page_context.findAttribute("loginInfo"),
  "username", request.getParameter("name"), request, "name", false);
...
out.write(
  org.apache.jasper.runtime.JspRuntimeLibrary.toString(
    (((beanexamples.LoginBean)_jspx_page_context.findAttribute("loginInfo")
  ).getPassword())));
```

Receiver_jsp.java

2 JavaBeans

How Does the Code Look Like?

```
<jsp:useBean id="loginInfo" class="beanexamples.LoginBean" scope="session"/>
<jsp:setProperty name="loginInfo" property="username" param="name"/>
...
<jsp:getProperty name="loginInfo" property="password"/>
```

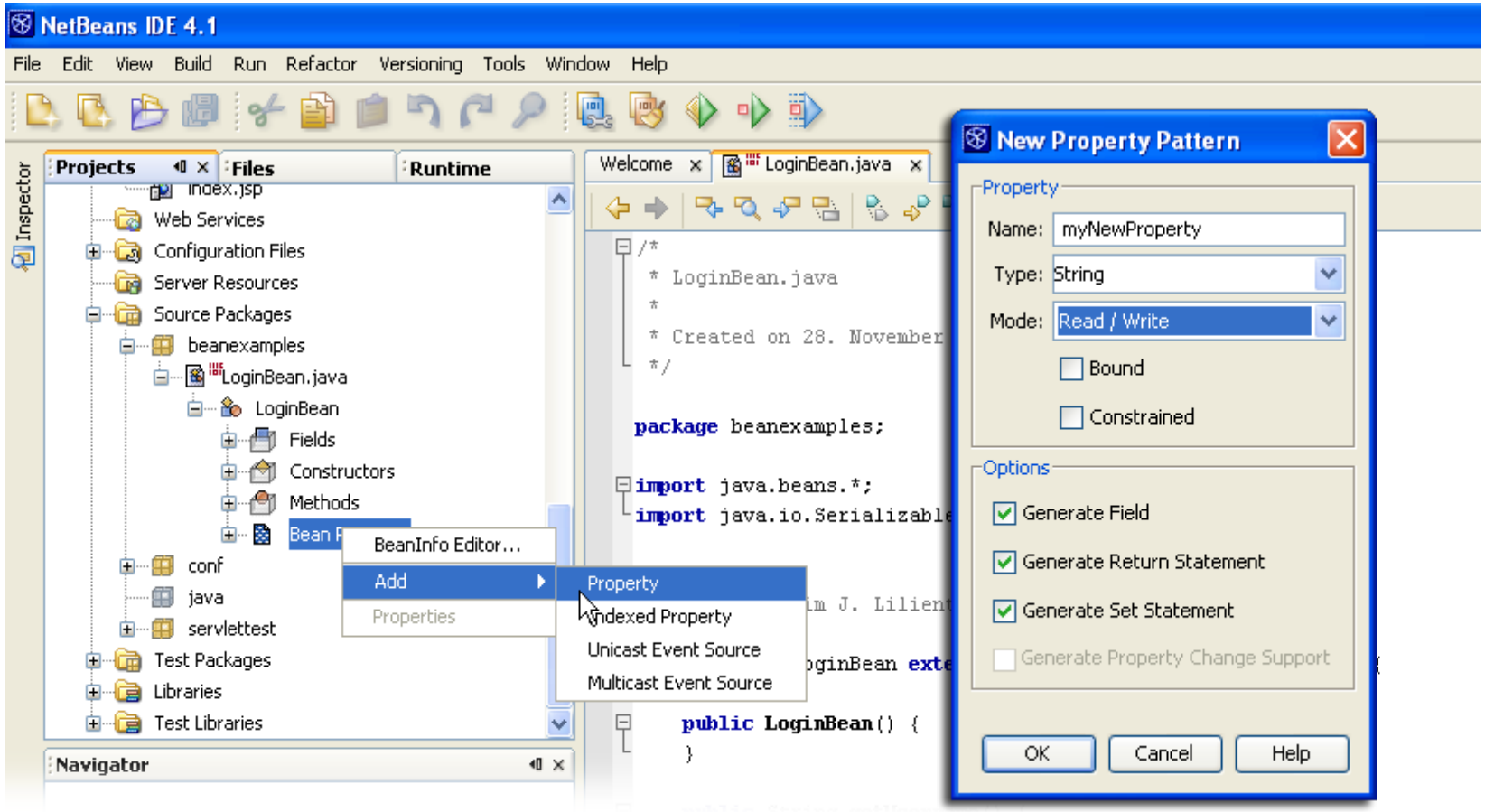
Receiver.jsp

Receiver_jsp.java

```
beanexamples.LoginBean loginInfo = null;
synchronized(session) {
  loginInfo = (beanexamples.LoginBean)
    _jspx_page_context.getAttribute("loginInfo", PageContext.SESSION_SCOPE);
  if (loginInfo == null) {
    loginInfo = new beanexamples.LoginBean();
    _jspx_page_context.setAttribute("loginInfo", loginInfo, PageContext.SESSION_SCOPE);
  }
}
org.apache.jasper.runtime.JspRuntimeLibrary.introspecthelper(
  _jspx_page_context.findAttribute("loginInfo"),
  "username", request.getParameter("name"), request, "name", false);
...
out.write(
  org.apache.jasper.runtime.JspRuntimeLibrary.toString(
    (((beanexamples.LoginBean)_jspx_page_context.findAttribute("loginInfo")
  ).getPassword())));
```

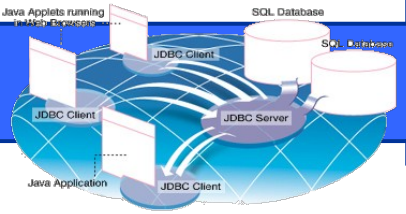
2 JavaBeans

JavaBean Support in NetBeans



JDBC

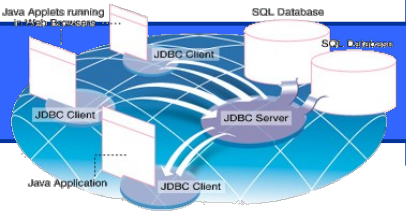
3 JDBC



What is JDBC?

- standard library for accessing relational databases
- send SQL statements in a vendor-independent way
  - standardised way to establish a connection to a database
  - standardised way of sending queries
  - standardised way of committing transactions
  - allows for vendor specific SQL syntax (try to avoid that!)
- part of JDK since version 1.4
- requires a JDBC driver for the specific database engine

3 JDBC



- Some Literature On JDBC ...
  - The Java Tutorial. Trail: JDBC Database Access
    http://java.sun.com/docs/books/tutorial/jdbc
  - Getting started with the JDBC API
    http://java.sun.com/j2se/1.5.0/docs/guide/jdbc/getstart/GettingStartedTOC.fm.html
  - JDBC 2.0 Fundamentals
    http://java.sun.com/developer/onlineTraining/Database
  - Get Started with **Mimer** JDBC
    http://developer.mimer.com/howto/howto_17.htm
  - Using JDBC with **MySQL**, Getting started
    http://www.developer.com/java/data/print.php/3417381

3 Mimer

Mimer Installation

- Version 9.3.8 (Windows, Linux)
- download from http://developer.mimer.com/downloads
- select operating system
- installation
  - double click installation (windows)
  - 'rpm -i MimerSQL-9.3.8b-1.i386.rpm', or unpack MimerEngineLinux938b.tar and run 'miminstall'
  - [*server*] to set up as server: run `mimadmin`
  - [*client*] text interface to existing server: run

```
bsql --username=<uname> --password=<pass>
<DBname>
```

3 MySQL

MySQL Installation

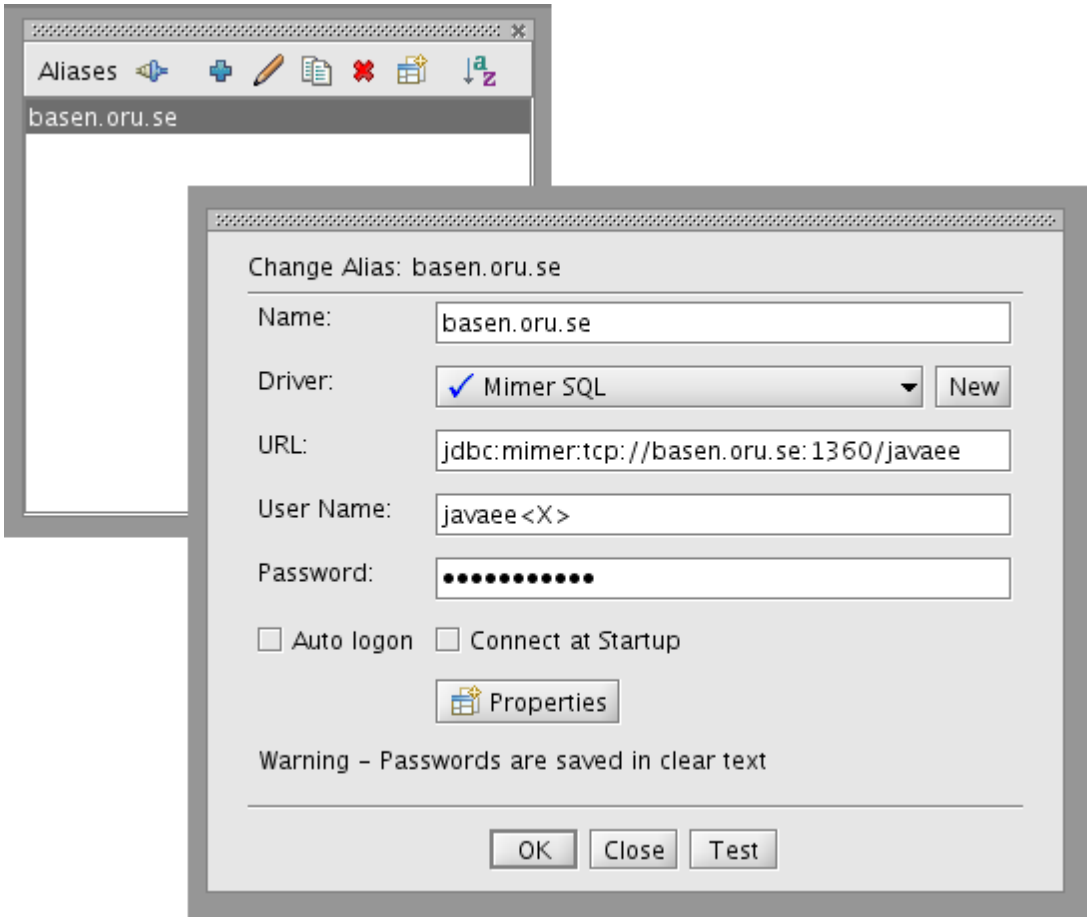
- Version 5.0.15 (Windows, Linux)
- download from http://dev.mysql.com/downloads (MySQL Community Server)
- double click installation (Standard Configuration)
- run MySQL Server Instance Configuration Wizard
  - default port: 3306
  - set standard character set to UTF8
  - set root password (***${rpwd}***)

3 SQuirrel SQL + (Mimer/MySQL/...)

- SQuirrel SQL: Universal SQL client
  - implemented with Java/JDBC
  - Version 2.6.5a (all platforms)
  - download from http://squirrel-sql.sourceforge.net/
  - installation
    - run `java -jar squirrel-sql-2.6.5a-install.jar` and follow instructions; run `squirrel-sql.[bat|sh]`
  - Driver: Mimer SQL
  - URL: `jdbc:mimer:tcp://basen.oru.se:1360/javaee`
  - Username/Password: will be distributed

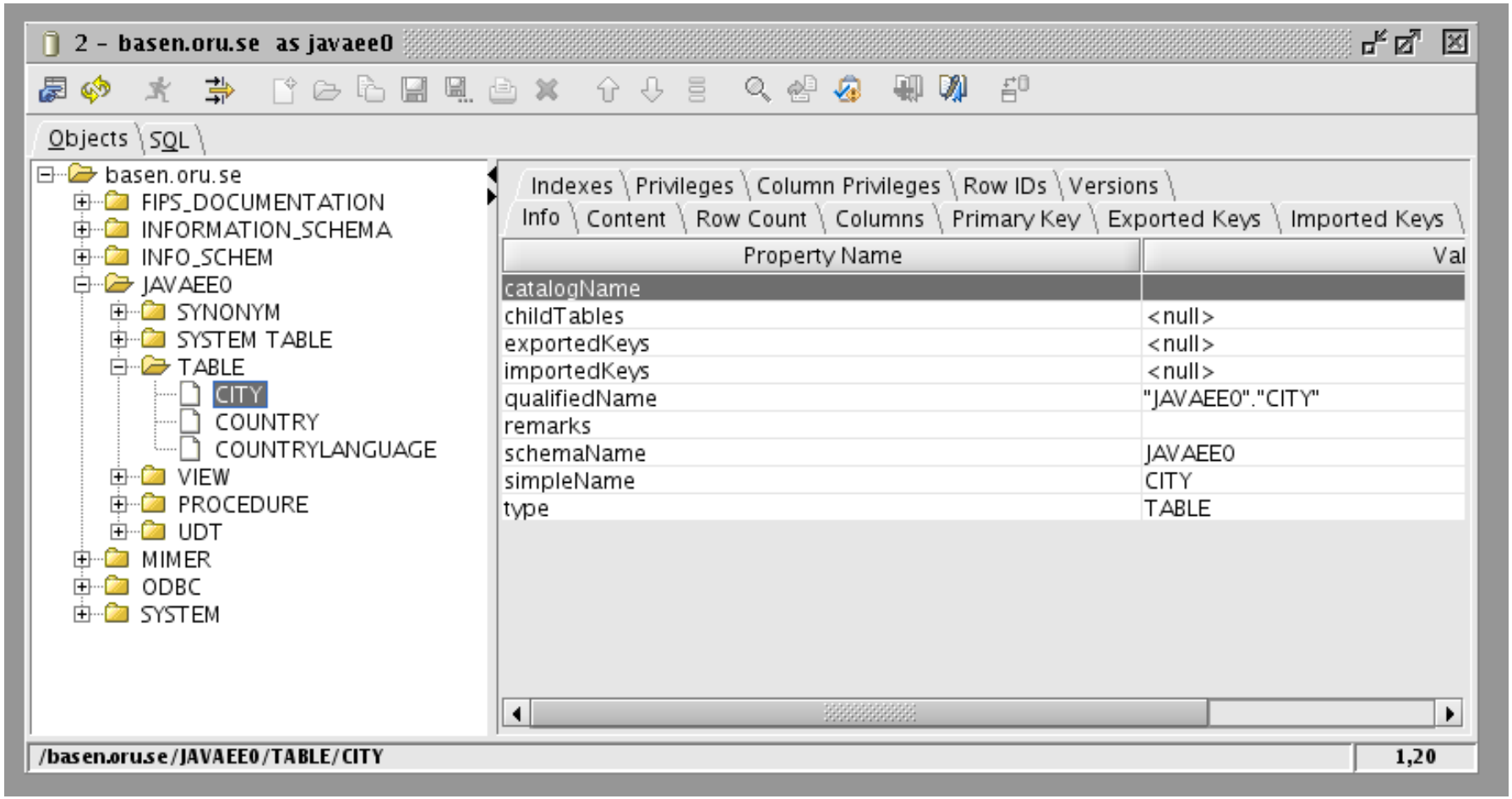
3 SQuirrel SQL + (Mimer/MySQL/...)

- SQuirrel SQL: Universal SQL client



3 SQuirrel SQL + (Mimer/MySQL/...)

- SQuirrel SQL: Universal SQL client



3 SQuirrel SQL + (Mimer/MySQL/...)

- SQuirrel SQL: Universal SQL client

2 - basen.oru.se as javaee0

Objects | SQL

select * from City

← Alt-Enter

Limit rows: 100

```
select * from City;
```

select * from City

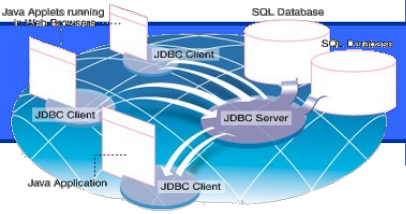
Limited to 100 rows; select * from City

Results | MetaData | Info

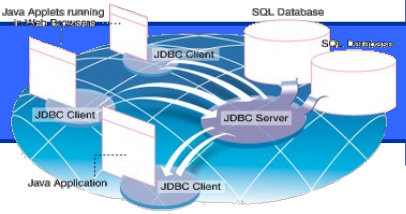
ID	NAME	COUNTRY...	DISTRICT	POPULATI...
2307	Kabul	AFG	Kabol	1780000
2308	Qandahar	AFG	Qandahar	237500
2309	Herat	AFG	Herat	186800
2310	Mazar-e-Sharif	AFG	Balkh	127800
2311	Amsterdam	NLD	Noord-Holland	731200
2312	Rotterdam	NLD	Zuid-Holland	593321
2313	Haag	NLD	Zuid-Holland	440900

/basen.oru.se/JAVAEE0/TABLE/CITY 1,20

3 JDBC



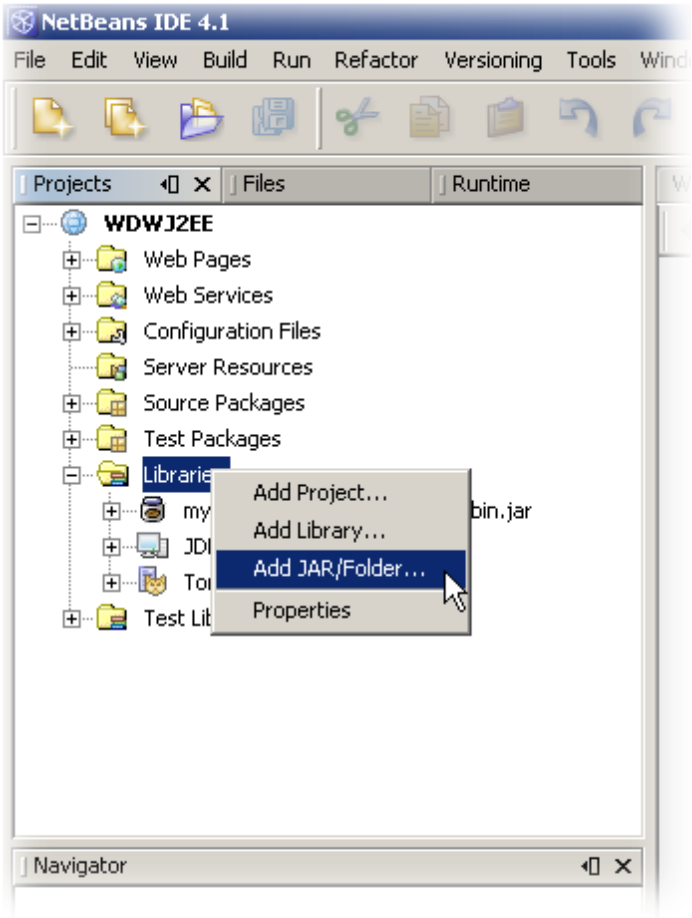
- JDBC Driver Installation: Mimer
  - download from (ver. 3.21)
    http://developer.mimer.com/downloads
- JDBC Driver Installation: MySQL
  - download from (ver. 5.1)
    http://dev.mysql.com/downloads/connector/j/5.1.html
- Unzip jar file and add to CLASSPATH

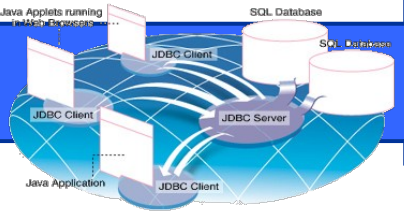


3 JDBC

JDBC Driver Installation

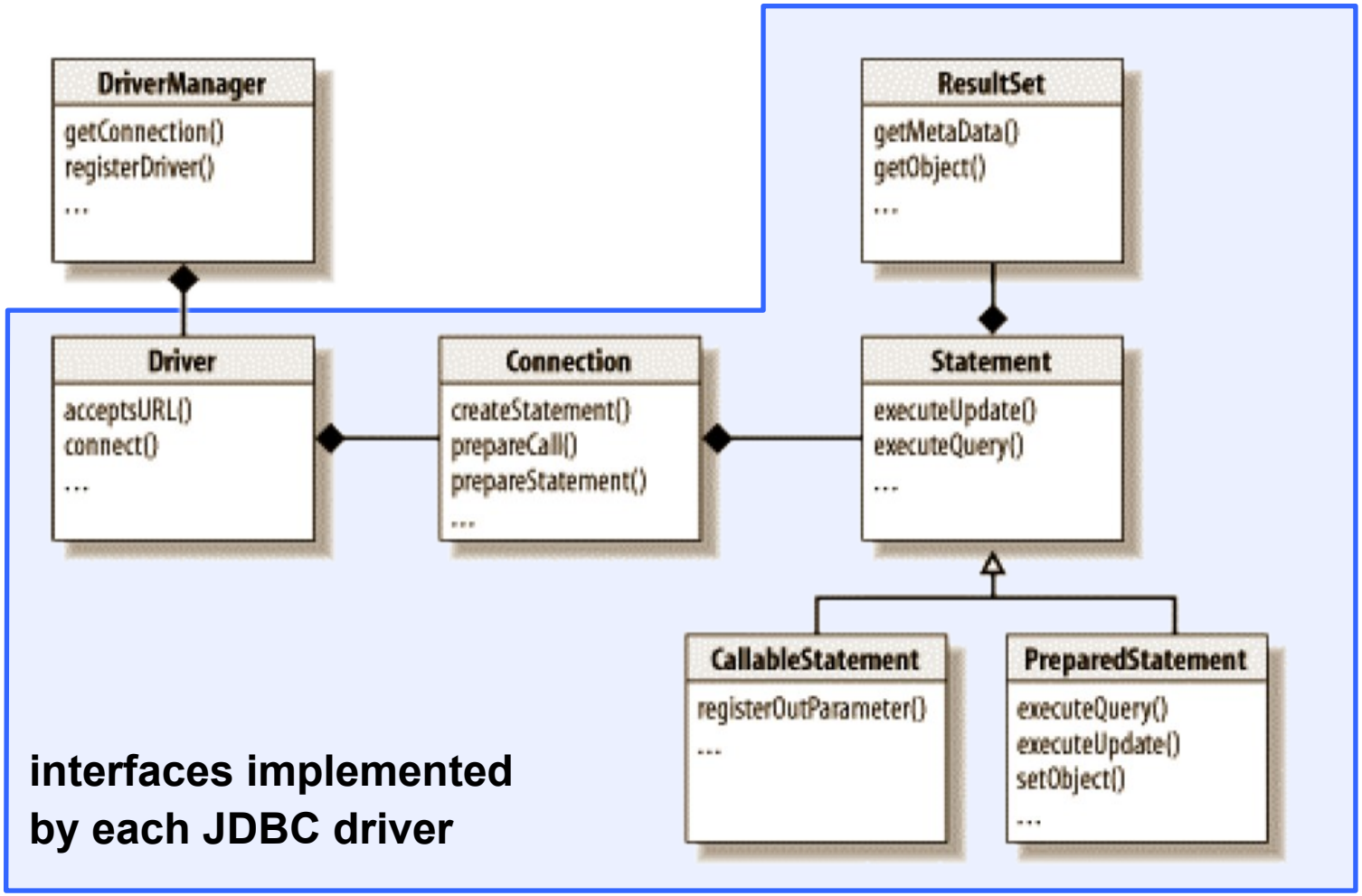
- Add JAR/Folder to the NetBeans project...
- Add JAR/Folder to the Eclipse project...
- ... or add to CLASSPATH if using command line

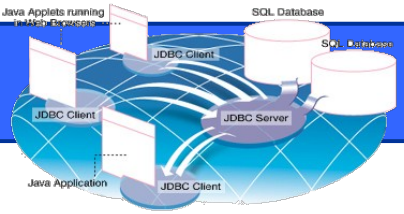




3 JDBC

JDBC Architecture





3 JDBC

Using JDBC – A First Example/1

- package: java.sql.*

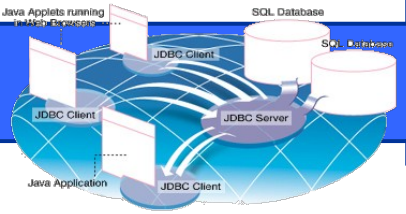
```
<%@page contentType="text/html" %>
<%@page pageEncoding="UTF-8" %>
<%@page import="java.sql.*"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP+JDBC Example 1 - Check The JDBC Connection</title>
  </head>
<body>

  <%! final String MIMER DRIVER = "com.mimer.jdbc.Driver"; %>

  <h1>JSP+JDBC Example 1 - Check The JDBC Connection</h1>
  <p>The basic steps when using JDBC are listed below:</p>
      ...
```

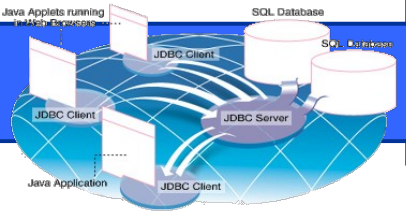
3 JDBC



Using JDBC – A First Example/2

- loading the database driver = load the driver class
  - without making an instance of it (init is handled by a `static` block)
  - avoid hard-coding the class name

```
...
  <ol>
  <li>Loading the driver ...
    <% try { Class.forName(MIMER_DRIVER); %>
         done!
    <% }
       catch(ClassNotFoundException cnfe) { %>
         failed because the class <%=MIMER_DRIVER%> was not found.
    <% } %>
  </li>
...
```

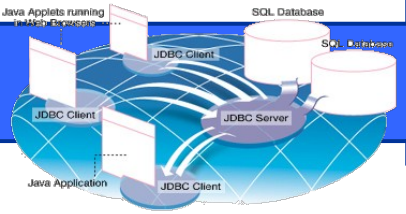


3 JDBC

Using JDBC – A First Example/3

- define the connection URL
  - the URL uses the jdbc: protocol
  - includes: server, host, port and the database name

```
...
  <%! final String MIMER_HOST = "basen.oru.se";
    final String MIMER_PORT = "1360";
    final String MIMER_PROT = "tcp";
    final String MIMER_DATABASE_NAME = "javaee";
    final String MIMER_CONNECTION_URL = "jdbc:mimer:" + MIMER_PROT + "://" +
      MIMER_HOST + ":" + MIMER_PORT + "/" + MIMER_DATABASE_NAME;
    Connection connection = null;
    final String MIMER_USERNAME = "javaeeX";
    final String MIMER_PASSWORD = "XXXXXXXX"; %>
...
```

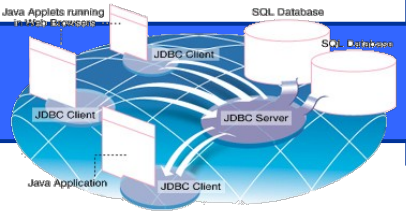


3 JDBC

Using JDBC – A First Example/4

- establish a connection to the database
- ... and catch the SQLException

```
...
  <li>Establish the connection... (<%=MIMER_CONNECTION_URL%>)
    <% try {
         connection = DriverManager.getConnection(
           MIMER_CONNECTION_URL,MIMER_USERNAME,MIMER_PASSWORD); %>
         done!
    <% }
       catch(SQLException sqle) { %>
         failed. Could not connect to <%=MIMER_CONNECTION_URL%>.
    <% } %>
  </li>
...
```



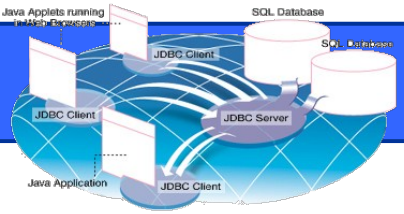
3 JDBC

Using JDBC – A First Example/5

- extract database metadata

```
...
  <%! DatabaseMetaData dbMetaData = null; %>

  <li>Extract database metadata ...
    <% if(connection != null) {
          try {
            dbMetaData = connection.getMetaData();
            String dbProductName = dbMetaData.getDatabaseProductName();
            String dbProductVersion = dbMetaData.getDatabaseProductVersion();
%>
      done! (Name = <%=dbProductName%>, Version = <%=dbProductVersion%>)
    <% }
          catch(SQLException sqle) { %>
      failed.
    <%   }
       } else { %>
      failed due to previous errors.
    <% } %>
  </li>
...
```



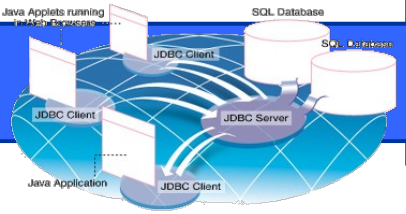
3 JDBC

Using JDBC – A First Example/6

- execute a query

```
...
  <%! Statement statement = null;
    ResultSet resultSet = null;
    final String MIMER_TABLE = "Country"; %>

  <li>Execute a query ...
    <% if(connection != null) {
          try {
            statement = connection.createStatement();
            String query = "SELECT * FROM " + MIMER_TABLE;
            resultSet = statement.executeQuery(query); %>
      done!
    <% } catch(SQLException sqle) { %>
      failed. Either the statement could not be created or the query failed.
    <%   }
       } else { %>
      failed due to previous errors.
    <% } %>
  </li>
...
```

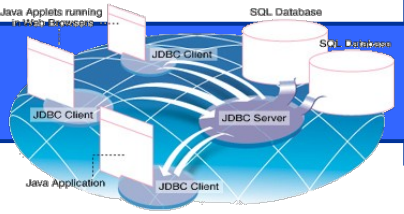


3 JDBC

Using JDBC – A First Example/7

- process the result

```
...
  <li>Display the result ...<br/>
    <% if(resultSet != null) {
          try {
            ResultSetMetaData resMetaData = resultSet.getMetaData();
            int colCount = resMetaData.getColumnCount();
            while(resultSet.next()) {
              for(int i = 1; i <= colCount; i++)  { %>
                 [<%=i%>] <%=resultSet.getString(i)%>
           <% }
            } %>
            done!
      <% }
         catch(SQLException sqle) { %>
           failed. Metadata could not be extracted from the result set.
      <% }
...
```



3 JDBC

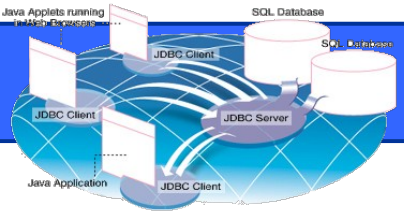
Using JDBC – A First Example/7

- process the result

```
...
  <li>Display the result ...<br/>
    <% if(resultSet != null) {
          try {
            ResultSetMetaData resMetaData = resultSet.getMetaData();
            int colCount = resMetaData.getColumnCount();
            while(resultSet.next()) {
              for(int i = 1; i <= colCount; i++)  { %>
                 [<%=i%>] <%=resultSet.getString(i)%>
          <% }
            } %>
            done!
      <%
      <%
...
```

alternative

```
String  strForename = resultSet.getString("forename");
String strSurname = resultSet.getString("surname");
String strGender = resultSet.getString("gender");
java.util.Date annDate = resultSet.getDate("anndate");
String strDescription = resultSet.getString("description");
```



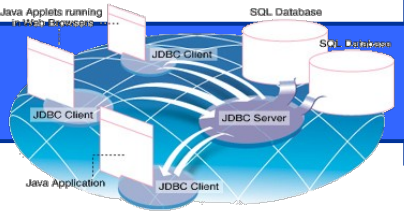
3 JDBC

Using JDBC – A First Example/8

- "cleaning up"

```
...
  finally {
    try {
      if(resultSet != null) {
        resultSet.close();
      }
      if(statement != null) {
        statement.close();
      }
      if(connection != null) {
        connection.close();
      }
    }
    catch(SQLException sqle) {
    }
  }
       } else { %>
          failed due to previous errors.
    <% } %>
  </li>
</ol></body></html>
```

3 JDBC



JSP+JDBC Example 1 - Check The JDBC Connection - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost:5512/JDBCExamples/JDBCExample01.jsp

JDBC sql client for mimer

Getting Started Latest BBC Headlines

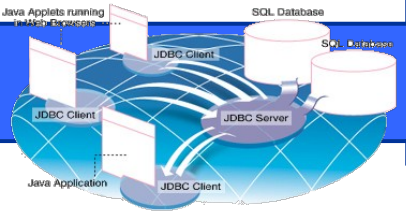
JSP+JDBC Example 1 – Check The JDBC Connection

The basic steps when using JDBC are listed below:

1. Loading the driver ... done!
2. Establish the connection... (jdbc:mimer:tcp://basen.oru.se:1360/javaee) done!
3. Extract database metadata ... done! (Name = Mimer SQL Engine, Version = 09.03.0007 Mimer SQL Engine 9.3.7e)
4. Execute a query ... done!
5. Display the result ...
   [1] ABW [2] Aruba [3] North America [4] Caribbean [5] 193.00 [6] null [7] 103000 [8] 78.4 [9] 828.00 [10] 793.00 [11] Aruba [12] Nonmetropolitan Territory of The Netherlands [13] Beatrix [14] 129 [15] AW [1] AFG [2] Afghanistan [3] Asia [4] Southern and Central Asia [5] 652090.00 [6] 1919 [7] 22720000 [8] 45.9 [9] 5976.00 [10] null [11] Afganistan/Afqanestan [12] Islamic Emirate [13] Mohammad Omar [14] 1 [15] AF [1] AGO [2] Angola [3] Africa [4] Central Africa [5] 1246700.00 [6] 1975 [7] 12878000 [8] 38.3 [9] 6648.00 [10] 7984.00 [11] Angola [12] Republic [13] José Eduardo dos Santos [14] 56 [15] AO [1] AIA [2] Anguilla [3] North America [4] Caribbean [5] 96.00 [6] null [7] 8000 [8] 76.1 [9] 63.20 [10] null [11] Anguilla [12] Dependent Territory of the UK [13] Elisabeth II [14] 62 [15] AI [1] ALB [2] Albania [3] Europe [4] Southern Europe [5] 28748.00 [6] 1912 [7] 3401200 [8] 71.6 [9] 3205.00 [10] 2500.00 [11] Shqipëria [12] Republic [13] Rexhep Mejdani [14] 34 [15] AL [1] AND [2] Andorra [3] Europe [4] Southern Europe [5] 468.00 [6] 1278 [7] 78000 [8] 83.5 [9] 1630.00 [10] null [11] Andorra [12] Parliamentary Coprincipality [13] [14] 55 [15] AD [1] ANT [2] Netherlands Antilles [3] North America [4] Caribbean [5] 800.00 [6] null [7] 217000 [8] 74.7 [9] 1941.00 [10] null [11] Nederlandse Antillen [12] Nonmetropolitan Territory of The Netherlands [13] Beatrix [14] 33 [15] AN [1] ARE [2] United Arab Emirates [3] Asia [4] Middle East [5] 83600.00 [6] 1971 [7] 2441000 [8] 74.1 [9] 37966.00 [10] 36846.00 [11] Al-Imarat al-´Arabiya al-Muttahida [12] Emirate Federation [13] Zayid bin Sultan al-Nahayan [14] 65 [15] AE [1] ARG [2] Argentina [3] South America [4] South America [5] 2780400.00 [6] 1816 [7] 37032000 [8] 75.1 [9] 340238.00 [10] 323310.00 [11] Argentina [12] Federal Republic [13] Fernando de la Rúa [14] 69 [15] AR [1] ARM [2] Armenia [3] Asia [4] Middle East [5] 29800.00 [6] 1991 [7] 3520000 [8] 66.4 [9] 1813.00 [10] 1627.00 [11] Hajastan [12] Republic [13] Robert Kot[illegible]arjan [14] 126 [15] AM [1] ASM [2] American Samoa [3] Oceania [4] Polynesia [5] 199.00 [6] null [7] 68000 [8] 75.1 [9] 334.00 [10] null [11] Amerika Samoa [12] US Territory [13] George W. Bush [14] 54 [15] AS [1] ATA [2] Antarctica [3] Antarctica [4] Antarctica [5] 13120000.00 [6] null [7] 0 [8] null [9] 0.00 [10] null [11] [illegible] [12] Co-administrated [13] [14] null [15] AQ [1] ATF [2] French Southern territories [3] Antarctica [4] Antarctica [5] 7780.00 [6] null [7] 0 [8] null [9] 0.00 [10] null [11] Terres australes françaises [12] Nonmetropolitan Territory of France [13] Jacques Chirac [14] null [15] TF [1] ATG [2] Antigua and Barbuda [3] North America [4] Caribbean [5] 442.00 [6] 1981 [7] 68000 [8] 70.5 [9] 612.00 [10] 584.00 [11] Antigua and Barbuda [12] Constitutional Monarchy [13] Elisabeth II [14] 63 [15] AG [1] AUS [2] Australia [3] Oceania [4] Australia and New Zealand [5] 7741220.00 [6] 1901 [7] 18886000 [8] 70.8 [9] 351182.00 [10] 392911.00 [11] Australia [12] Constitutional Monarchy, Federation

Find: connect Next Previous Highlight all Match case

Done



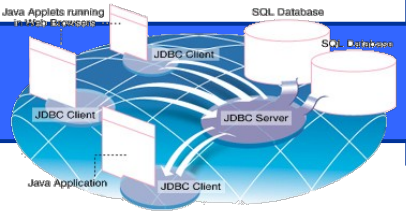
3 JDBC

Using JDBC – Table Modifications

- INSERT, DELETE, UPDATE, CREATE/DROP TABLE, ALTER TABLE, CREATE/DROP INDEX

```
...
  <li>Make changes in the database ...
    <% if(connection != null) {
         try {
           statement = connection.createStatement();
           String mod = "INSERT INTO " + MYSQL_TABLE + " VALUES ('" +
             "'Giordano', 'Bruno', 'm', '1600-02-17', 'day of death')";
           int rowCount = statement.executeUpdate(mod); %>
      done!
    <% }
         catch(SQLException sqle) { %>
      failed.
    <%   }
       } else { %>
      failed due to previous errors.
    <% } %>
  </li>
...
```

3 JDBC

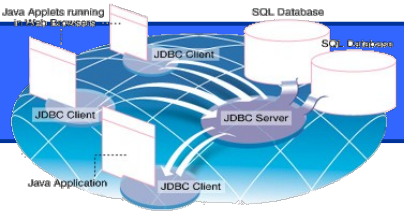


SQL Injection

Input	SQL Statement
admin,Lemon12	SELECT * FROM users WHERE username='admin' AND password='Lemon12'


```
statement = connection.createStatement();
String query = "SELECT * FROM users " +
  " WHERE username = '" + strUsername + " AND password = '" + strPassword;
resultSet = statement.executeQuery(query); %>
```

3 JDBC



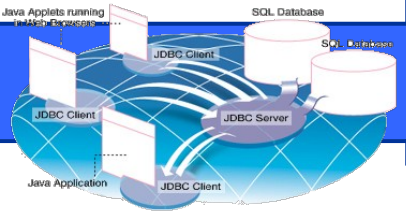
SQL Injection

Input	SQL Statement
admin,Lemon12	SELECT * FROM users WHERE username='admin' AND password='Lemon12'

- *user input is incorrectly filtered for string literal escape characters embedded in SQL statements,*
- *user input is not strongly typed and thereby unexpectedly executed*

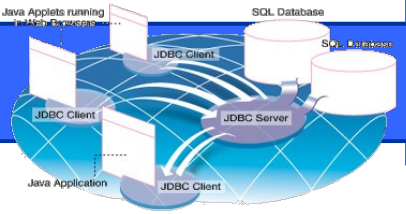
```
statement = connection.createStatement();
String query = "SELECT * FROM users " +
  " WHERE username = '" + strUsername + " AND password = '" + strPassword;
resultSet = statement.executeQuery(query); %>
```

3 JDBC



SQL Injection

Input	SQL Statement
admin,Lemon12	SELECT * FROM users WHERE username='admin' AND password='Lemon12'
' OR '1'='1,' OR '1'='1	SELECT * FROM users WHERE username='' OR '1' = '1' AND password='' OR '1' = '1'
admin' --,	SELECT * FROM users WHERE username='admin' --' AND password=''
admin'; DELETE FROM users --,	SELECT * FROM users WHERE username='admin'; DELETE FROM users --' AND password=''
' INSERT INTO users VALUES('got','you') --,	SELECT * FROM users WHERE username=,' INSERT INTO users VALUES('got','you') --' AND password=''



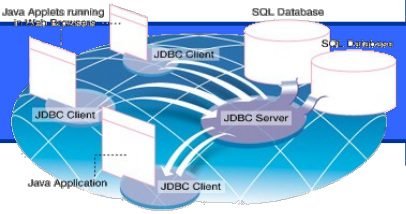
3 JDBC

Prepared Statements

- precompiled queries
  - avoid SQL injection (security!)
  - speed up SQL queries

```
...
<%
  String psql = "INSERT INTO " + MYSQL_TABLE +
    "(forename,surname,gender,anndate,description) " + "VALUES(?,?,?,?,?)";
  PreparedStatement pstatement = connection.prepareStatement(psql);
  pstatement.setString(1,aForename);
  pstatement.setString(2,aSurname);
  pstatement.setString(3,aGender);
  pstatement.setDate(4,aDate);
  pstatement.setString(5,aDescription);
  int rowCount = pstatement.executeUpdate();
%>
...
```

3 JDBC

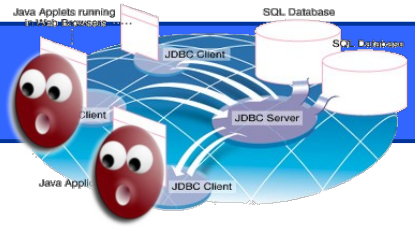


Callable Statements

- interface to SQL stored procedures
  - handled similar to prepared statements
  - result parameters must be registered as OUT

```
...
<%
  String csql = "{CALL MyProcedure(?,?,?,?)}";
  CallableStatement cstatement = connection.prepareStatement(csql);
  cstatement.setInt(1,id);
  cstatement.setInt(2,age);
  cstatement.setString(3,name);
  cstatement.registerOutParameter(4,Types.INTEGER);
  cstatement.execute();

  int nAnswer = cstatement.getInt(4);
%>
...
```



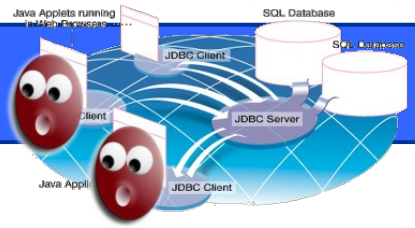
3 JDBC

Java Beans To Store Query Results

```
public class SongQuery {
  public SongBean[] getSongsByTitle(String strTitle, Connection conn) {
    ArrayList songs = new ArrayList();
    try {
      String songsByTitleSql =
        "SELECT album, artist, releaseyear FROM songsDb " +
        "WHERE title = '" + strTitle + "'";
      Statement stmt = conn.createStatement();
      ResultSet rs = stmt.executeQuery(songsByTitleSql);
      while(rs.next()) {
        String album    = rs.getString(1);
        String artist   = rs.getString(2);
        int releaseYear = rs.getInt(3);
        SongBean song = new SongBean(strTitle, album, artist, releaseYear);
        songs.add(song);
      }
      stmt.close(); conn.close();
    }
    catch(SQLException sqle) { ... }
    return (SongBean[]) songs.toArray(new SongBean[0]);
  }
...
```

! toArray() argument specifies the type of the returned array

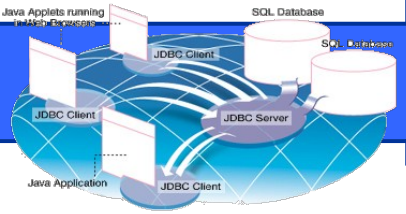
3 JDBC



Java Beans To Store Query Results

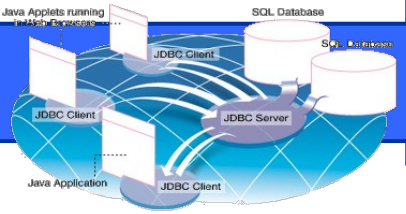
```
<%@ page import="java.sql.*, se.oru.wdwj2ee.SongBean" %>
<jsp:useBean id="songQuery" class="se.oru.wdwj2ee.SongQuery" scope="page" />
<% String strSongTitle = request.getParameter("songtitle");
   try {
     Class.forName("com.mysql.jdbc.Driver");
     Connection connection = DriverManager.getConnection(
       "jdbc:mysql://localhost:3306/test", "wdwj2ee", "SecurePwd");
   }
   catch(...){ ... }
   SongBean[] songs = songQuery.getSongsByTitle(strSongTitle, connection); %>
<html>
<head><title>Song Search</title></head>
<body>
  <p>All songs with title = <%=strSongTitle%></p>
  <table>
  <% for(int i = 0; i < songs.length; i++) { %>
     <tr>
       <td><%=songs[i].getAlbum()%></td> <td><%=songs[i].getArtist()%></td> ...
     </tr>
  <% } %>
  </table>
</body></html>
```

3 JDBC



- Multithreaded Database Access
  - standard example: transferring money between accounts
    - statement 1: remove MONEY_DIFF from Account1
    - statement 2: add MONEY_DIFF to Account2
  - statement 1 successful but statement 2 fails ⇒ problem
- Solutions
  - only one statement – SELECT inside INSERT
  - transactions
  - stored procedures

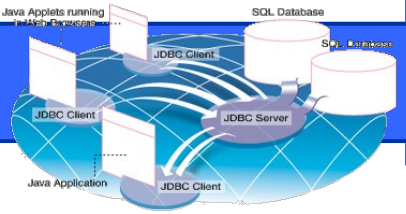
3 JDBC



- Transactions
  - atomic transactions of several SQL statements in sequence
  - if one statement fails: *roll back*
  - if all statements were successful: *commit*
- Transactions Pseudocode

```
switch off autoCommit
carry out a number of SQL statements
conclude with …
   COMMIT if all SQL statements were successful, or
   ROLLBACK otherwise
switch on autoCommit again
```

3 JDBC



Transactions

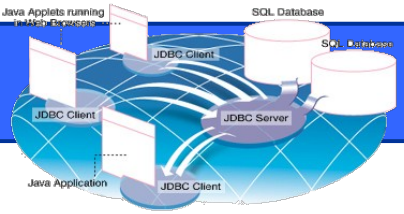
```
try {
  Class.forName("com.mysql.jdbc.Driver");
}
catch(ClassNotFoundException cnfe) {...}

Connection connection = null;
boolean bTransactionStarted = false;

try {
  connection = DriverManager.getConnection(
    "jdbc:mysql://localhost:3306/test", "wdwj2ee", "SecurePwd");
  connection.setAutoCommit(false);
  boolean bTransactionStarted = true;

  // Perform a number of SQL statements (using the connection)

  connection.setAutoCommit(true);
}
catch(SQLException sqle){
  ...
}
finally {
  ...
}
```



3 JDBC

Transactions

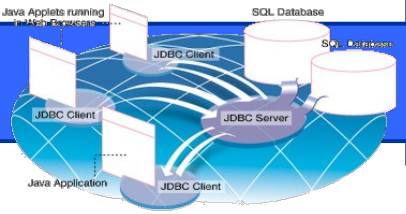
```
try {
  connection.setAutoCommit(false);
  boolean bTransactionStarted = true;

  // Perform a number of SQL statements (using the connection)

  connection.setAutoCommit(true);
}
catch(SQLException sqle){
  if(bTransactionStarted == true) {
    try {
      connection.rollback();
    }
    catch(SQLException sqle){ ... }
  }
}
finally {
  try {
    if(connection != null) {
      connection.setAutoCommit(true);
      connection.close(); }
  }
  catch(SQLException sqle){ ... }
}
```

! always executed no matter whether an exception was thrown or not

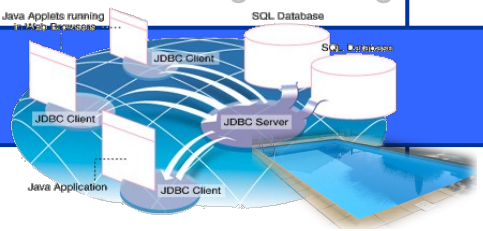
3 JDBC



- Database Access in a Multi-User Environment
  - using a single **`Connection`** per resource
    - ☹ JDBC driver may not support multiple threads per **`Connection`**
    - ☹ transactions might include multiple users
  - using one **`Connection`** per user
    - ☹ creating database connections is time-consuming
    - ☹ the connection will be inactive most of the time
    - ☹ keeping database connections open is expensive (memory, money)
- Solution: A Connection Pool

  - **`Connection`** objects shared by all Servlets and JSPs

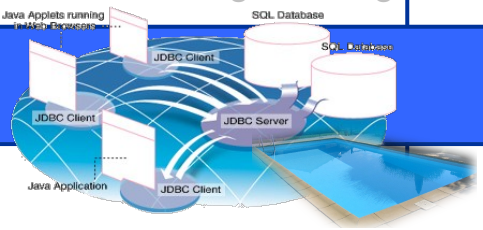
3 JDBC



- Connection Pool
  - check out a **`Connection`** from the pool for each request
  - put back the **`Connection`** to the pool after it was used
- Benefits of a Connection Pool Class
  - connections are created only once (up to N_{max} connections)
  - each request gets its own **`Connection`** (one thread at a time)
  - efficiency: “recycling” of connections (N_{max} can be adjusted)
  - database engine cannot enforce access restrictions

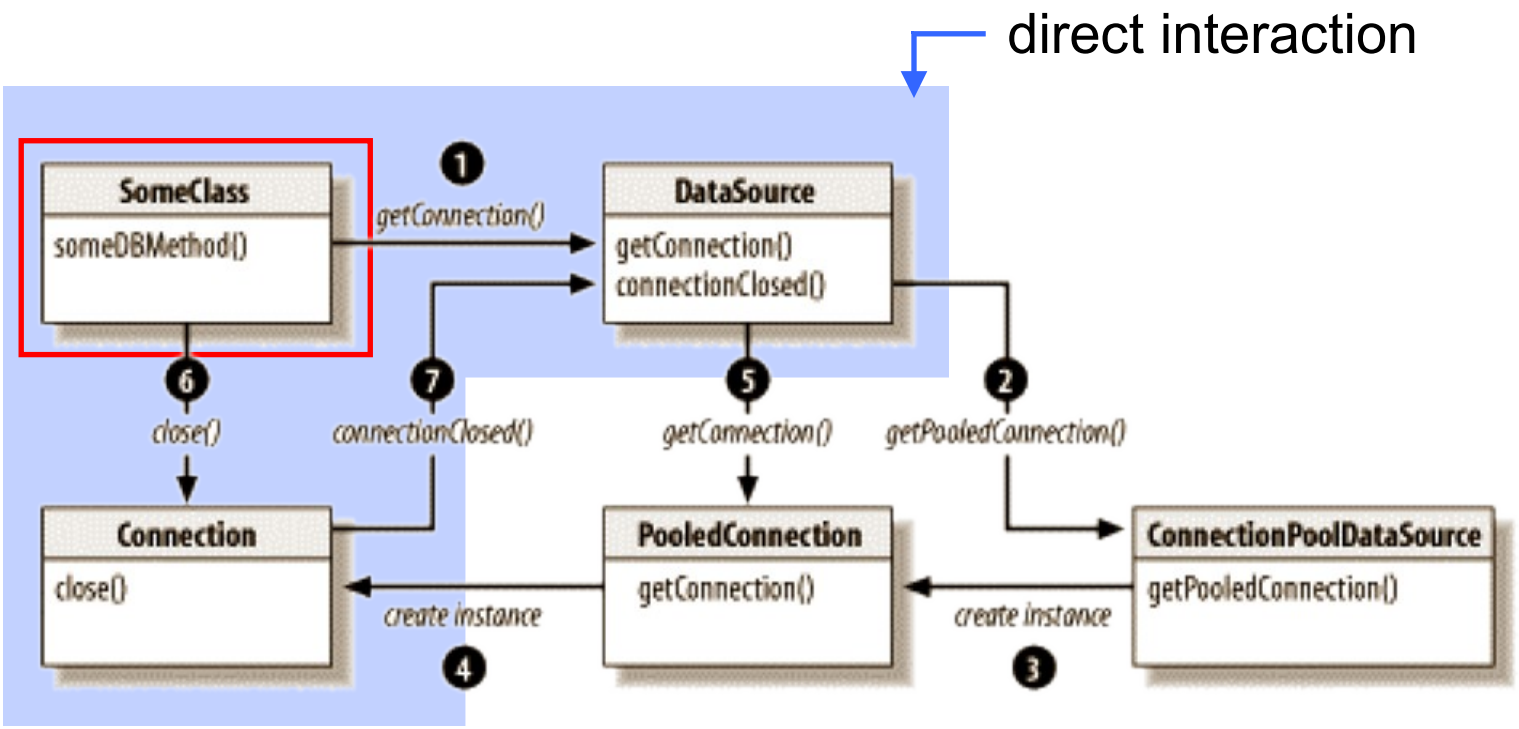
    $\Rightarrow$ use a connection pool for each role
  - code for database access is encapsulated “at one place”

3 JDBC

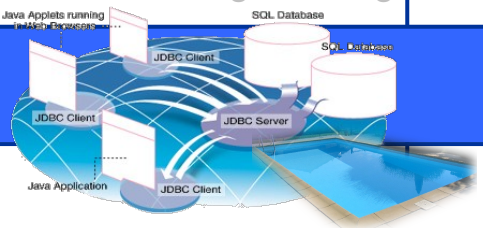


Connection Pool Interface

- since JDBC 3.0 (included in JDK 1.4, JDK 5, JDK 6)

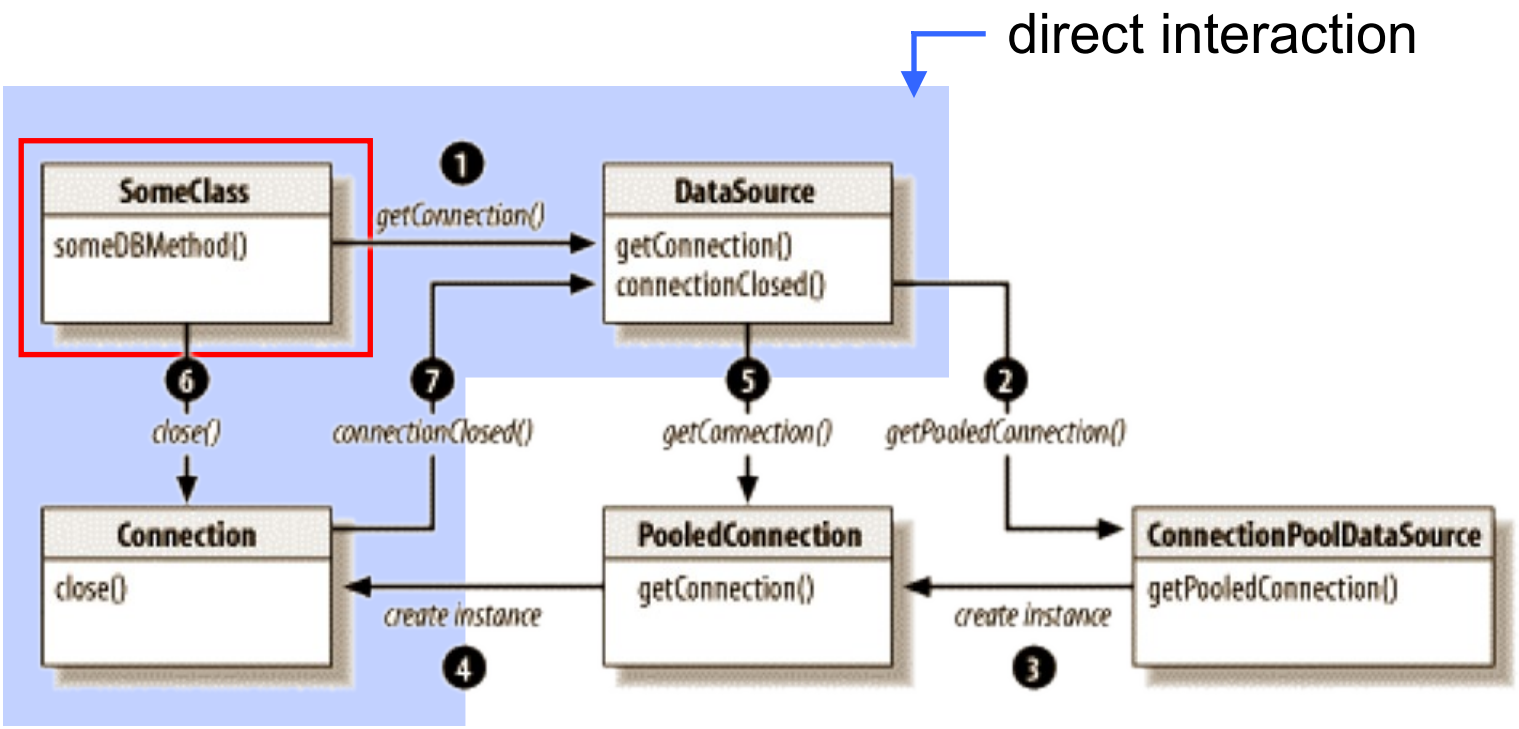


3 JDBC

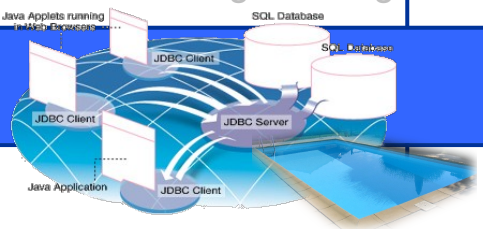


Connection Pool Interface

- close() releases a **`Connection`** rather than closing it



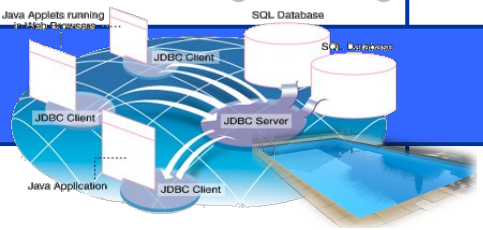
3 JDBC



Connection Pool Implementations

- Tomcat
  - provides the `javax.sql.DataSource` class for connection pooling
- Mimer JDBC driver
  - provides `com.mimer.jdbc.MimerConnectionPoolDataSource` class for connection pooling
- DBConnectionBroker (http://www.javaexchange.com/)
  - product-independent – works with any database accessible via JDBC

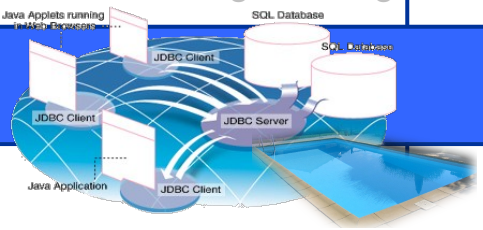
3 JDBC



DBConnectionBroker

```
<%@ page import="java.sql.*,java.io.*,com.javaexchange.dbConnectionBroker.*" %>
<%! final String MYSQL_DRIVER = "com.mysql.jdbc.Driver";
    final String MYSQL_CONNECTION_URL = "jdbc:mysql://localhost:3306/test";
    final String MYSQL_USERNAME = "wdwj2ee";
    final String MYSQL_PASSWORD = "SecurePwd";
    final int INIT_CONNECTIONS = 2;
    final int MAX_CONNECTIONS = 10;
    final String LOGFILE = "C:\\ConnectionBroker.log";
    final double REFRESH_INTERVAL = 1.0;
    DbConnectionBroker myBroker = null; %>
<%
  try {
    myBroker = new DbConnectionBroker(MYSQL_DRIVER,
      MYSQL_CONNECTION_URL, MYSQL_USERNAME, MYSQL_PASSWORD,
      INIT_CONNECTIONS, MAX_CONNECTIONS, LOGFILE, REFRESH_INTERVAL);
  }
  catch(IOException ioee) { ... }
  }
  ...
```

3 JDBC



DBConnectionBroker

```
<%! Connection connection = null;
    PreparedStatement prStatement = null;
    ResultSet resultSet = null;
    final String MYSQL_Table = "users"; %>
<%
  ...
  if(myBroker != null) {
    connection = myBroker.getConnection();
    int nThisConnId = myBroker.idOfConnection(connection);
  }
  if(connection != null) {
    try {
      prStatement = connection.prepareStatement("SELECT * FROM " + MYSQL_TABLE
+
        " WHERE username LIKE '%a%' AND birthday > ?");
      prStatement.setString(1,"1995-01-01");
      resultSet = prStatement.executeQuery();
    }
    catch(SQLException sqle) { ... }
  }
  connection.close();
  ...
```

Character Encoding

4 Character Encoding

- Providing Dynamic Content in Several Languages
  - I18N (Internationalisation)
    - identify the parts that are different for different geographical regions
  - L10N (Localisation)
    - provide the messages, graphics, ... for a particular region
- Locale (java.util.Locale)
  - represents a geographical, political or cultural region
  - sv_SE or sv-SE (*LanguageCode_CountryCode*)
  - HTTP request contains an Accept-Language header
  - stored as locale / locales in the request scope

4 Character Encoding

- Charsets
  - Mapping between 8 bits and a character
    - ISO-8859-1 for English, Swedish, German, ...
    - ISO-8859-2 for Polish, Hungarian, ...
  - More than 8 Bits
    - Big5 (Chinese), Shift_JIS (Japanese), EUC-KR (Korean)
- Unicode
  - 2 Bytes (UTF-8: 1, 2, or 3 bytes)
  - Java uses Unicode internally
  - UTF-8 supported by all browsers (correct display if fonts are installed)

4 Character Encoding

Unicode Output

how is the response sent to a browser ?

how is the JSP file encoded ?

```
<%@ page pageEncoding="UTF-8" contentType="text/html;charset=UTF-8" %>
...
```

Unicode Input

- form parameters must be encoded as UTF-8
- receiver servlet must treat the form parameters as UTF-8
- database must be configured properly

4 Character Encoding

- Unicode Input
  - form parameters must be encoded as UTF-8
    - `<%@ page contentType="text/html;charset=UTF-8" %>`
    - or `<form enctype="UTF-8" ... >`
  - receiver servlet must treat the form parameters as UTF-8
    - ! some browsers don't send the Content-Type request header

      ⇒we must tell the container which charset to use for the request

```
<%-- Method 1 --%>
<% try { pageContext.getRequest().setCharacterEncoding("UTF-8"); }
   // might also use the implicit 'request' object directly
   catch(UnsupportedEncodingException ueee) { } %>
<%-- Method 2 --%>
<%@taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<fmt:requestEncoding value="UTF-8" />
```

Contents

Web Development with Java EE

JavaBeans and JDBC

Thank you!