XSL-Preprocessing Layer for DbForms

1 Preface

Understanding this paper requires

- Basic knowledge of XML and XSL
  http://www.xml.org
- Basic knowledge of Java and JSP
  http://www.javasoft.com
- Familiarity with DbForms Framework
  http://www.wap-force.net/dbforms

Applying the methods described in this paper requires

- Servlet 2.2-compliant JSP-Container
  http://jakarta.apache.org/tomcat
- DbForms release 0.9 or higher
  http://www.wap-force.net/dbforms
- XSL transformation tool
  http://alphaworks.ibm.com (search for 'xsl')
  http://xml.apache.org/
  http://java.sun.com/xml/

2 Introduction

DbForms helps speeding up development of web based database applications (at least I hope so). But there are still lots of tedious (monotonous) tasks to do - you **still** have to write the **JSP code for each view**. The more views you have to write, the more monotonous work you have to do.

Because the JSPs are very similar to each other, you'll probably do much of the work using copy-and-paste and then apply the changes necessary by hand. This works well, but do we really want this kind of development? I don't think so!

For instance, if you have got a database containing 40 tables and you want to write a web based data maintenance tool against it. Imagine you want for each table a "list-view" (allowing the users to view all elements of the table at one time) and a "single-view" (viewing only 1 row at a time) so that the user may view and edit the data in 2 different ways. Well, having 40 tables and 2 views for each, implicates that you will have to create 40x2 = 80 jsp files, and a application-menu as well => this means 81 jsp files need to be written.

This is not only painful and boring work, it costs really lots of time. Imagine you need 30 minutes for each view (5 minutes for copying from another similar file, 10 minutes for adapting the jsp code, 15 minutes for bug tracking because code was not adapted correctly) then you will need to take 2410 minutes or 40 hours for this application!

In the following chapters we will examine a simple method which could shrink that 40 hours down to a few minutes! Of course you will gain such extreme productivity enhancements only if the JSP views follow a certain scheme – if the are template*able.* If every JSP requires its unique outfit and/or functionality, than the concept demonstrated in this paper will barely bring such great results!

3 The concept

Thanks to XML and XSL the solution I am going to describe is pretty simple and straightforward:

- For every "standard-layout" (ie. in our application we have got “listview-layout”, “singleview-layout” and “menu-layout”) we create **one XSL file**. Each XSL file gets applied to your application's XML-configuration file (this file contains detailed information about all tables and fields of the database[1])

- The result of the XSL transformations are JSP files, created according to the style rules defined in the XSL file.

Current implementation of this concept:

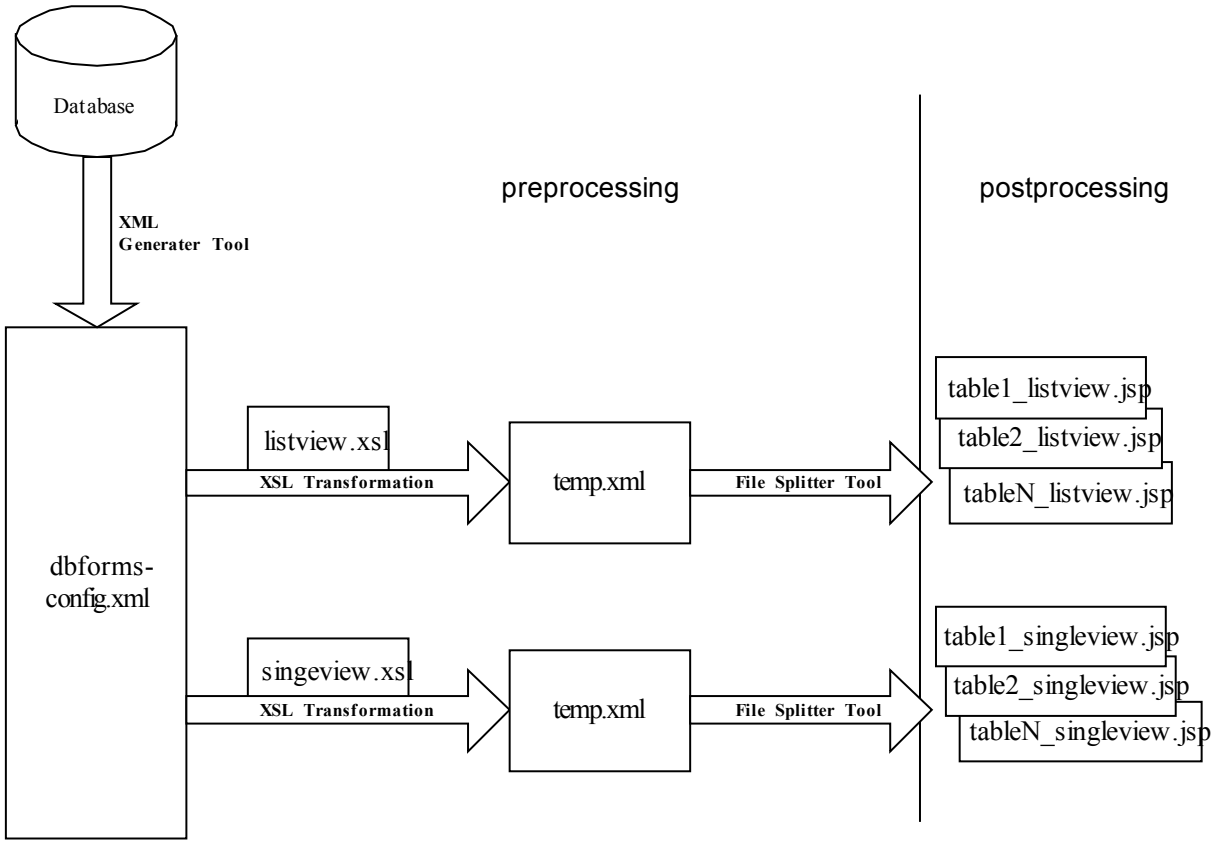

Figure 1 – chain of transformations that builds a DbForms-application – in the example shown 2 stylesheets are used. (The number of stylesheets is unlimited of course)

[1] since version 0.8 DbForms provides an utility-class "com.itp.dbforms.xmldb.DbTool" for automatic generation of the dbforms-config.xml config file [**check user's guide 0.8, Appendix III - "Application Configuration Generator" for more info!**]

Short description:

- First you need to create the XML configuration file "dbforms-config.xml" for DbForms– you can create it either by hand or you may try the XML generator tool for automatic generation (be sure to check the results in any case and apply corrections if needed!)
  Appendix I contains an example for such a configuration file.

- Next, you will need to define a XSL file for each template you want.
  Appendix II shows an example for such a stylesheet.

- Then you have to apply the XSL stylesheet(s) to the dbforms-config.xml file. You can use any XSL transformation tool (as long as it understands the XSLT grammar used by your stylesheets ;=). I am successfully using XSL editor from IBM alphaworks.
  Appendix III shows a screenshot of the XSL editor in action.

- After translation you get a more or less huge file containing all JSPs generated with the same template. You can either manually copy the files into new or existing JSPs or you can use the File-Splitter Tool included in DbForms (0.8 users: you will need to use the XSL-Tool-Patch until DbForms 0.8.5 or 0.9 will be released) This patch is available at the DbForms Project homepage.

- Finished! Now you may want to do some modifications to your generated JSPs - we all know that nothing can be standardised to 100%.

4 Future development

- More and better XSL stylesheets
- A simple User Interface for the whole process of transformations using one of the following techniques
  - implementing ANT-Task-Interface (http://jakarta.apache.org/ant)
  - designing a SWING-UI *[update April 29, 2001: done]*

The goal is to make the whole thing as easy to use as "form wizards" included in Microsoft Access or other RAD tools.

5 Feedback

Questions, hints, corrections, suggestions are welcome! Please do not hesitate to send an e-mail to joepeer@excite.com

Appendix I – A sample XML-config

Every DbForms-based application needs a configuration file describing the tables, fields and fieldTypes of the underlying database. See DbForms User's Guide (version 0.8 or higher) for more information!

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<dbforms-config>
	<table name="locations">
		<field name="id" fieldType="int" size="11" isKey="true"/>
		<field name="title" fieldType="varchar" size="50"/>
		<field name="address" fieldType="varchar" size="60"/>
		<field name="pcode" fieldType="varchar" size="10"/>
		<field name="country" fieldType="varchar" size="4"/>
		<field name="city" fieldType="varchar" size="20"/>

		<!-- add "granted-privileges" element for security constraints -->

	</table>

	<table name="messages">
		<field name="id" fieldType="int" size="11" isKey="true"/>
		<field name="username" fieldType="varchar" size="60"/>
		<field name="message_dump" fieldType="varchar" size="255"/>

		<!-- add "granted-privileges" element for security constraints -->

	</table>

	<table name="sessions">
		<field name="id" fieldType="int" size="11" isKey="true" autoInc="true" />
		<field name="sessiondate" fieldType="date" size="10"/>
		<field name="annotation" fieldType="varchar" size="100"/>
		<field name="loc" fieldType="int" size="11"/>

		<!-- add "granted-privileges" element for security constraints -->

	</table>

	<!-- ========== Connection ================================== -->
	<!--
	uncomment this if you have access to JNDI of an application server (see users guide for more info)
	<dbconnection
		name = "jdbc/dbformstest"
		isJndi = "true"
	/>
	-->

	<dbconnection
		name   = "jdbc:mysql://localhost/fashion"
		isJndi = "false"
		conClass  = "org.gjt.mm.mysql.Driver"
	/>
</dbforms-config>
```

Appendix II – A XSL-styleheet

The file works against a XML configuration file as shown in appendix I. This concreate stylesheet generates a “single view” . Appendix V shows a jsp generated with this template.

```
<?xml version='1.0'?>

<xsl:stylesheet xmlns:xsl='http://www.w3.org/XSL/Transform/1.0'>
<xsl:output indent="yes"/>

  <xsl:template match="table">

//--file "<xsl:value-of select="@name"/>_single.jsp" -------------------------------------

<xsl:text disable-output-escaping="yes">
&lt;%@ taglib uri="/WEB-INF/taglib.tld" prefix="db" %&gt;
</xsl:text>

  <html>
    <head>
      <db:base/>
      <link rel="stylesheet" href="dbforms.css"/>
    </head>
    <body>

      <db:style template="horizontalbar" part="begin" paramList="align='right'"/>
               <img src="img/pfeilchen.gif"/><a href="{@name}_list.jsp">[List]</a>
               <img src="img/pfeilchen.gif"/><a href="menu.jsp">[Menu]</a>
               <img src="img/pfeilchen.gif"/><a href="logout.jsp">[Log out]</a>
      <db:style template="horizontalbar" part="end"/>

     <db:dbform tableName="{@name}" maxRows="1" followUp="/{@name}_single.jsp"
autoUpdate="false">
       <db:header>
        <h1><xsl:value-of select="@name"/></h1>
       </db:header>
       <db:errors/>
       <db:body>
         <table border="0" align="center" width="400">

           <xsl:for-each select="field">

           <!-- we create input fields for NON-AUTOMATIC values only -->
           <xsl:if test="@autoInc!='true'">

           <xsl:variable name="bgcolor">
             <xsl:choose>
               <xsl:when test="position() mod 2 = 0">fee9aa</xsl:when>
               <xsl:otherwise>fee9ce</xsl:otherwise>
             </xsl:choose>
           </xsl:variable>


           <tr bgcolor="#{$bgcolor}">
             <td><xsl:value-of select="@name"/></td>
             <td>


               <xsl:choose>
                 <xsl:when test="@fieldType='int' or @fieldType='smallint'  or
@fieldType='tinyint'">
                    <db:textField fieldName="{@name}" size="{@size}" />
                 </xsl:when>
                 <xsl:when test="@fieldType='char' or @fieldType='varchar'  or
@fieldType='nvarchar'  or @fieldType='longvarchar'">
                   <xsl:choose>
                     <xsl:when test="@size>80">
                        <db:textArea fieldName="{@name}" cols="40" rows="3" wrap="virtual" />
                     </xsl:when>
                     <xsl:otherwise>
                        <db:textField fieldName="{@name}" size="{@size}" />
                     </xsl:otherwise>
```

```
                  </xsl:choose>
                </xsl:when>
                <xsl:when test="@fieldType='date'">
                  <db:textField fieldName="{@name}" size="{@size}" />
                </xsl:when>
                <xsl:when test="@fieldType='timestamp'">
                  <db:textField fieldName="{@name}" size="{@size}" />
                </xsl:when>
                <xsl:when test="@fieldType='double'">
                  <db:textField fieldName="{@name}" size="{@size}" />
                </xsl:when>
                <xsl:when test="@fieldType='float' or @fieldType='real'">
                  <db:textField fieldName="{@name}" size="{@size}" />
                </xsl:when>
                <xsl:when test="@fieldType='numeric'">
                  <db:textField fieldName="{@name}" size="{@size}" />
                </xsl:when>
                <xsl:when test="@fieldType='blob' or @fieldType='image'">
                  <db:file fieldName="{@name}" />
                </xsl:when>
                <xsl:when test="@fieldType='diskblob'">
                  <db:file fieldName="{@name}" />
                </xsl:when>
                <xsl:otherwise>
                *** error - fieldtype <xsl:value-of select="@fieldType"/> is currently not
supported. please send an e-mail to joepeer@wap-force.net ! ***
                </xsl:otherwise>
              </xsl:choose>


            </td>
          </tr>


          </xsl:if>

          </xsl:for-each>
        </table>
        <br/>
        <center><db:insertButton caption="Create new {@name}"/></center>

      </db:body>

      <db:footer>
        <table align="center" border="0">
          <tr>
            <td><db:navFirstButton caption="&lt;&lt; First" style="width:90"/></td>
            <td><db:navPrevButton caption="&lt; Previous" style="width:90"/></td>
            <td><db:navNextButton caption="Next &gt;" style="width:90"/></td>
            <td><db:navLastButton caption="Last &gt;&gt;" style="width:90"/></td>
          </tr>
        </table>
        <table align="center" border="0">
          <tr>
            <td><db:updateButton caption="Update" style="width:90"/></td>
            <td><db:deleteButton caption="Delete" style="width:90"/></td>
            <td><db:navNewButton caption="New" style="width:40"/></td>
          </tr>
        </table>
      </db:footer>

    </db:dbform>
   </body>
  </html>
 </xsl:template>
</xsl:stylesheet>
```

Important: did you notice the following line?

```
//--file "<xsl:value-of select="@name"/>_single.jsp" -------------------------------------
```

This line serves as **separation mark** for the tool that cuts a single transformation result file into multiple JSP files. It is important that this separation line starts with the string “//--file “.

This string should be followed by double quotes, the name of the JSP file and then double qutotes again. The trailing “---“ characters are optional.

```
Valid examples:

//--file "<xsl:value-of select="@name"/>_single.jsp" ------------------------------------
//--file "<xsl:value-of select="@name"/>_test.jsp"
//--file "foofoo_<xsl:value-of select="@name"/>_barbar.jsp"
```

Appendix III – XSL transformation using IBM's XSL Editor

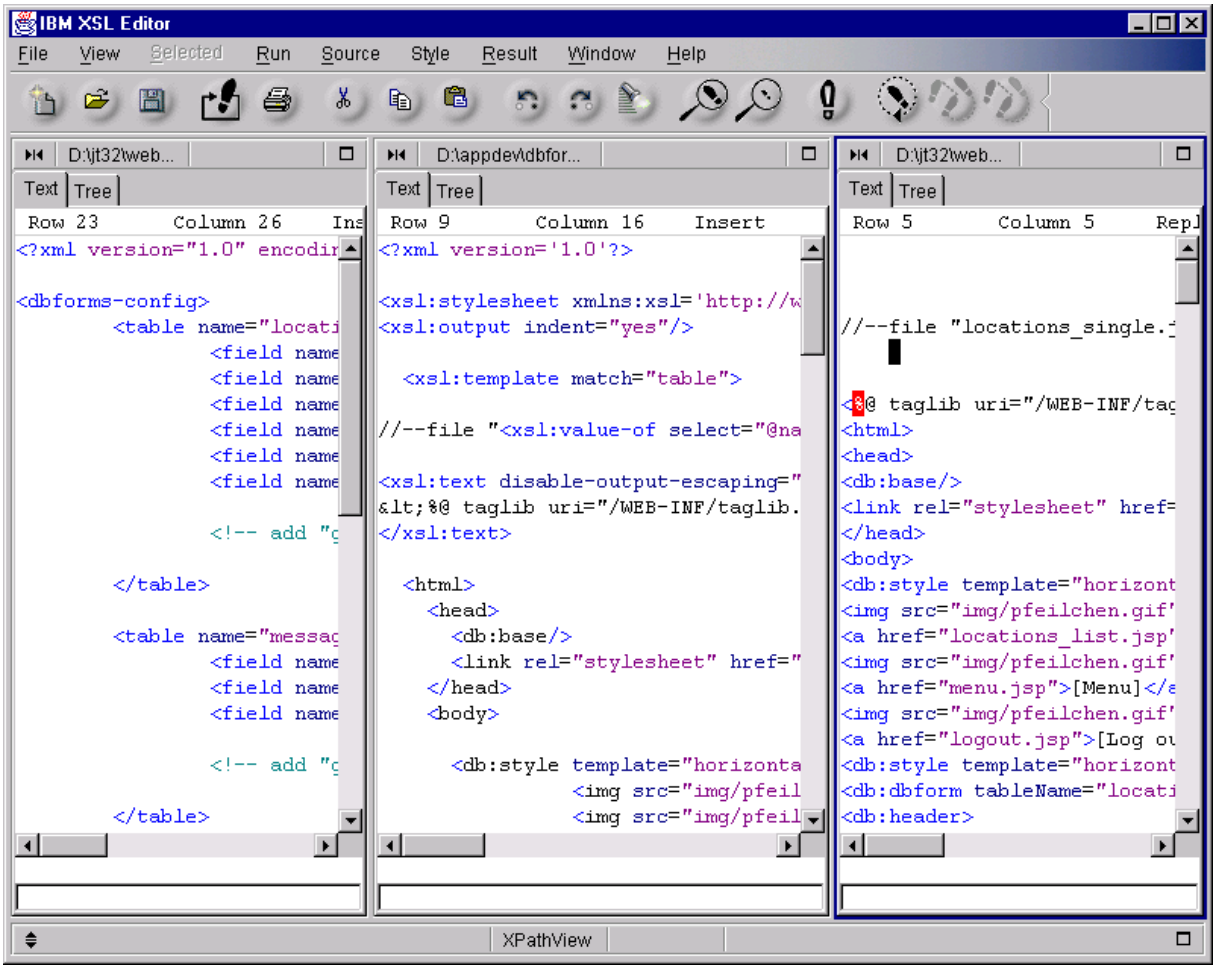

Figure 2 – XSL transformation in action

This screenshot shows three code-windows:

- Left window: the XML configuration data for DbForms (this is the "XML source" for the transformation) – as shown in Appendix I
- Middle window: the XSL template (stylesheet) - as shown in Appendix II
- Right window: the transformation result - as shown in Appendix V

Appendix IV – Using the File Splitter Tool

Included in upcoming versions of DbForms as well as in the **XSL-Tool-Patch** is a trivial but still useful tool for splitting up the temp-files containing the JSPs generated by a XSL template.

Usage:

```
java com.itp.dbforms.FileSplitter sourceFile destinationDirectory
```

Imagine we have done 3 tranformations (using 3 different stylesheets) and we have generated the files lists.tmp, menu.tmp and singles.tmp then we will execute the following lines:

```
java com.itp.dbforms.xmldb.FileSplitter d:\jt\webapps\ft_wi\lists.tmp d:\jt32\webapps\ft_wi\
java com.itp.dbforms.xmldb.FileSplitter d:\jt\webapps\ft_wi\menu.tmp d:\jt32\webapps\ft_wi\
java com.itp.dbforms.xmldb.FileSplitter d:\jt\webapps\ft_wi\singles.tmp d:\jt32\webapps\ft_wi\
```

To get this working make sure that the correct version of dbforms.jar is in your classpath (v0.8 will NOT work, you will need to download the xsl patch)

Appendix V - Results

After transformations and file-splitting you have got a JSP file for each table (or table collection) and each stylesheet; like the following:

```
<%@ taglib uri="/WEB-INF/taglib.tld" prefix="db" %>
<html>
<head>
<db:base/>
<link rel="stylesheet" href="dbforms.css"/>
</head>
<body>
<db:style template="horizontalbar" part="begin" paramList="align='right'"/>
<img src="img/pfeilchen.gif"/>
<a href="locations_list.jsp">[List]</a>
<img src="img/pfeilchen.gif"/>
<a href="menu.jsp">[Menu]</a>
<img src="img/pfeilchen.gif"/>
<a href="logout.jsp">[Log out]</a>
<db:style template="horizontalbar" part="end"/>
<db:dbform tableName="locations" maxRows="1" followUp="/locations_single.jsp"
autoUpdate="false">
<db:header>
<h1>locations</h1>
</db:header>
<db:errors/>
<db:body>
<table border="0" align="center" width="400" class="xmp">
<tr bgcolor="#fee9ce">
<td>id</td>
<td>
<db:textField fieldName="id" size="11"/>
</td>
</tr>
<tr bgcolor="#fee9aa">
<td>title</td>
<td>
<db:textField fieldName="title" size="50"/>
</td>
</tr>
<tr bgcolor="#fee9ce">
<td>address</td>
<td>
<db:textField fieldName="address" size="60"/>
</td>
</tr>
<tr bgcolor="#fee9aa">
<td>pcode</td>
<td>
<db:textField fieldName="pcode" size="10"/>
</td>
</tr>
<tr bgcolor="#fee9ce">
<td>country</td>
<td>
<db:textField fieldName="country" size="4"/>
</td>
</tr>
<tr bgcolor="#fee9aa">
<td>city</td>
<td>
<db:textField fieldName="city" size="20"/>
</td>
</tr>
</table>
<br/>
<center>
<db:insertButton caption="Create new locations"/>
</center>
</db:body>
<db:footer>
<table align="center" border="0">
<tr>
<td>
<db:navFirstButton caption="&#60;&#60; First" style="width:90"/>
```

```
</td>
<td>
<db:navPrevButton caption="&#60; Previous" style="width:90"/>
</td>
<td>
<db:navNextButton caption="Next &#62;" style="width:90"/>
</td>
<td>
<db:navLastButton caption="Last &#62;&#62;" style="width:90"/>
</td>
</tr>
</table>
<table align="center" border="0">
<tr>
<td>
<db:updateButton caption="Update" style="width:90"/>
</td>
<td>
<db:deleteButton caption="Delete" style="width:90"/>
</td>
<td>
<db:navNewButton caption="New" style="width:40"/>
</td>
</tr>
</table>
</db:footer>
</db:dbform>
</body>
</html>
```

And this is how the concreate JSP will look like in a user's browser – and we achieved this without programming!

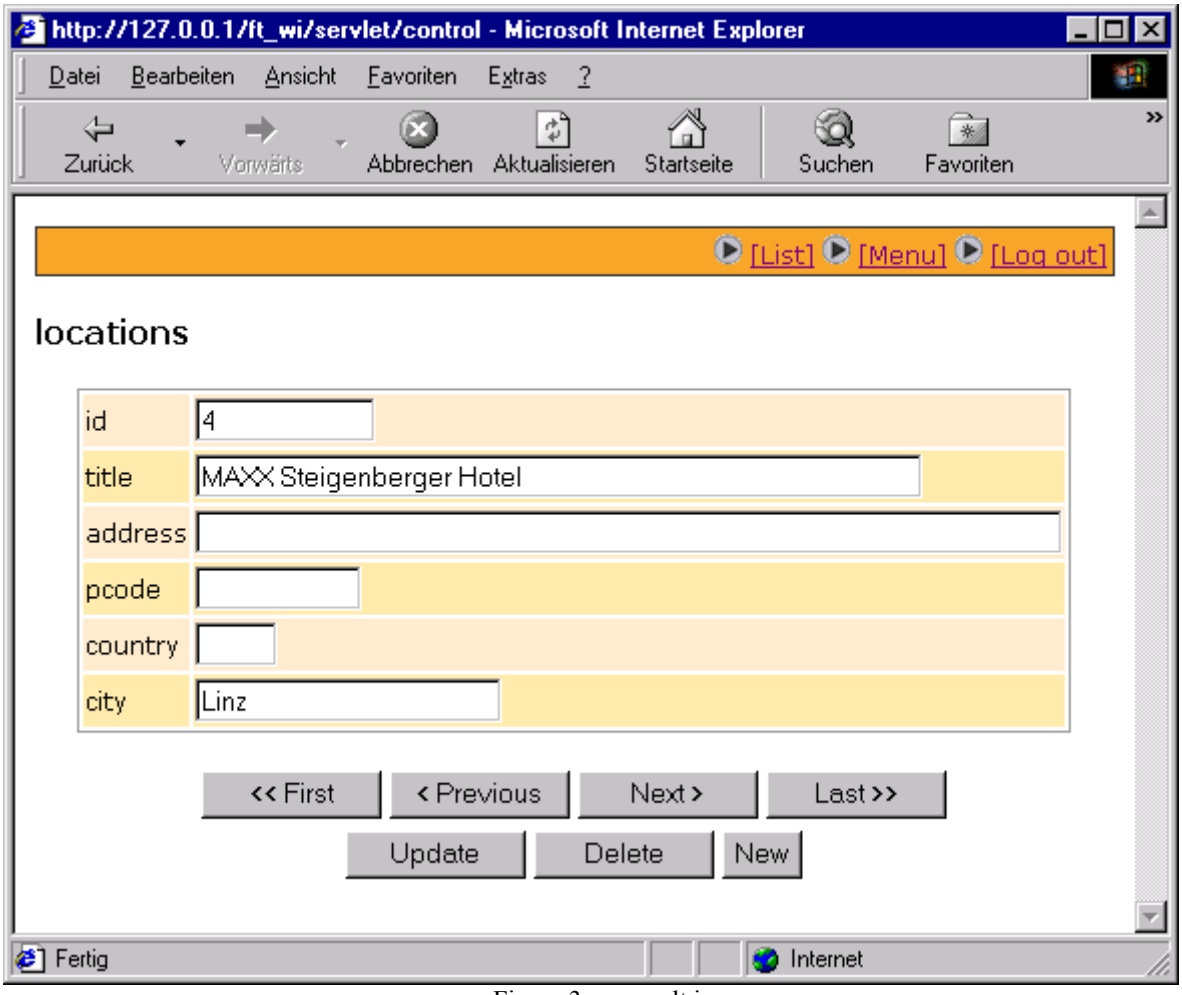

Figure 3 – a result jsp