INSTITUTIONEN FÖR DATA-
OCH SYSTEMVETENSKAP
SU / KTH

DATABASES & INTERNET LABORATION

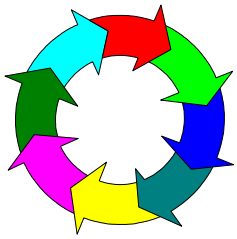
v. 1.1

*63

RELATIONSDATABASHANTERINGSSYSTEM

HÖSTTERMINEN 1999

http://L238.dsv.su.se/courses/stjarna63/



nikos dimitrakas

Table of contents

Table of figures

1 Introduction to the Environment

In this exercise (described in chapter 2 & 3) we will build a little web application that provides the possibility to on-line retrieve and manipulate data stored in a database. We will provide the user with some simple database operations:

- Basic DML commands such as:
  - SELECT
  - INSERT
  - UPDATE
  - DELETE

We will try to do all this by using two different techniques:
- ASP (Active Server Page)
- Servlets and JSP (Java Server Page)

To do all this we will use the following products (they are all running under Windows NT, equivalent software may exist for other operating systems):

Netscape Enterprise Server 2.6	This is a web server product that provides the possibility to publish the applications on the web.
Chili!Soft ASP 3.0	This is a plug-in that activates the web server for ASP. Without this plug-in the ASPs are just text files for the web server.
IBM WebSphere Application Server 2.03	This is the application server that takes care of servlets, compiles jsp-files, creates and manages sessions and database connections. WebSphere Application Server is connected to the Netscape web server.
IBM WebSphere Studio 1.0	This tool provides help for designing servlets, javabeans, queries, and connections between all these. WebSphere Studio generates some of the code which can be completed in an editor of your choice. NetObjects Scriptbuilder is the default editor that is installed with WebSphere Studio.
Microsoft Access 97	This is the database manager that has been used to create the database used in the exercises in the following chapters.
ODBC Data Source Administrator	This is the “bridge” between the database and the ASPs and Servlets. ODBC Data Source Administrator can be found in the Control Panel.

The following figure illustrates how the web server, the application server, the ASP plug-in, and the database are connected to each other and to the internet. The web server is in charge of receiving requests from clients and then forwarding them to the application server, which is in charge of servlets and JSPs, or to the ASP plug-in which is in charge of ASPs.

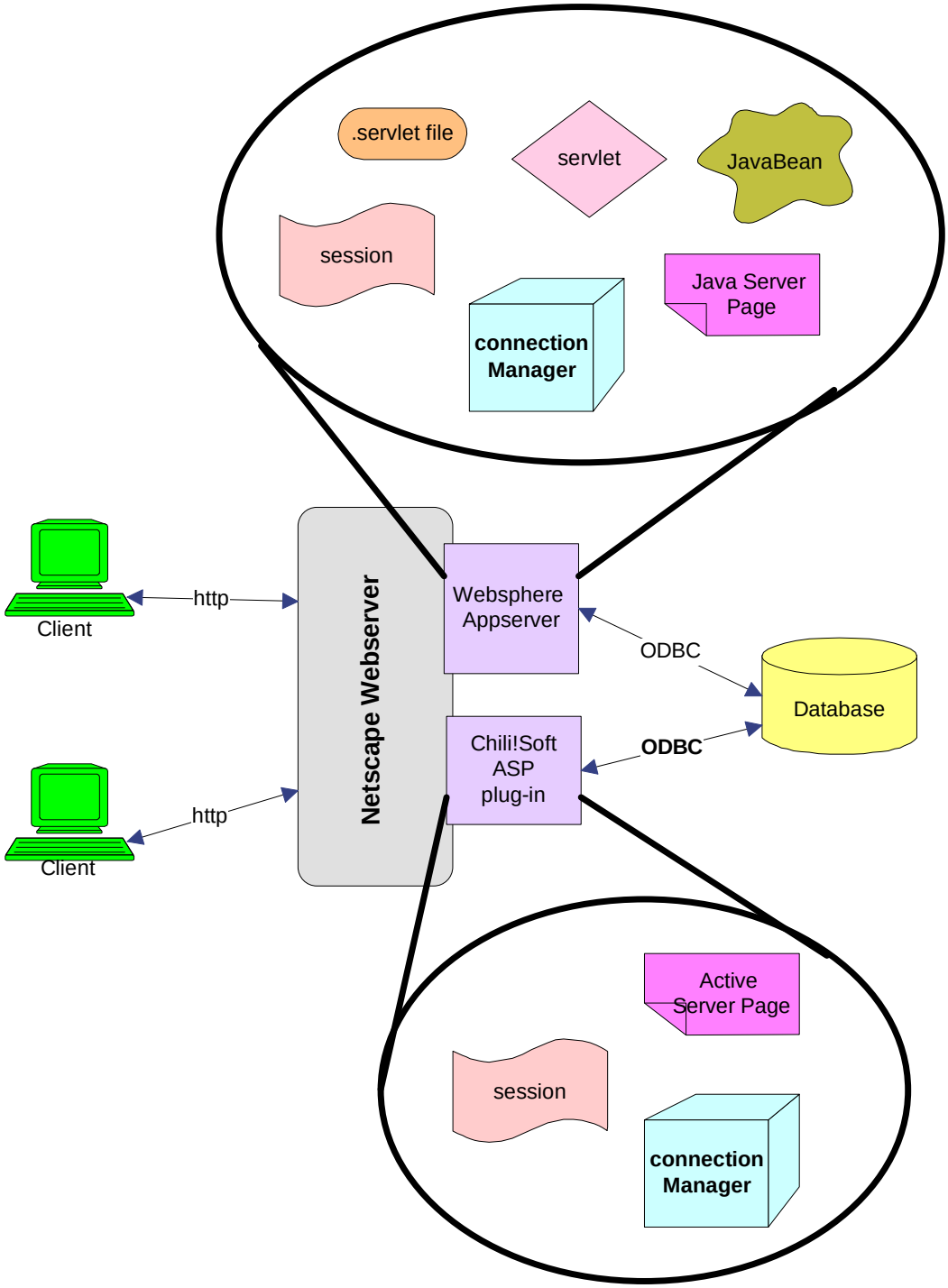
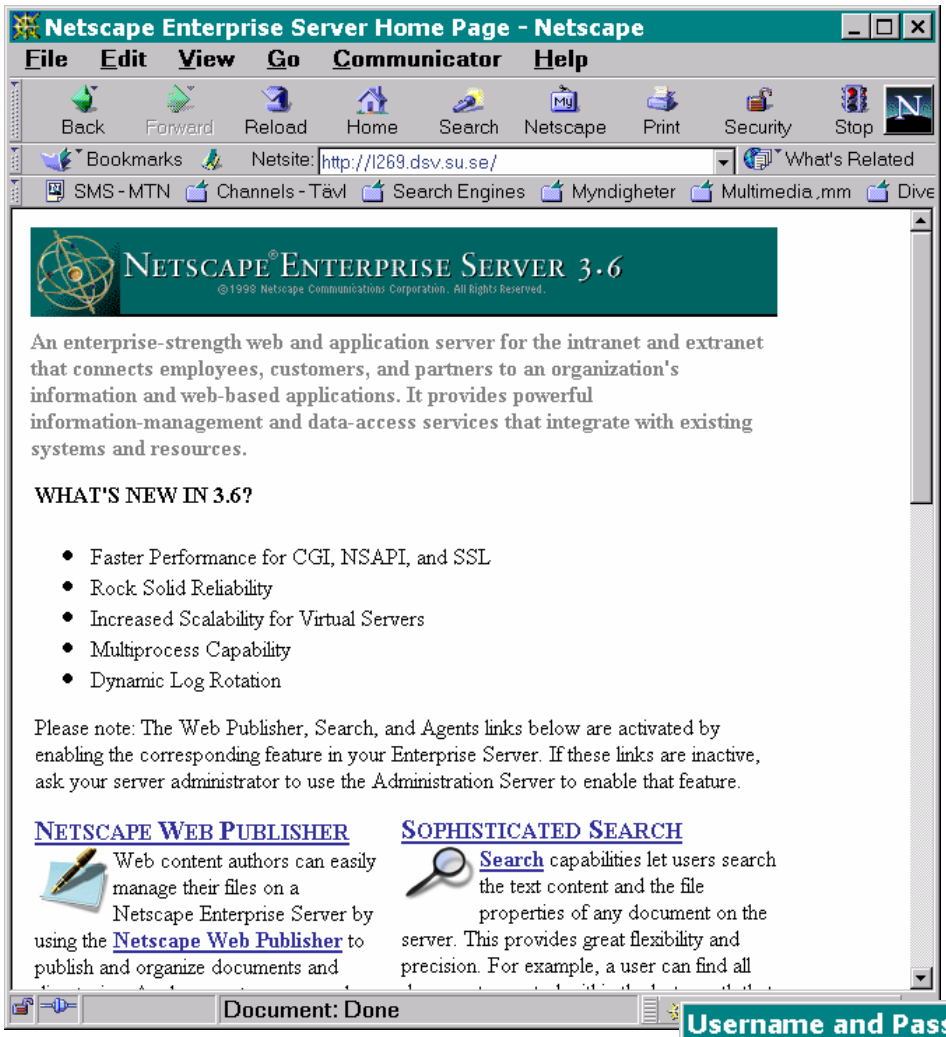


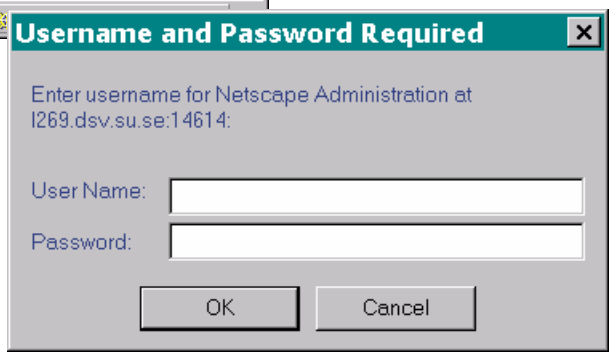
Figure 1 Environment components and their role

1.1 Netscape Server Enterprise

On every machine (that is used for this course) there is a Netscape web server installed (normally under c:\Netscape\SuiteSpot). This means that the machine listens on port 80 for http requests. You can test this by starting a browser and trying the following URL: http://localhost/ or http://%nameofmachine%[1] for example http://L269.dsv.su.se/

If the browser returns an error message then the web server may be stopped. If the server is running the browser should show the default homepage:



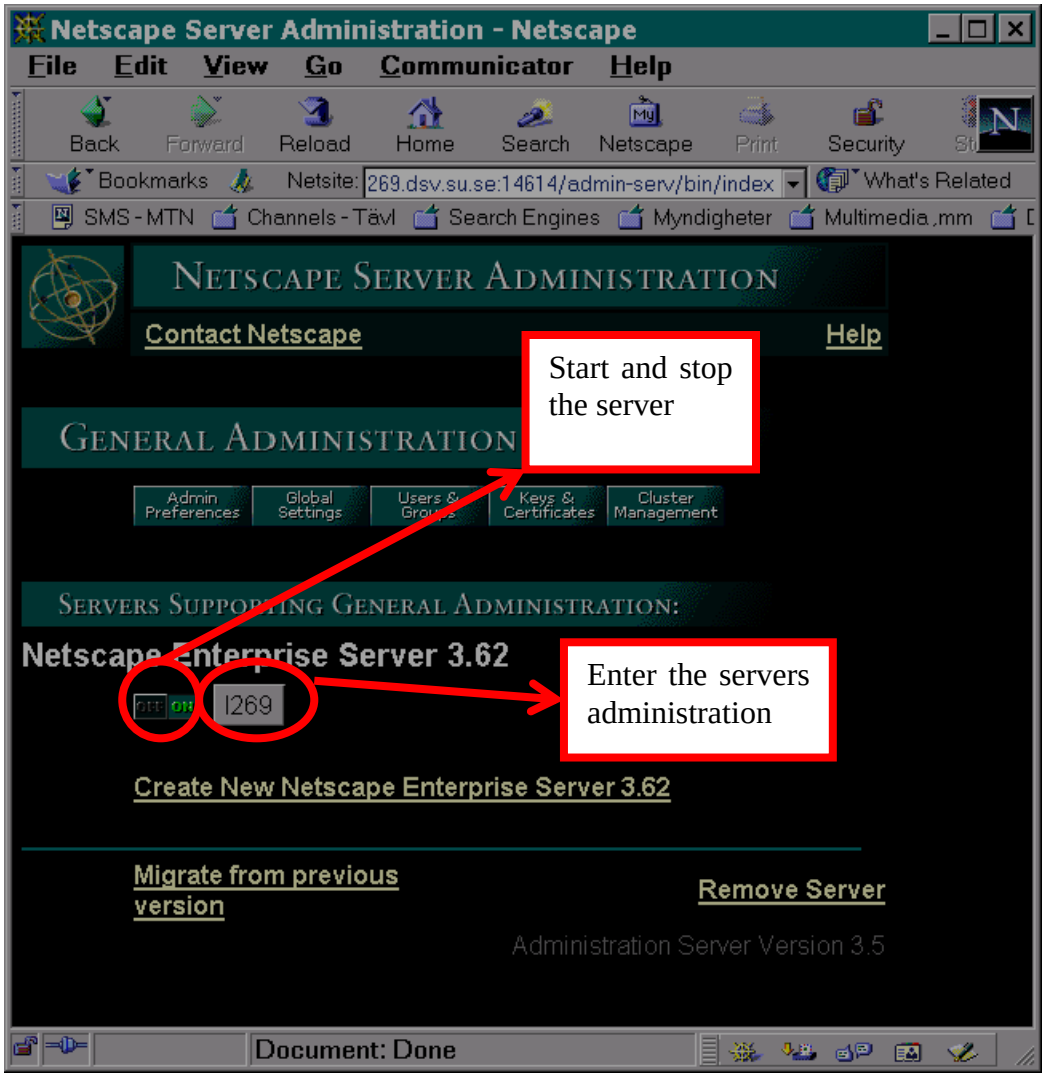


The web server can be administered on port 14614:
For example: http://l269.dsv.su.se:14614/

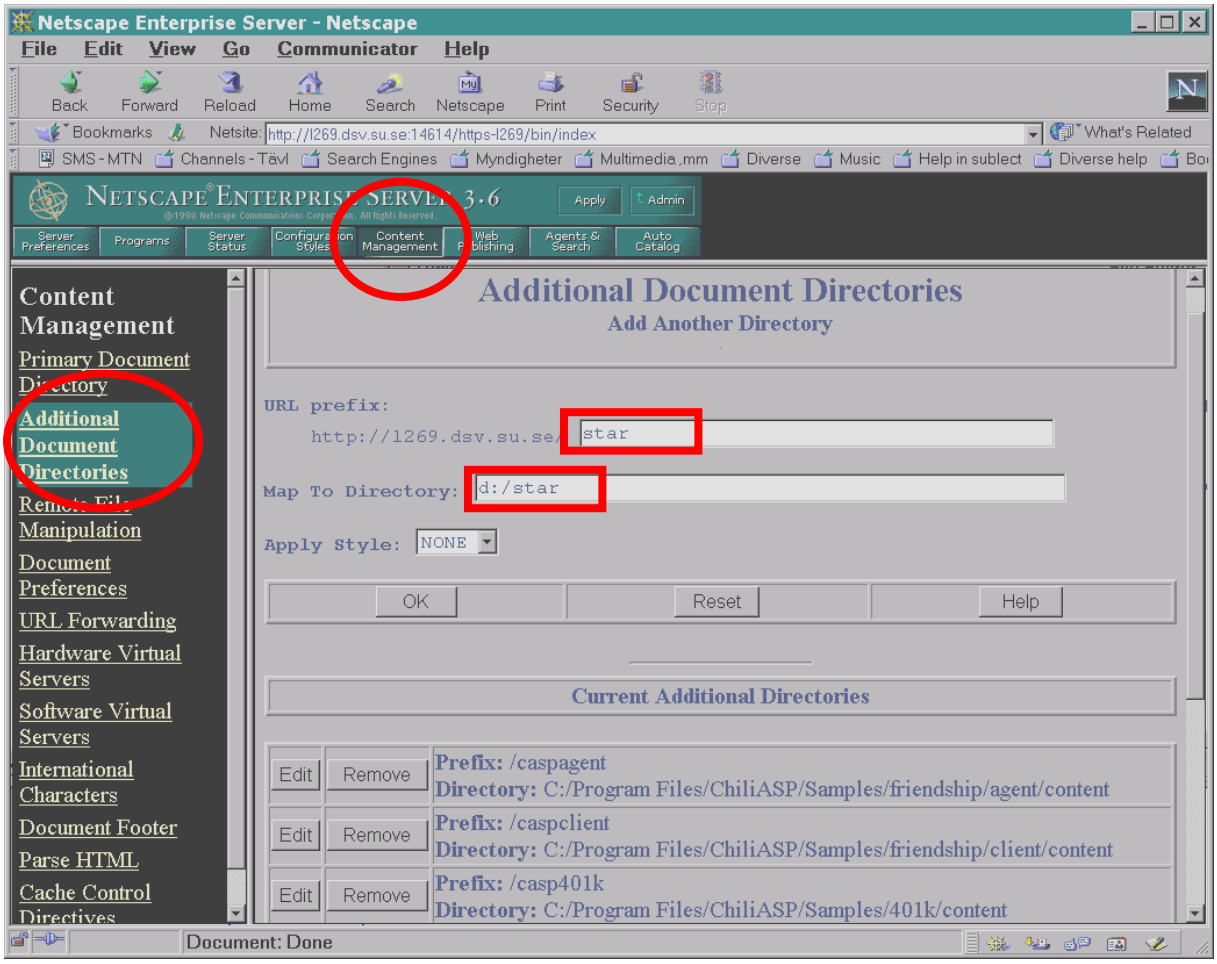
username: ***admin***
password: ***admin***

[1] **The name of the machine is usually "L" and a three digit number and it is written on the front part of the computer tower.**

From here you can start and stop the web server.

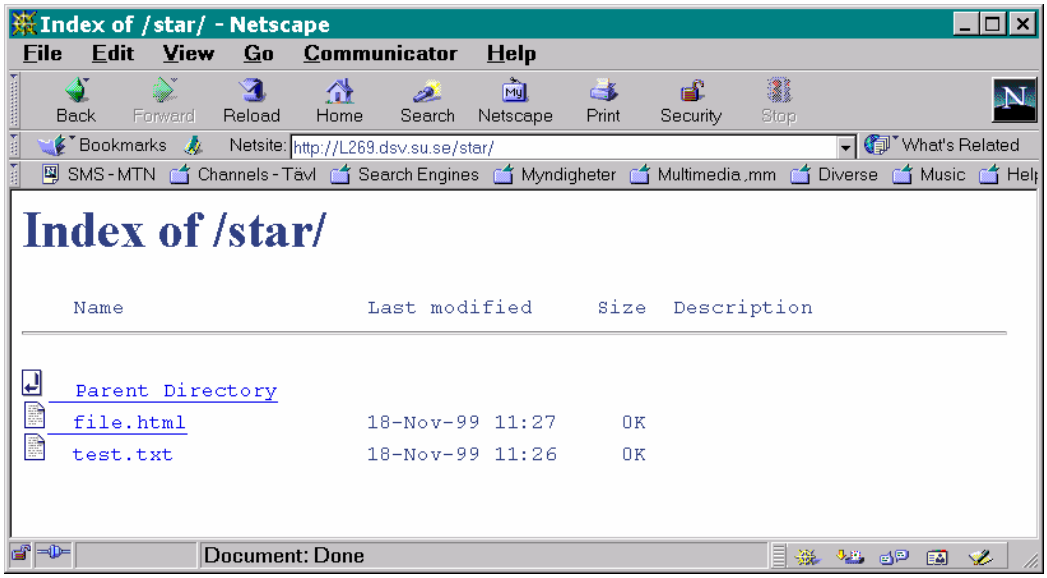


An other very useful feature here is the possibility to create aliases for directories or files so that they can be accessed through http. For example we can create an alias (URL prefix) called star and map it to the d:\star directory:



After pressing OK and applying the changes it will be possible to see the contents of the d:\star directory with a web browser at the following URL:

http://L269.dsv.su.se/star/



1.2 WebSphere Application Server

To every Netscape Server there is a WebSphere Application Server attached. The WebSphere Appserver is located under e:\WebSphere\AppServer. The servlets that we will create later have to be deployed on the WebSphere application server. WebSphere searches automatically for .class (compiled java classes) and .servlet (servlet configuration) files in the e:\WebSphere\Appserver\classes directory and the e:\WebSphere\Appserver\servlet directory.

WebSphere Application Server can be administered at port 9527:
http://localhost:9527

username: ***admin***
password: ***admin***

Useful information about WebSphere Application Server is also available at:
http://localhost:9527/doc/index.html

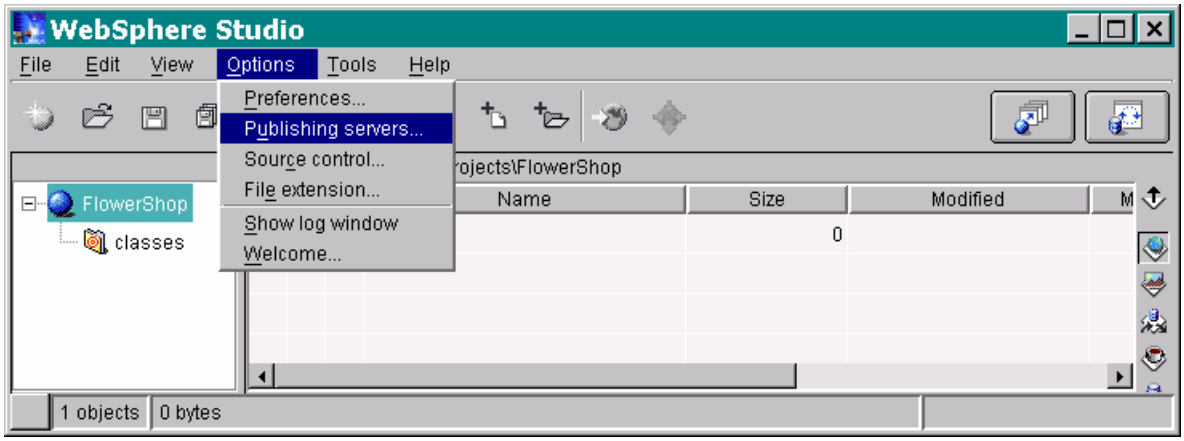
1.3 Chili!Soft ASP

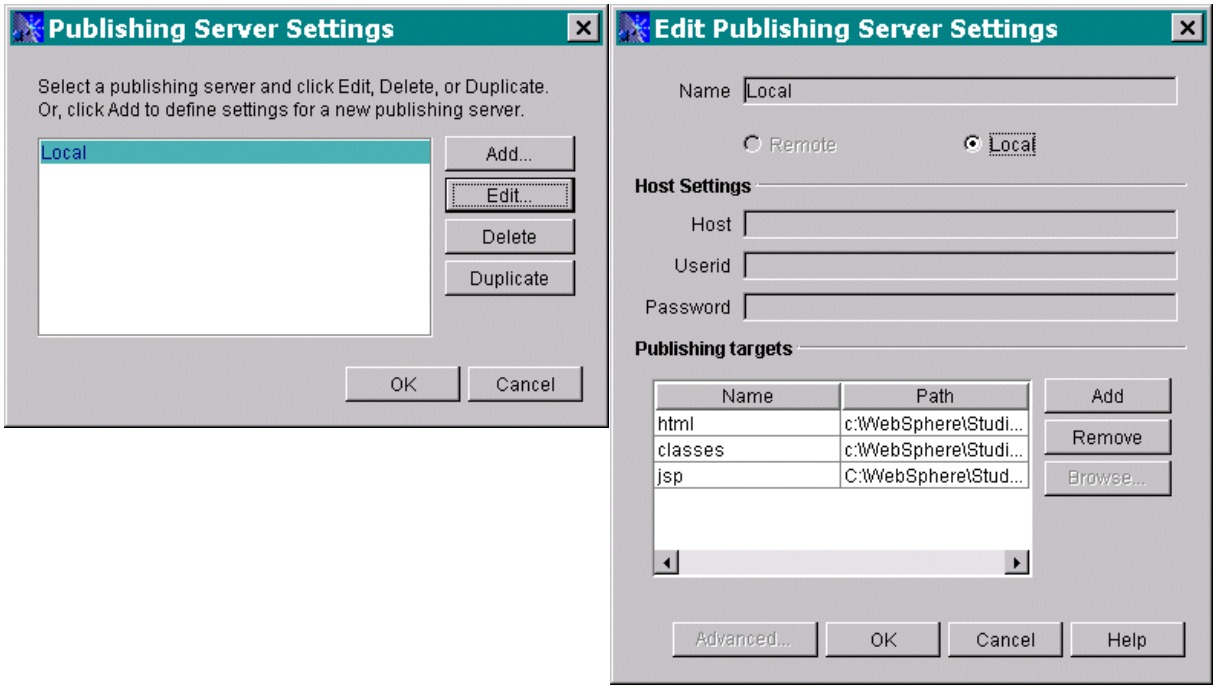
Chili!Soft ASP is a plug-in that is also attached to the web server. This product does not require any configuration.

For more information about Chili!Soft ASP:
http://localhost/caspdoc/Index.html

1.4 WebSphere Studio

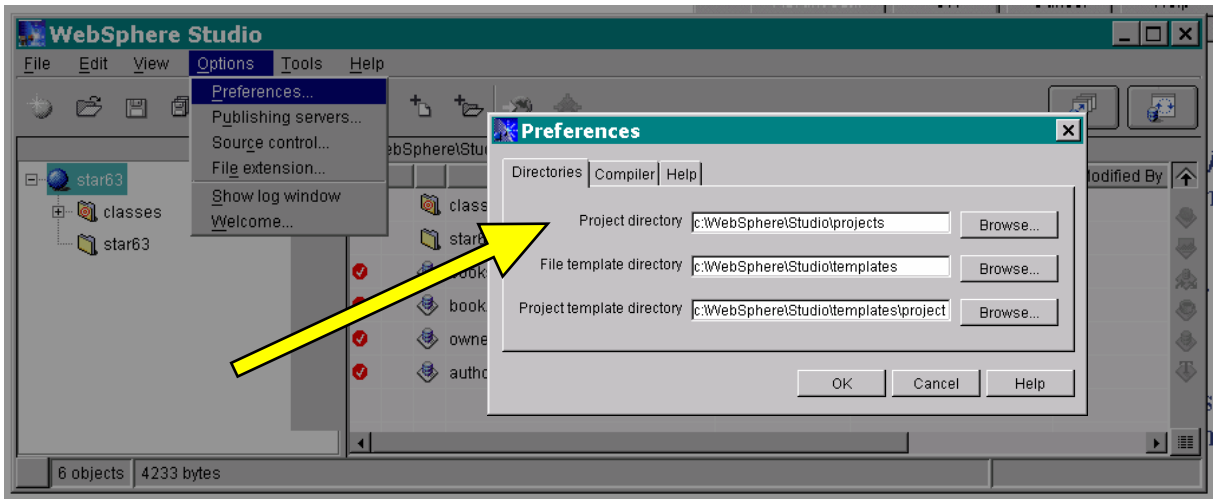
WebSphere Studio is a tool for creating most of the necessary files for web applications based on servlets and JSPs. Studio can also be configured so that it can "publish" all the publishable files (what it basically does, is copy them) to predefined directories where WebSphere Application Server and Netscape Server can find them. That can be set in the Options > Publishing servers menu:





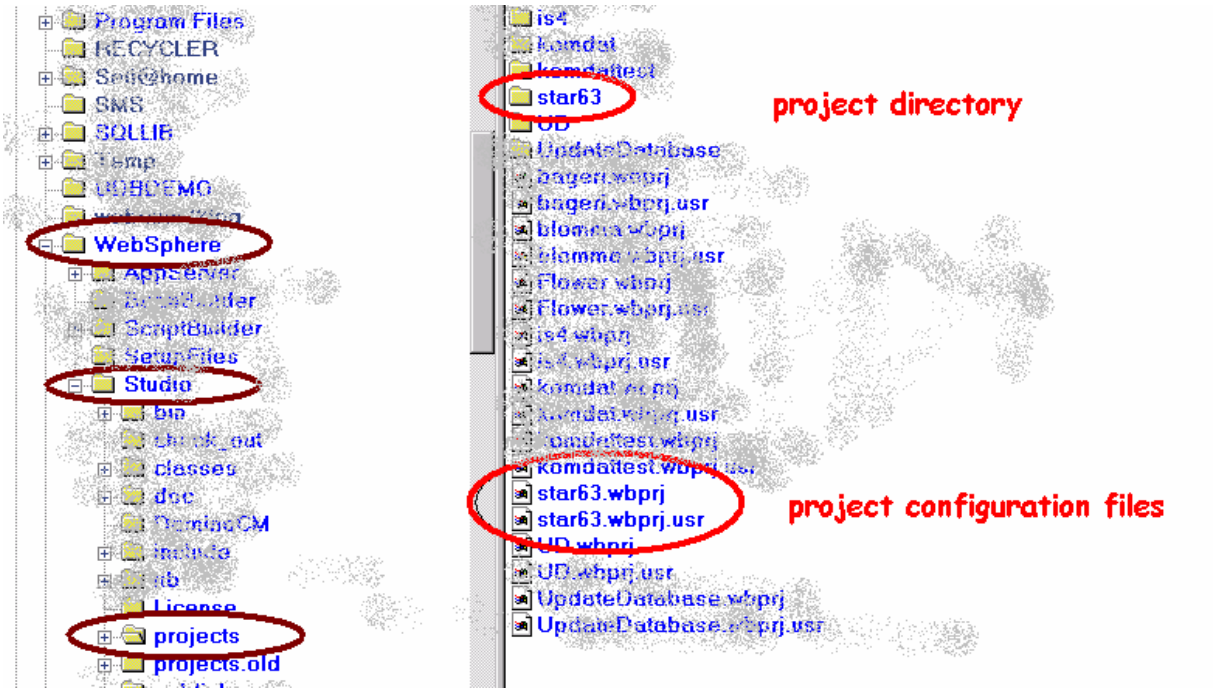
Here you can define the "publish"-paths for different types of files[2].
All classes and servlet files should be published to c:\Websphere\Appserver\classes
All html, jsp, gif, jpg etc. should be published either to c:\Netscape\Suitespot\docs *(which is the default document root for the web server*) or to a directory that you have mapped with an alias from the web server (see section 1.1 on page 5).

WebSphere works with projects. Every project is saved in a directory with the name of the project under c:\Websphere\Studio\projects. This location can be redirected:



[2] **Notice that a whole folder is assigned to a publishing location and that all the publishable files are published to the associated location. You don't need to worry so much about the publishing, Websphere Studio is quite good at publishing the right files to the right place.**

Together with the directory where the project files are located, there is two more configuration files:



You can back-up your project by copying these three items (project directory & project configuration files).

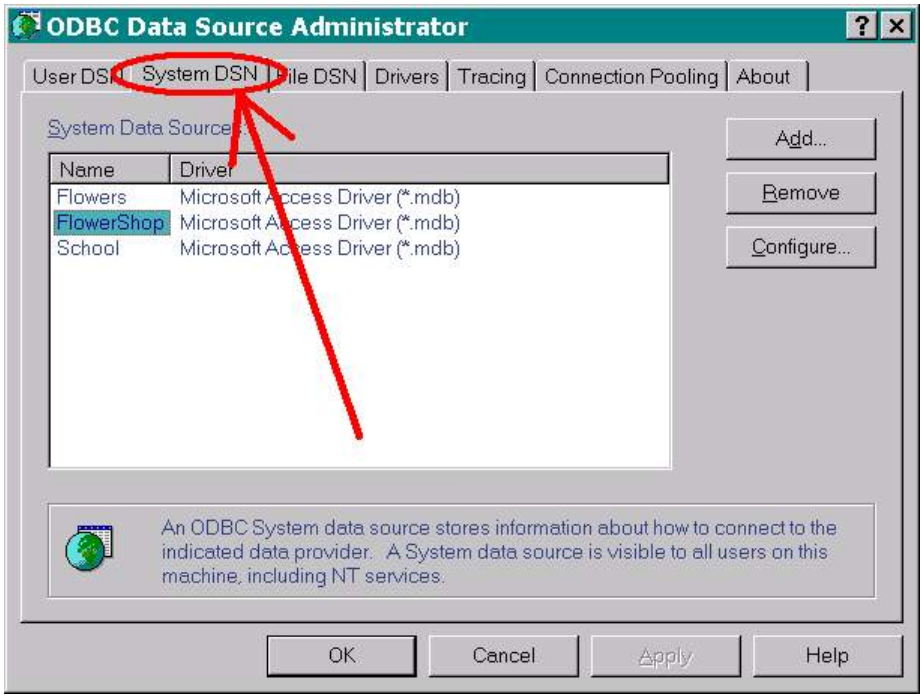
WebSphere Studio also provides Wizards that guide you through the most common steps of creating JavaBeans, servlets, SQL statements, html and jsp files. We will explore these features in more detail later.

The documentation of WebSphere Studio can be accessed with a web browser at:
c:/WebSphere/Studio/DOC/html/2tabcontents.html

1.5 Other tools

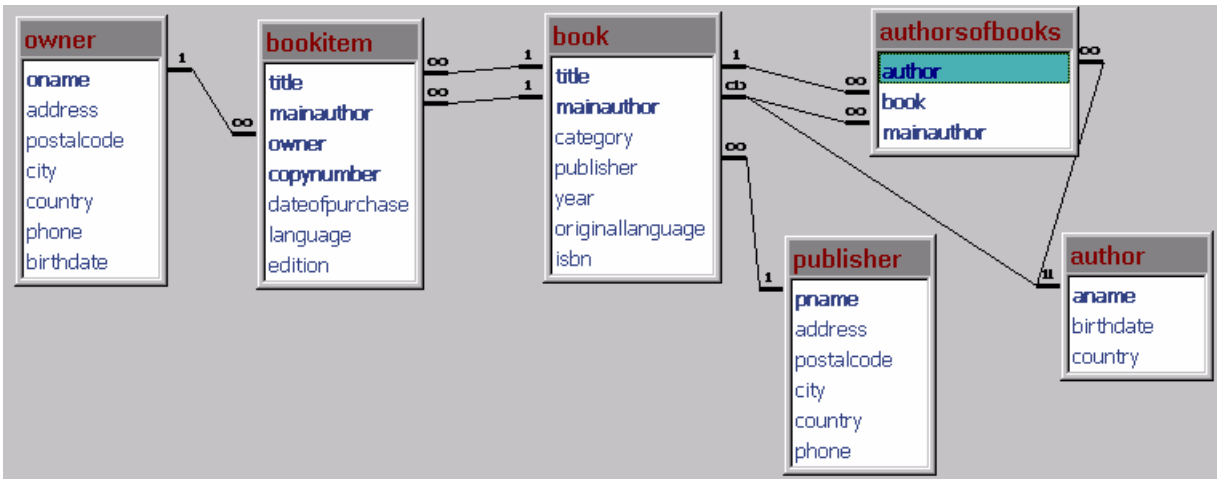
Other relevant tools are:

- **ScriptBuilder** – for editing .html, .jsp, .java, .servlet, .asp files
  ScriptBuilder provides some support on those types of files.
- **MS Access** – to explore and alter the database
  The database used in the following exercises can be downloaded from
  - http:\\L238.dsv.su.se\courses\stjarna63 or
  - \\DB-SRV-1\StudKursInfo\x63 Ht1999\ASP-Servlet Laboration
- **ODBC Data Source Administrator**
  For creating an ODBC alias for a database so that servlets and ASPs can access the database. The ODBC Data Source Administrator can be opened from the control panel. When you register an alias make sure to place it under the System DSN tab, so that the servlets and ASPs can find it:



2 Database

In chapter 3 and 4 we will build two small applications that connect to a database with information about books, authors and book owners. This database is built in MS Access and includes some test data. The following figure illustrates the tables of the database as well as the relationships between them:



Owner Includes data about people that own books

Publisher Includes data about publishers

Author Includes data about authors

Book Includes data about books. A book is identified by its name and main author.

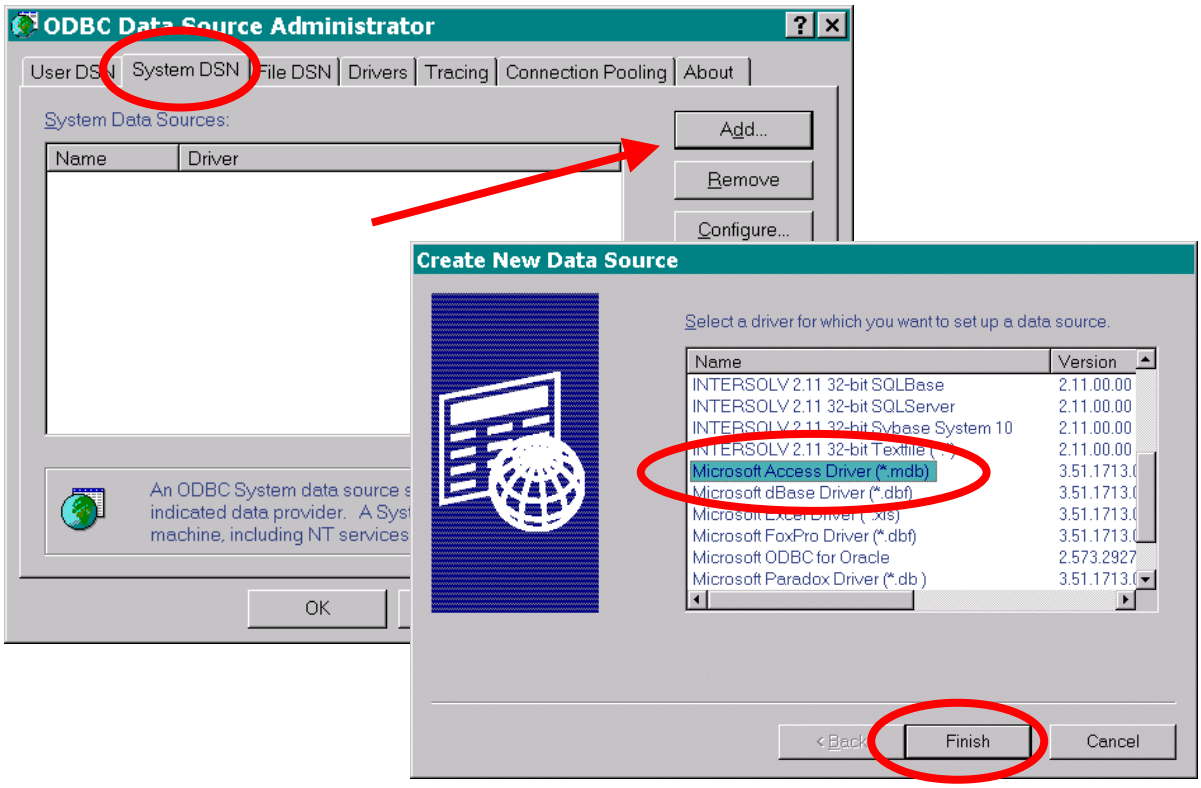
AuthorsOfBook Includes data about additional authors of books. This table has been added in order to remove the many-to-many relationship between books and authors.

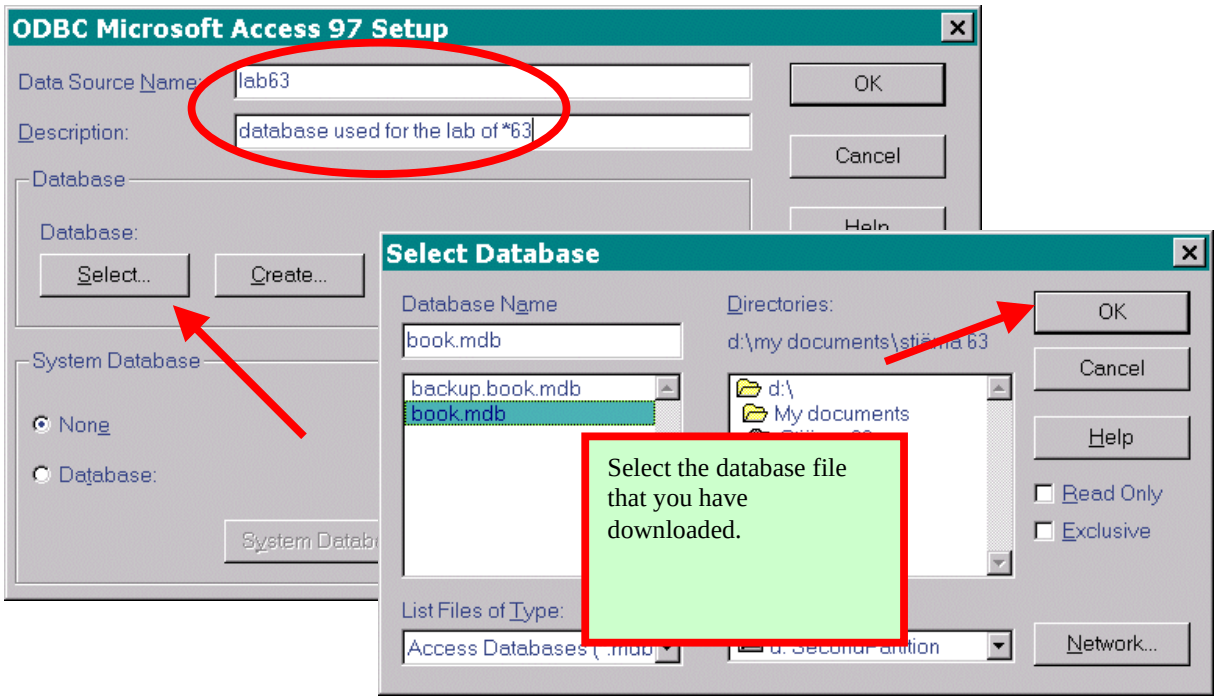
BookItem Here we have all the copies of a certain book with their attributes (that differ from copy to copy) for example their owner.

2.1 Connect a database

To use the database in our web applications we need to do the following things:

1. Download the database from
   - http:\\L238.dsv.su.se\courses\stjarna63 or
   - \\DB-SRV-1\StudKursInfo\x63 Ht1999\ASP-Servlet Laboration
2. Create an ODBC alias (also known as DSN) in the ODBC Data Source Administrator.





Now the database is available through an ODBC driver.

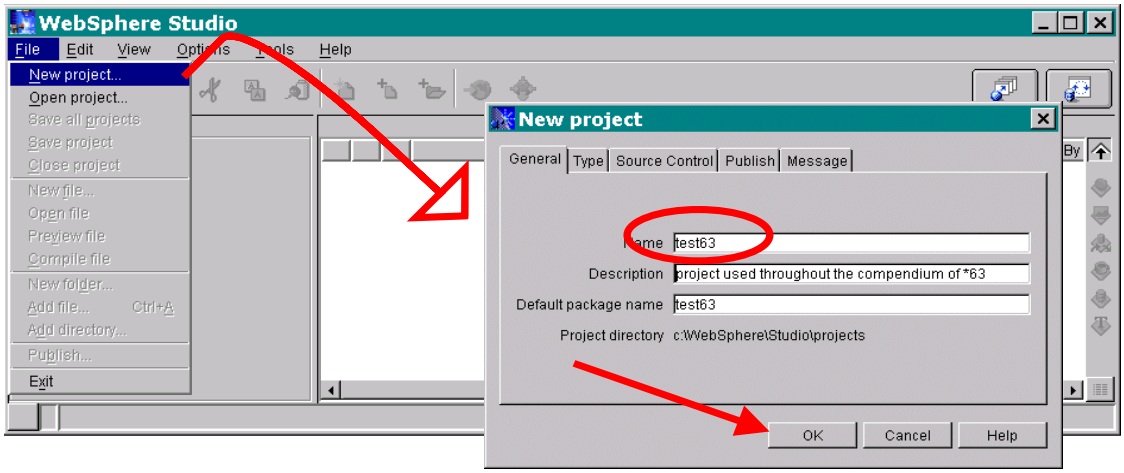
3 Servlets

In this section we will build a project with servlets and JSPs. The following two functions will be available:

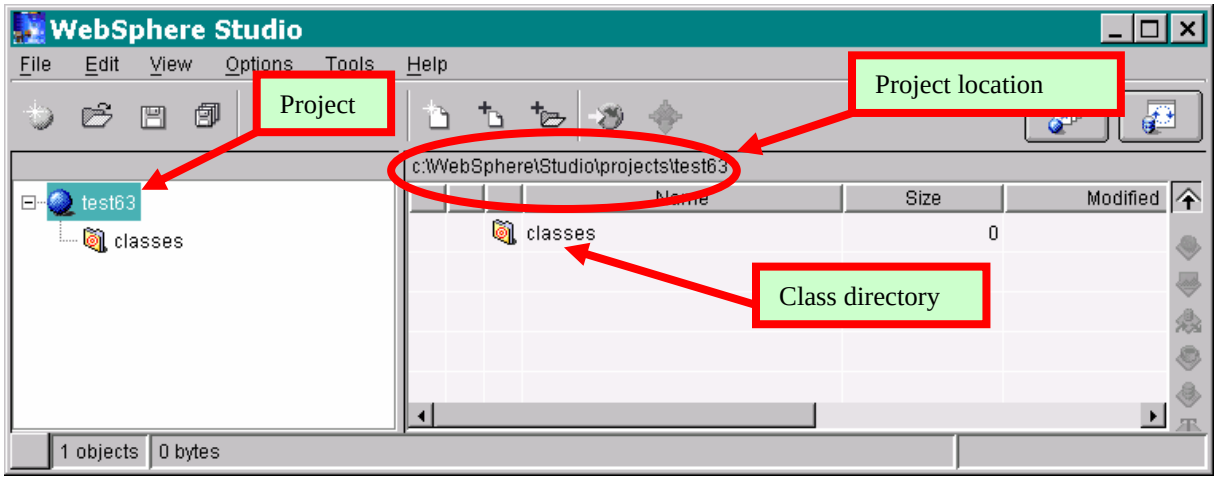
- List all the books (title, mainauthor, isbn) and their publisher (name and country) ordered by publisher, author and book title.
- Update the information of an author.

To do that we will use WebSphere Studio and its Wizards.

We can start by creating a new project in WebSphere Studio.



The name of the project used throughout this compendium is *test63*. We will use this project as a container for all the servlets and JSPs that we will create. When the new project is created, WebSphere Studio automatically creates a directory for the entire project and in that directory another directory for all the classes.



3.1 Using the SQL Wizard

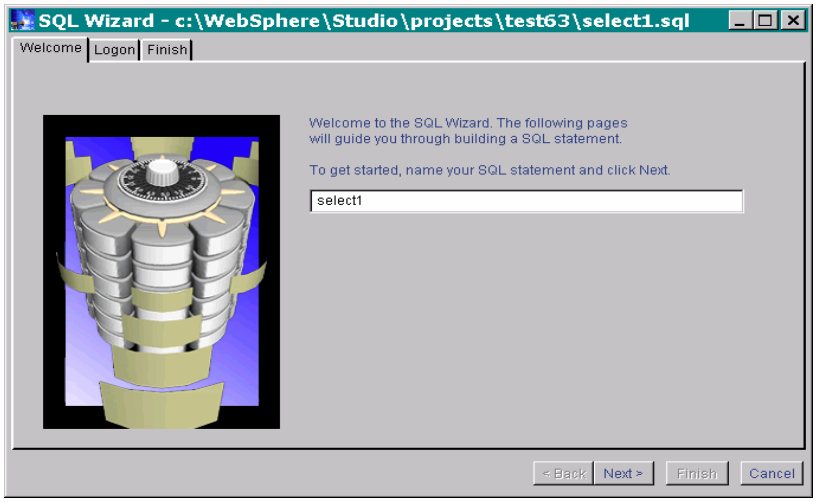
When you want to create a servlet that provides database functionality, you first have to specify an SQL statement and some database configuration parameters (user-name, password, driver…). The SQL Wizard provides that.

✓ Create a servlet that returns a table with the results of the following request:
List all the books (title, mainauthor, isbn) and their publisher (name and country) ordered by publisher, author and book title.

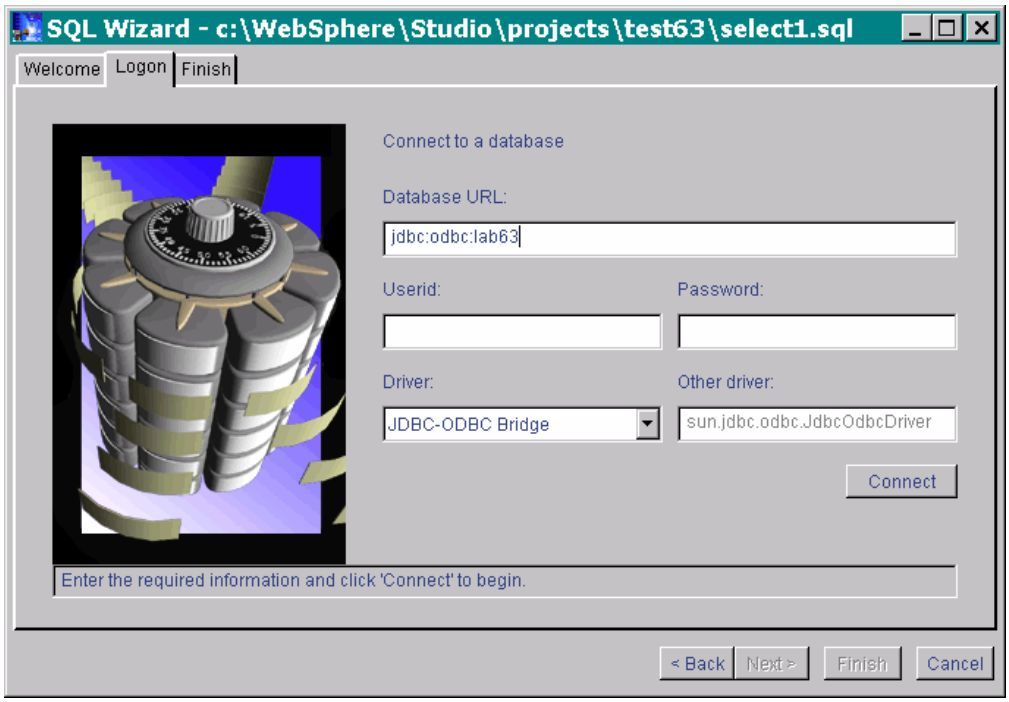
To do that we can use the SQL Wizard to create an SQL statement that complies with that request.

➢ Start the SQL Wizard! (*Tools* > *SQL Wizard* or *the button on the right top side*)

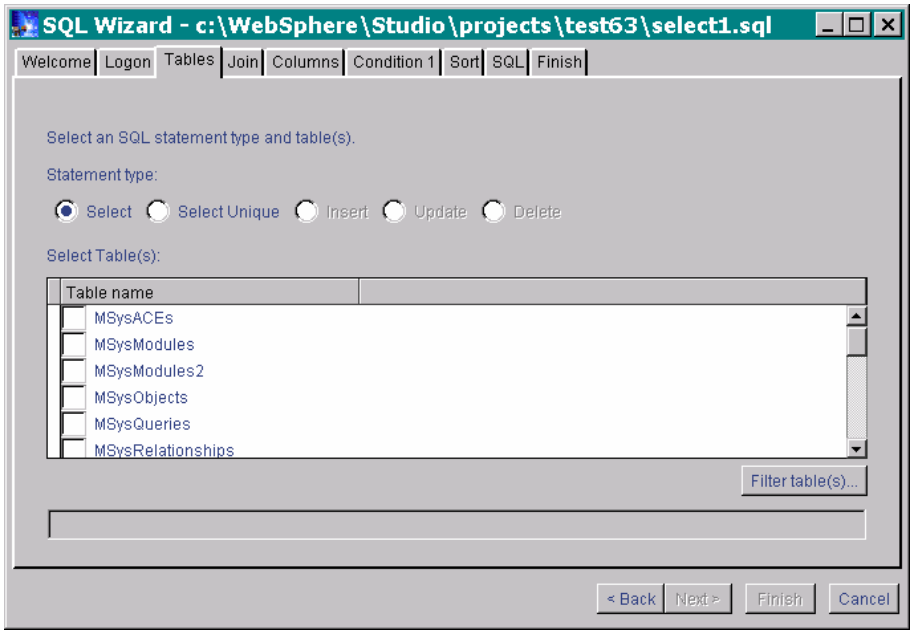
➢ Name your query! (For example select1)



- Press Next and fill out the form according to the next figure:

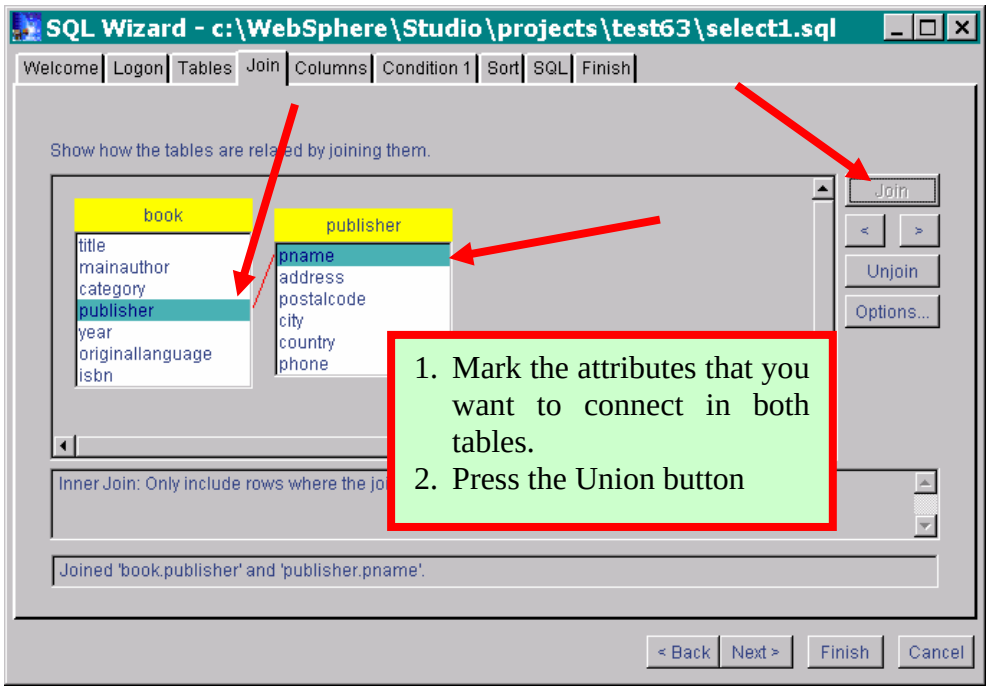


- Press the Connect button to get to the next step. If the connection is successful then a list of all the available tables should come up:



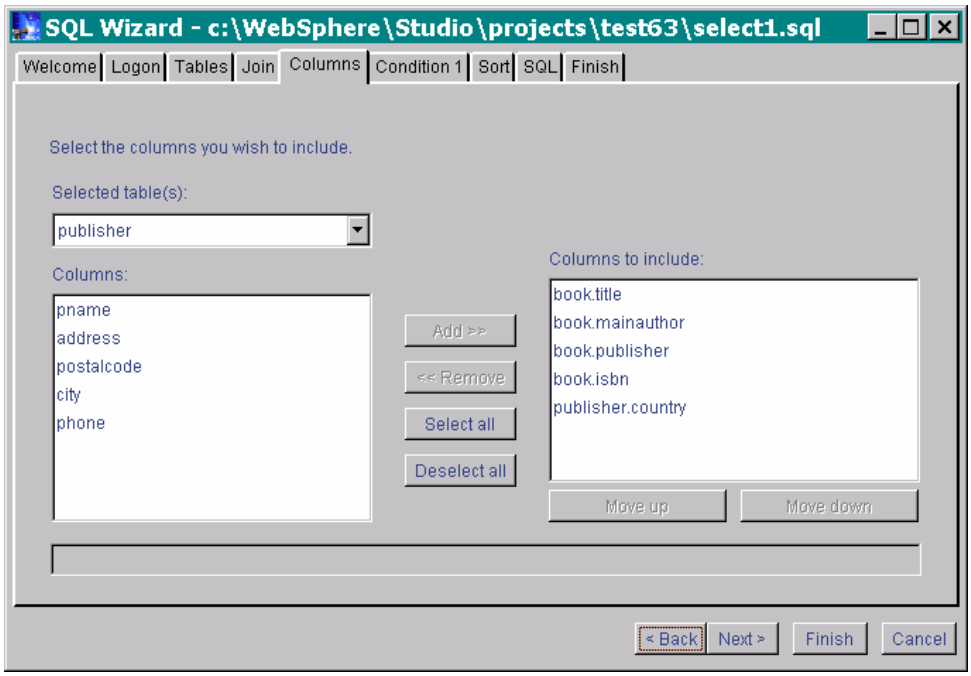
- Select the tables that should be involved. (in this case book and publisher)

- Press Next to go to the “Join” tab. Here you can define rules for joining the two tables (basically for the foreign keys).



- Press Next to go to the “Columns” tab.

- Include the columns that are requested. (e.g. book.title, book.mainauthor, book.publisher, publisher.country, book.isbn)

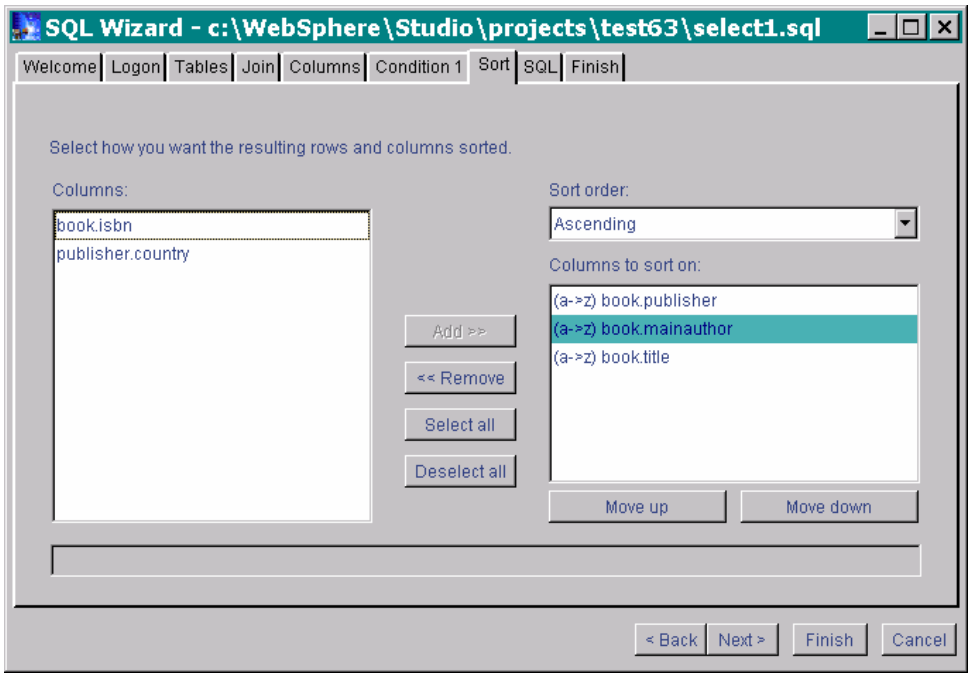


- Press Next to go to the “Condition” tab.

- You can leave the conditions empty.

- Press Next to go to the “Sort” tab

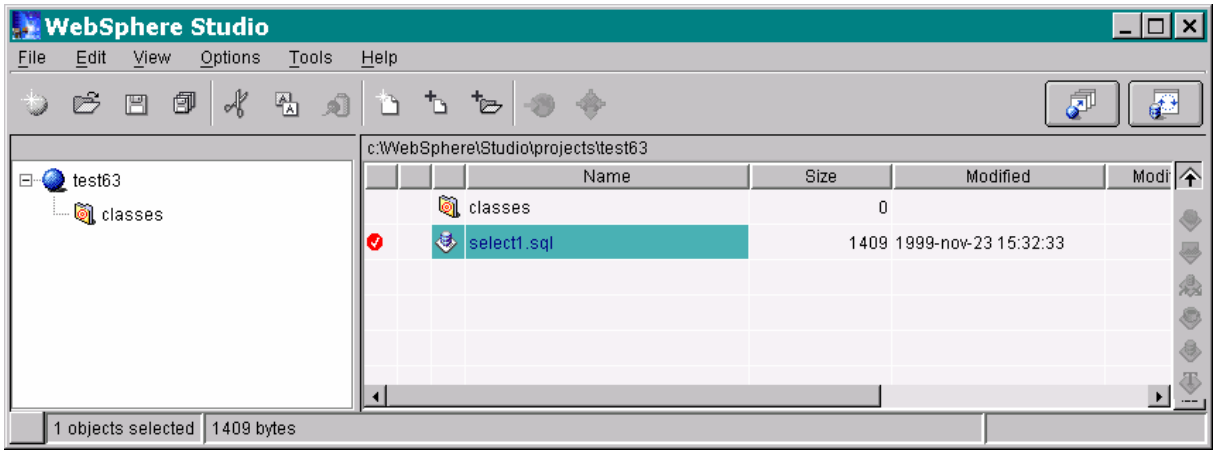
- Here you can define which column(s) the result should be ordered by.



- In the next tab you can see the SQL statement that you have graphically built.

- You can now hit the “Finish” button.

A new file has been added to the project containing all the information about the SQL statement:



3.2 Using the Studio Wizard

Based on the SQL created by the SQL Wizard, the Studio Wizard can create a servlet, a JavaBean and JSP. The following figure illustrates how these three components work together to provide the web client with the result.

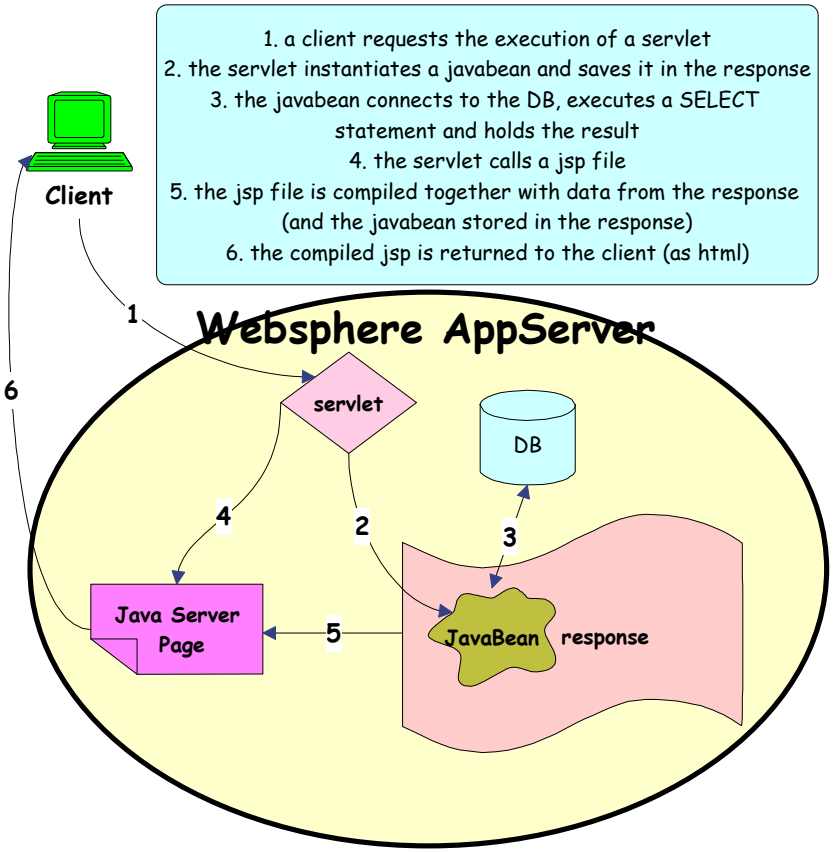
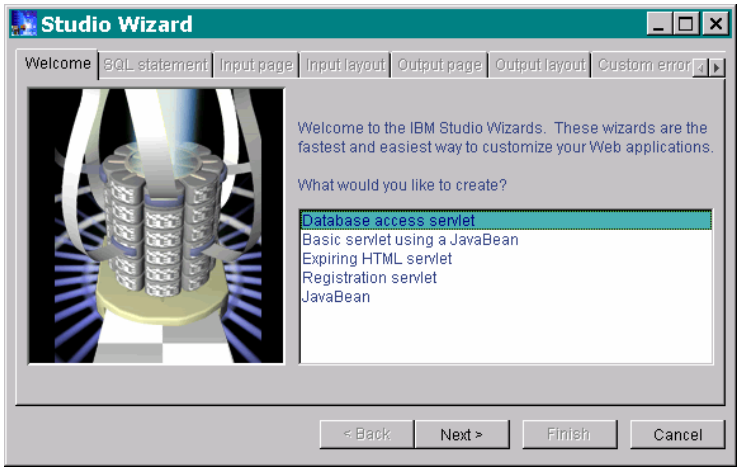
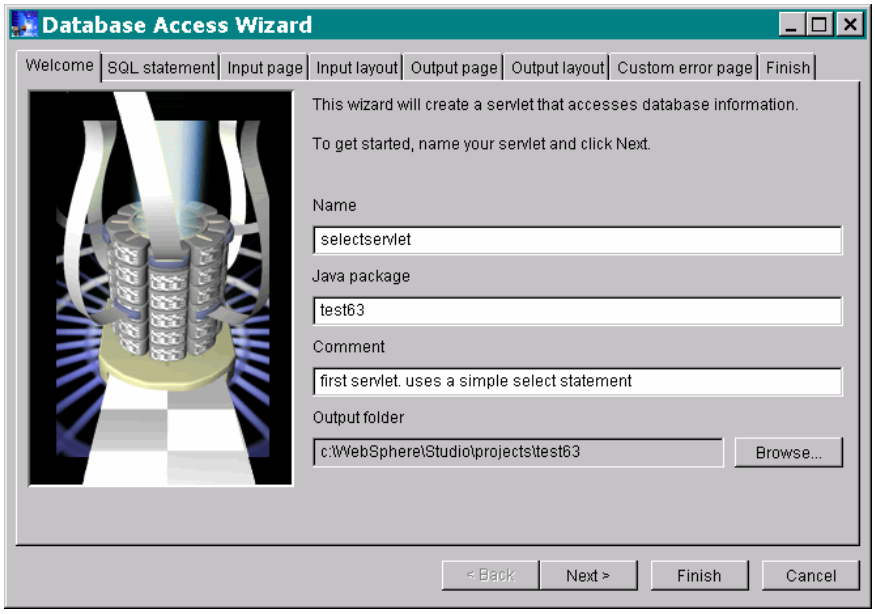


Figure 2 Inside WebSphere Application Server – Execution of a servlet

- Start the Studio Wizard. (Tools > Studio Wizard)
- Choose to create a Database access servlet and press Next.



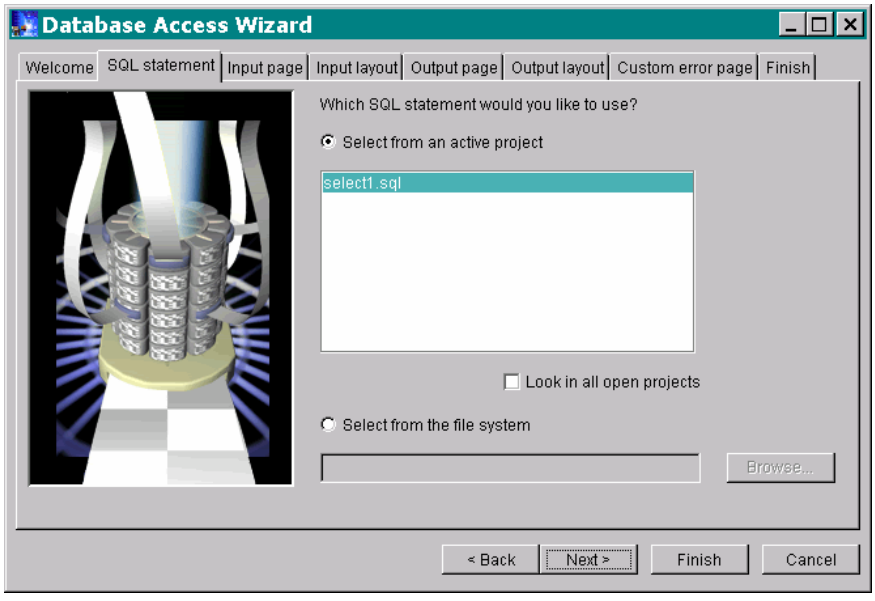
- Give the servlet a name. (e.g. selectservlet)



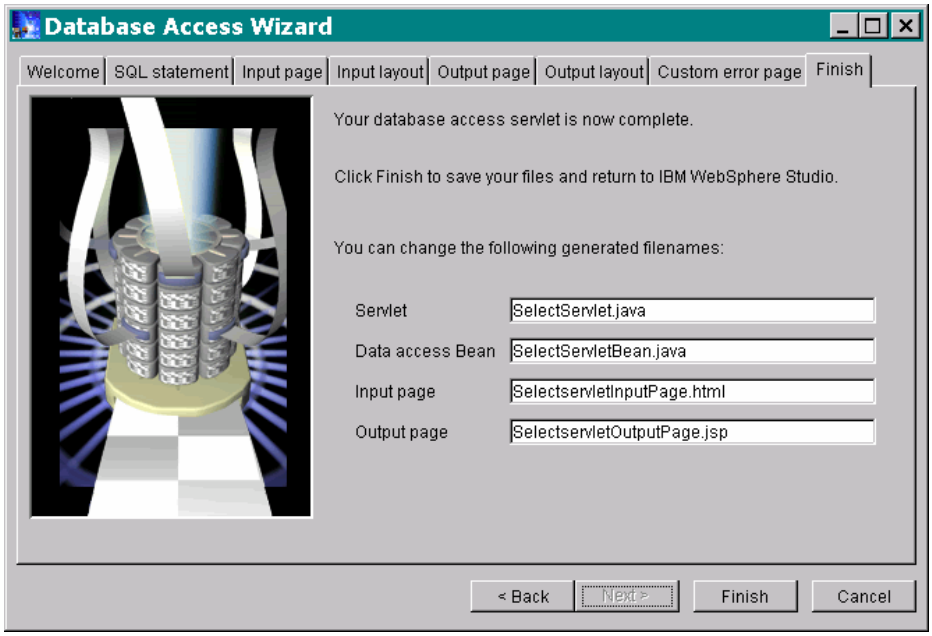
ⓘ The Java package property is normally the same as the project name. In projects with many servlets it can be good to have a better structure of java packages.

➢ Press Next.

➢ Choose the SQL statement that was created by the SQL Wizard before.



➢ Leave the next four tabs as they are and go to the “Finish” tab.

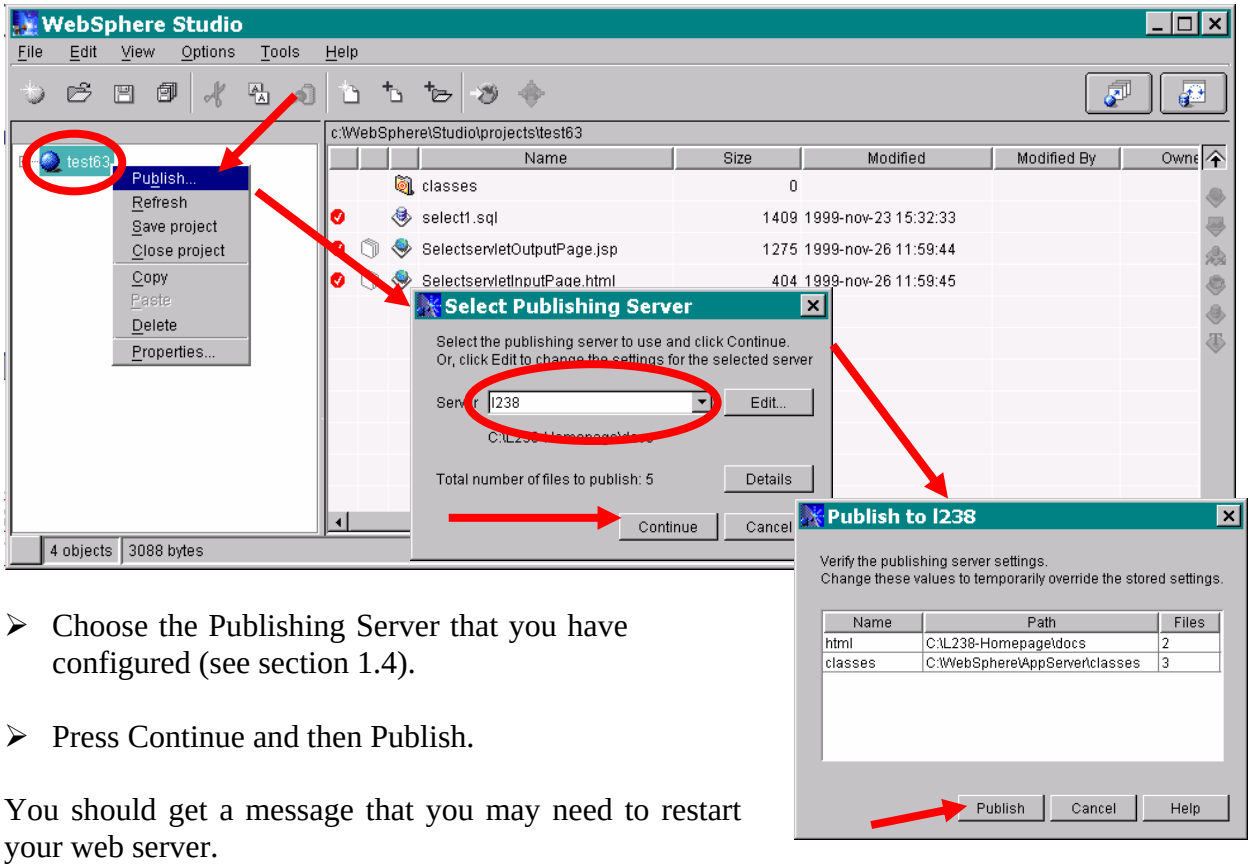


Here you have the possibility to choose the filenames for all the components that are about to be created. The Input page is just an html page with a button that calls the servlet. The Data access Bean is a javabean that makes the connection to the database and executes the SQL statement. The Output page is a JSP file that gets the information from the javabean and formats it into plain html. The servlet is the connection between all the other components. It receives and handles a request, instantiates the javabean and initiates a response (see Figure 2).

- Press Finish!

All the files have now been created and the classes have been compiled. It is now possible to edit these files, for example to make design changes. For now we will not make any changes. The only thing remaining is to publish the files.

- Mark the project, right click on it and choose Publish…



- Choose the Publishing Server that you have configured (see section 1.4).

- Press Continue and then Publish.

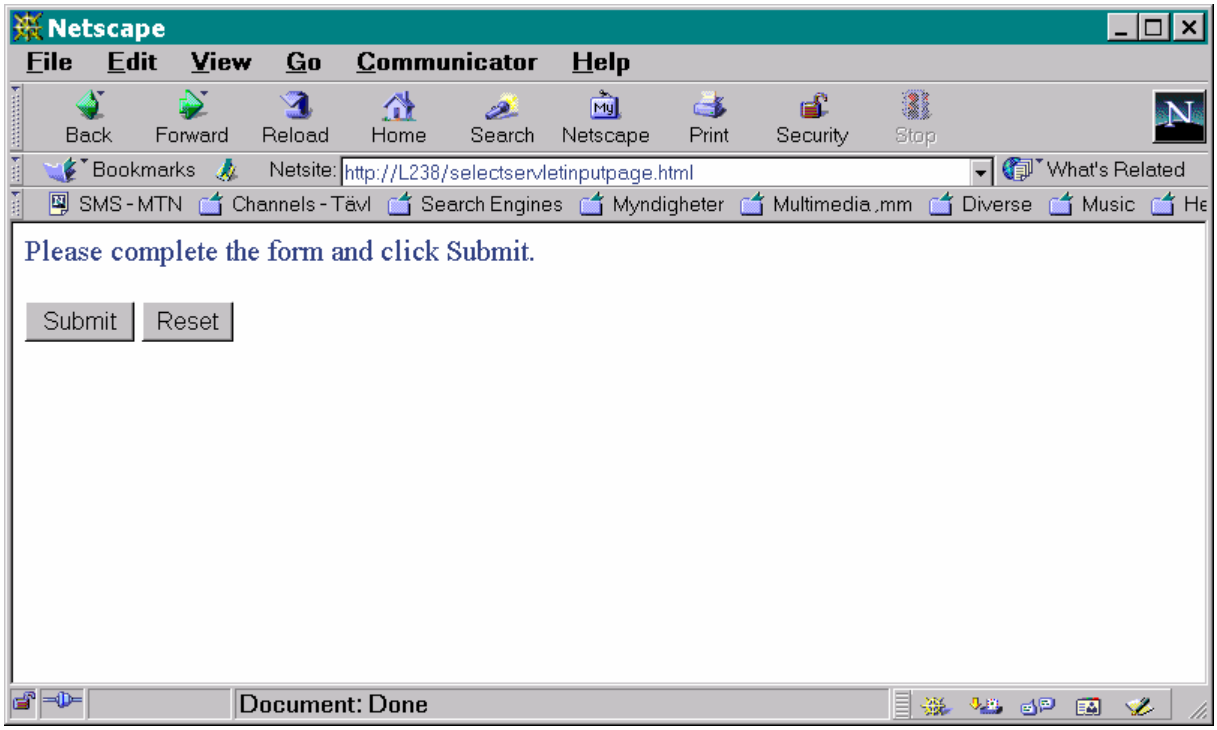
You should get a message that you may need to restart your web server.

- Press OK.

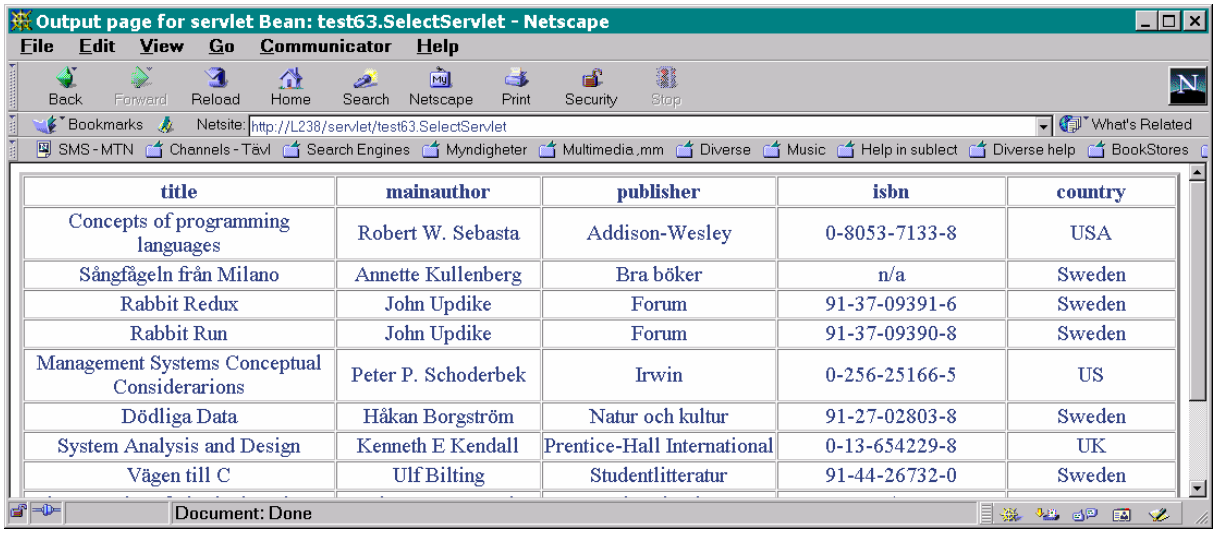
It should now be possible to run the servlet.

- Start a web browser and go to the following URL:
  http://%nameofmachine%/%InputPagePath%%InputPageFilename%
  Where %nameofmachine% is your machine's name, e.g. L269,
  %InputPagePath% is the path to your input page, starting from the web server's root. In this case nothing,
  %InputPageFilename% is the name of the input file generated by WebSphere Studio. In this case SelectservletInputPage.html.

For example http://l238/selectservletInputPage.html:



- Press Submit.



title	mainauthor	publisher	isbn	country
Concepts of programming languages	Robert W. Sebasta	Addison-Wesley	0-8053-7133-8	USA
Sångfågeln från Milano	Annette Kullenberg	Bra böker	n/a	Sweden
Rabbit Redux	John Updike	Forum	91-37-09391-6	Sweden
Rabbit Run	John Updike	Forum	91-37-09390-8	Sweden
Management Systems Conceptual Considerarions	Peter P. Schoderbek	Irwin	0-256-25166-5	US
Dödliga Data	Håkan Borgström	Natur och kultur	91-27-02803-8	Sweden
System Analysis and Design	Kenneth E Kendall	Prentice-Hall International	0-13-654229-8	UK
Vägen till C	Ulf Bilting	Studentlitteratur	91-44-26732-0	Sweden

This is the default layout of the output page. It can easily be changed by editing the selectservletoutputpage.jsp file. To edit it you can double click on the filename in WebSphere Studio.

A jsp file is very similar to an html file. In addition to the usual html tags there are tags that help to access the javabean. Such tags appear in selectservletoutputpage.jsp:

<BEAN> <REPEAT> <INSERT> <% %>

When editing jsp files be careful with those tags. If they are not correct, the server will return an error. After making changes you have to republish the altered file(s).

3.3 Updating data

Updating data can be a little more complicated. WebSphere Studio cannot create all the components needed to insert, update and delete data.

In this section we will create three servlets for updating the information of an author:
1. One that shows a list of all the authors in the database so that the user can choose one.
2. A second servlet that shows the information stored in the database for the selected author, so that the user can change it.
3. A final servlet that commits the changes to the database.

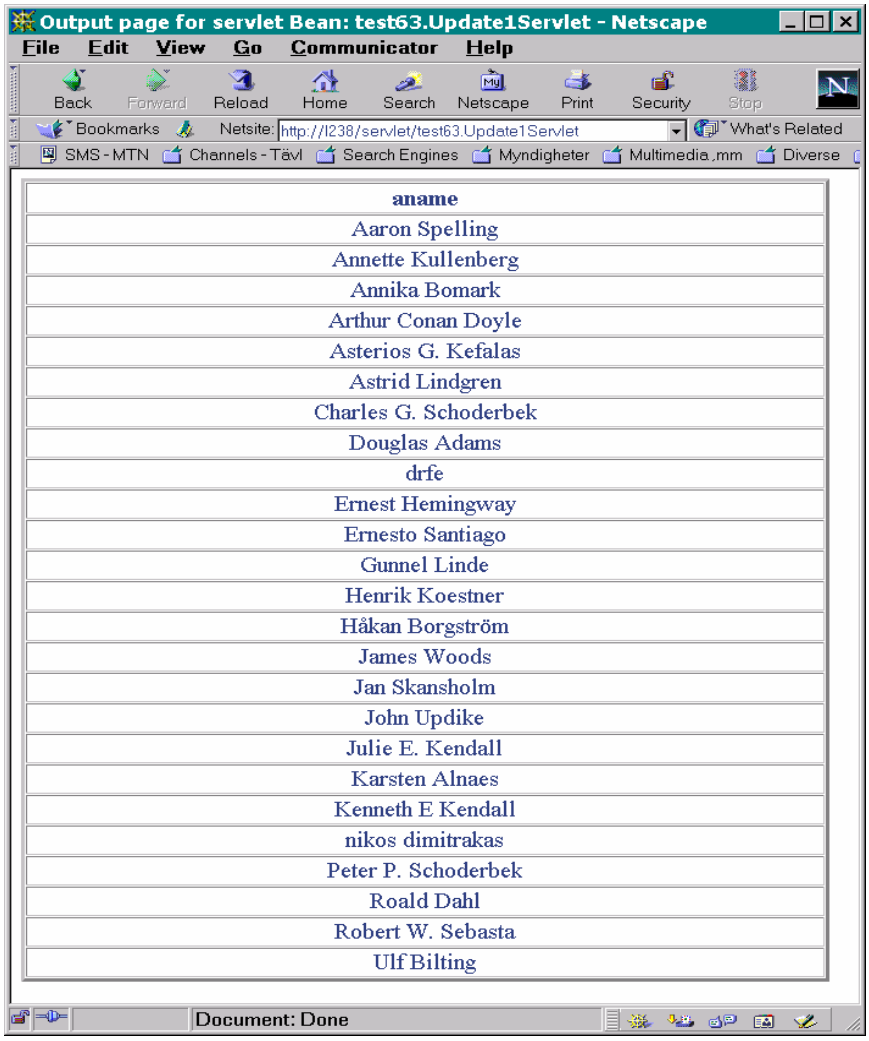
The first two can more or less be generated by WebSphere Studio. The third servlet requires a little coding.

3.3.1 Showing all authors

This part is similar to sections 3.1 & 3.2.

- Create in a similar way a servlet that shows the names of all the authors.
  - Create an sql with the SQL Wizard.
  - Create a database access servlet with the Studio Wizard.
  - Use smart filenames, think that there is going to be around 15 files.
  - Don't forget to publish your project.

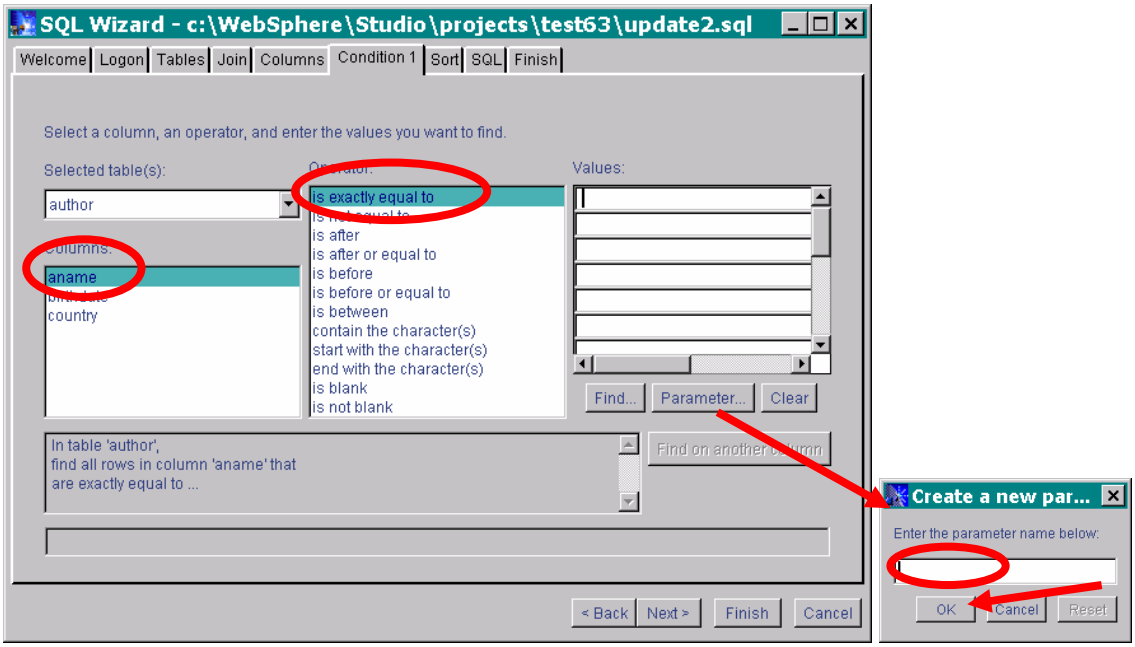
When you are done you should have a servlet that produces a result similar to the following:



3.3.2 Showing selected author's information

The second step is to show the information of the selected author. We can create an SQL with the SQL Wizard that uses a parameter:

- Create a new sql with the SQL Wizard in the same way as before until the "Condition" tab.
- At the "Condition" tab, select the column that is going to be in the condition (the aname).
- Select the condition operator (equal to).
- Press the parameter button and give the parameter a name.
- Press OK.



- Continue with the SQL Wizard as before.

- Create a database access servlet as before.

Now it is time to write some code!

WebSphere Studio has generated a javabean that has code for activating the parameter in the sql. That code works with most databases but not with MS Access. Therefore that code has to be replaced with some more primitive embedded SQL.

- Open the javabean in an editor (by double-clicking on it)

Here there is the following code:

```
…
/**
* Instance variable for the SQLString property
*/
protected java.lang.String SQLString = "SELECT \"author\".\"aname\", \"author\".\"birthdate\", \"author\".\"country\" FROM \"author\" WHERE ( ( \"author\".\"?\" = aname ) )";
…
// Create placeholders for the parameters
metaData.addParameter("aname", java.lang.String.class , 12);
…
initialize();

// Initialize the parameters for the query
selectStatement.setParameterFromString("aname", getAname());
```

The SQLString variable contains the SQL statement to be executed. In that statement there is a question mark. That question mark is supposed to be replaced with the value of the aname variable (the parameter). This technique is not accepted by MS Access.

Instead we can build the SQLString first and then execute it:

- Scroll down to the method called initialize().
- Remove the following rows of code:
  ```
  // Create placeholders for the parameters
  metaData.addParameter("aname", java.lang.String.class , 12);
  ```
- Scroll down to the method called execute().
- Remove the following rows of code:
  ```
  // Initialize the parameters for the query
  selectStatement.setParameterFromString("aname", getAname());
  ```

To replace the rows that we just removed we can add the following code to the method execute(), exactly before this row

```
initialize();
```

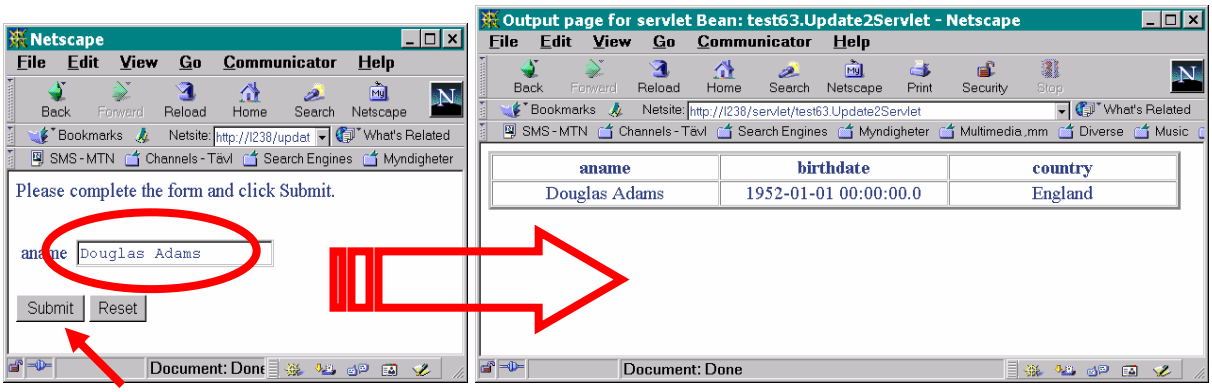
Add the following:

```
SQLString = "SELECT author.aname, author.birthdate, author.country FROM author WHERE author.aname = '" + aname + "'";
```

In this way we have built the SQLString before we initialize the database connection and MS Access cannot complain.

- Save the file and compile it in WebSphere Studio (Mark the file and choose File > Compile file).
- Publish the project (If you have already tried to run the servlet before the changes, then you may need to restart the web server (see section 1.1)).

After all that is done, you should have a servlet with the following input and output pages:



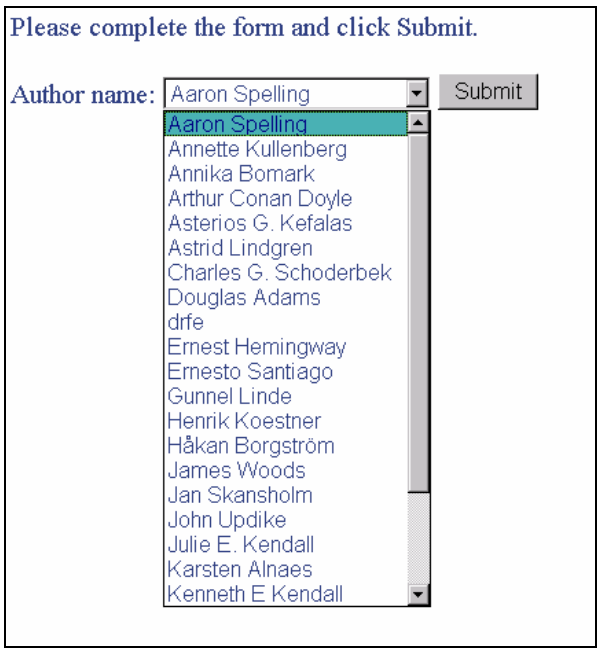
Now we have to connect these to servlets. The user should use the output page of the first servlet as an input page to the second. Similarly the output page of the second servlet should

be used as the input page to the third servlet. To do this we need to "merge" the output page of the first servlet with the input page of the second servlet.

The input page of the second servlet is quite simple. It is an html file with a form. The form's action calls the second servlet. The form has an input field for the parameter and a submit button.

The output page of the first servlet has a loop (<REPEAT>) for going through the result of the SQL.

By combining these two we can get an output page that looks like this:



3.3.3 Updating data

The third servlet is a little different. Instead of a SELECT statement it should contain an UPDATE statement. Even though WebSphere Studio cannot create all the code needed, it is a good idea to use the Wizards to create as much as possible. To do that we can create an SQL statement (SELECT statement) that takes as parameters the same parameters as the UPDATE statement would take (oldaname, newaname, newbirthdate, newcountry - ID for the old record and all the new information).

- Create an SQL with the SQL Wizard.

- Set multiple condition by clicking on the "Find on another column" button
  - ⓘ Since the SQL Wizard does not allow you to create conditions on date columns, set the condition on another column. We would anyway change the code… It doesn't matter if the sql doesn't make any sense right now. We will just use this sql statement to create a servlet and a javabean with parameter variables.

- Create now a database access servlet with the Studio Wizard based on the new SQL.

We need to make the following changes to the generated files:

1. Change the javabean. Instead of a SELECT statement we should have an UPDATE statement. There is a lot of code that can be removed: all the methods that have to do with the result set (an UPDATE statement does not produce a result set).
2. Once there is no result set, there is nothing to show on the output page. We can modify that page to some static html.
3. Merge the input page with the output page of the previous servlet.

- Start by opening the javabean for editing.

All the following variables and methods have to do with the result set and are therefore of no use. Remove them:

```
private static final int author_aname_COLUMN
private static final int author_birthdate_COLUMN
private static final int author_country_COLUMN
protected java.lang.String SQLString
public java.lang.String getSQLString()
public java.lang.Object getAuthor_aname(int row)
public java.lang.Object getAuthor_birthdate(int row)
public java.lang.Object getAuthor_country(int row)
private java.lang.Object valueAtColumnRow(int column, int row)
public void closeResultSet()
protected com.ibm.servlet.connmgr.IBMJdbcConn getPooledConnection(java.lang.String driver,
    java.lang.String URL, java.lang.String userID, java.lang.String password)
protected com.ibm.servlet.connmgr.IBMJdbcConn getPooledConnection(java.lang.String poolname,
    java.lang.String driver, java.lang.String URL, java.lang.String userID, java.lang.String password)
```

There are also a few variables that take care of the database connection:

```
protected com.ibm.servlet.connmgr.IBMConnMgr connectionManager;
protected com.ibm.servlet.connmgr.IBMJdbcConn ibmJdbcConn;
protected com.ibm.db.SelectStatement selectStatement;
protected com.ibm.db.SelectResult result;
```

We will replace them with variables from the java.sql package:

```
static protected Connection con;
static protected Statement stmt;
```

Add the following line at the top of the file:

```
import java.sql.*;
```

Also remove the following method:

```
public void initialize()
```

This method established the connection based on the ibm classes. We will now add a method for establishing a connection based on the java.sql classes:

```
// establish DB connection
  public void dbConnect()
  {
     try
     {
       // register the driver with DriverManager
       Class.forName(getDriver());
       con = DriverManager.getConnection(getURL(), getUserID(), password);
       con.setAutoCommit(true);
     }
     catch (Exception e)
     {
        e.printStackTrace();
     }
  }
```

This method simply creates a database connection.

The main method of this javabean is the execute() method. It is this method that the servlet calls to execute the SQL statement. We can change this method to first call the dbConnect() method and then execute the UPDATE statement.

- Remove all the code of the method:

```
initialize();
// Initialize the parameters for the query
selectStatement.setParameterFromString("oldaname",);
selectStatement.setParameterFromString("newaname",);
selectStatement.setParameterFromString("newcountry",);
selectStatement.setParameterFromString("newbirthdate",);
// Execute the SQL statement
selectStatement.execute();
result = selectStatement.getResult();
// release the connection for use by another SQL statement
ibmJdbcConn.releaseIBMConnection();
```

- And replace it with this:

```
dbConnect();
String query;
query = "UPDATE author SET aname = '" + getNewaname()+ "', country = '" +
    getNewcountry()+ "', birthdate = CDATE(LEFT('" + getNewbirthdate()+ "', 10)) WHERE
    aname = '" + getOldaname()+ "'";
stmt = con.createStatement();
stmt.executeUpdate (query);
stmt.close();
con.commit();
```

  - ⓘ CDATE and LEFT are functions that MS Access has. CDATE transforms a string to a date. LEFT returns a sub-string. They can be omitted if there is some other code that controls the length of the date parameter. Dates in MS Access are not exactly compatible with the java.sql.date format.

- Also change the declaration of the method to throw the appropriate exception:

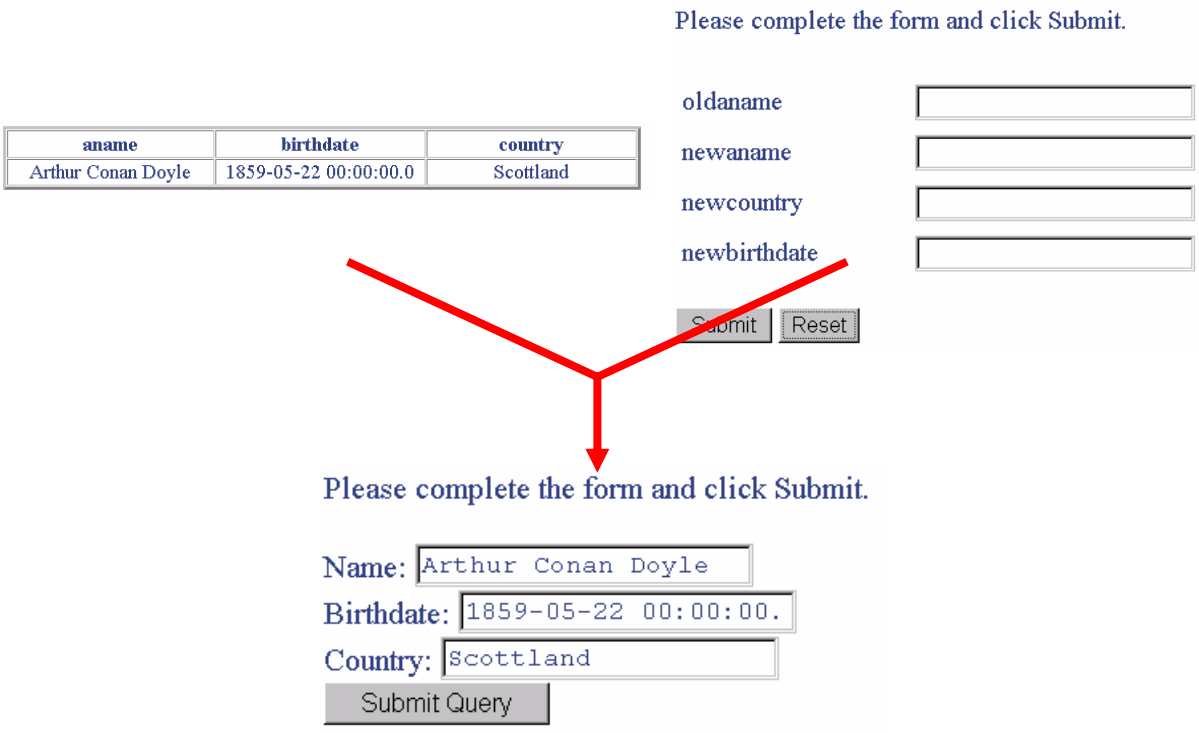
```
public void execute() throws java.sql.SQLException
```

When all this is done you should be able to compile the javabean without errors.
If you would now try to execute the servlet, it would return the output page which would complain. That is because the output page is still configured according to the old javabean.

- Edit the output page and remove all the JSP specific tags that were supposed to show the result set of the SELECT statement. Here is an example of how the output page could be:

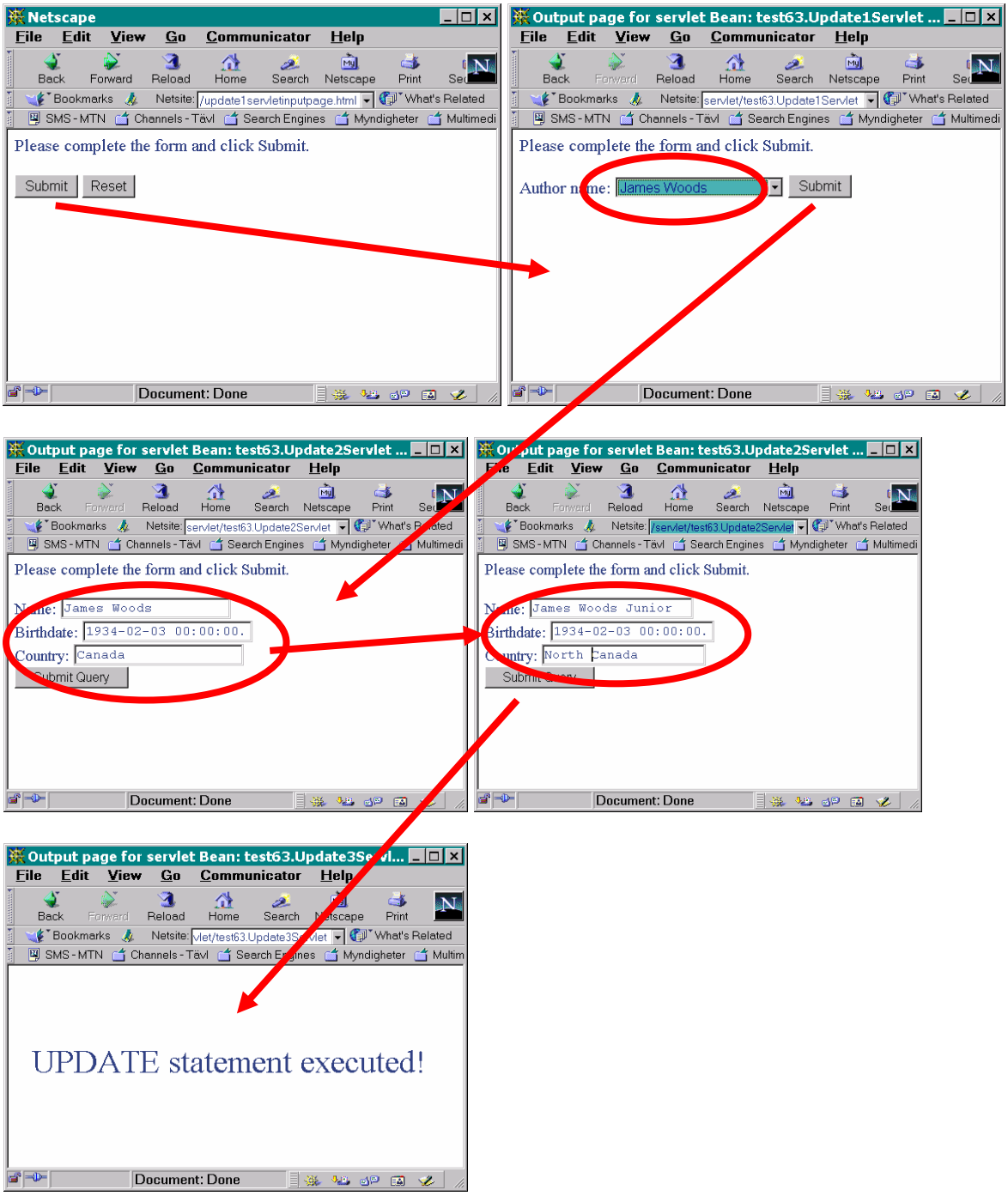
```
<HTML>
<HEAD>
<TITLE>Output page for servlet Bean: test63.Update3Servlet</TITLE>
</HEAD>
<BODY>
<FONT SIZE="+3"><BR><BR>
<CENTER>UPDATE statement executed!</CENTER>
</FONT>
</BODY>
</HTML>
```

The last thing is to connect the second servlet to the third.
Here is an example of how to merge the output page of the second servlet with the input page to the third servlet:



ⓘ The field “oldaname” is also present in the new form but as a hidden field!

After publishing the project you should have a chain of these three servlets:



4 ASPs

In this section we will try to create a set of ASPs to provide the same functionality as before. ASPs work in a way similar to servlets. The following figure illustrates how an ASP is executed:

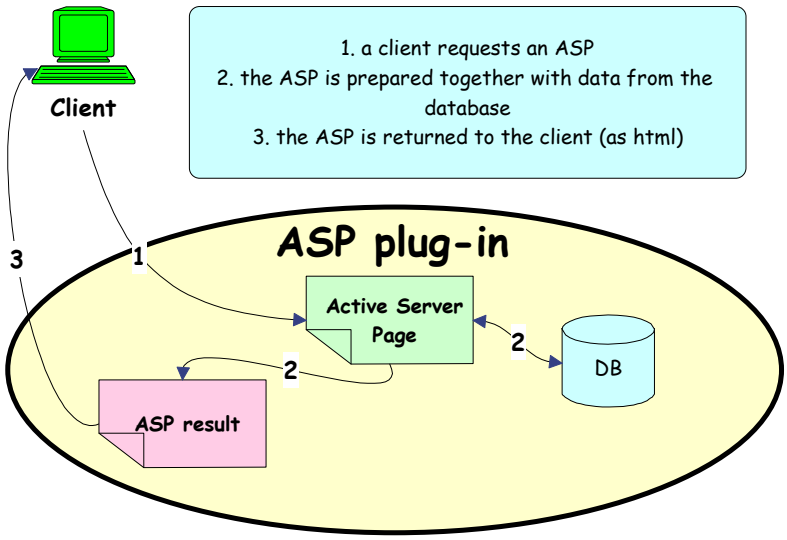


Figure 3 Inside ASP plug-in - ASP execution

ASPs are very much like html files. In addition to the usual html content, an ASP includes script commands within the <% and %> delimiters. The default script language is VBScript. Everything that appears between those delimiters is executed on the server (in our case by the ASP plug-in). An ASP can look like this:

```
<HTML>
<% name = “nikos” %>
<BODY>
<B>Hej <%= name%></B>
</BODY>
</HTML>
```

A variable “name” with value “nikos”

It can be embedded to the html content. Use the <%= and %> to return the value of a variable.

After the script part of the ASP has been executed the ASP should look like this:

```
<HTML>
<BODY>
<B>Hej nikos</B>
</BODY>
</HTML>
```

4.1 Accessing a database

To access a database with ASPs is very simple. The following two lines of code establish a connection to a database:

```
<% Set db = Server.CreateObject(“ADODB.Connection”)
db.open “%ODBC-DSN%”, “%user-name%”, “%password%” %>
```

The first line creates a connection object and assigns it to the variable name “db”. The second line instantiates the connection to the specific database, through an ODBC DSN. The user-name and password to the database can also be included.

At the end of the ASP the database connection can be closed:

```
<% db.close
Set db = Nothing %>
```

To execute an SQL statement is also simple:

```
<%
SQLString = "SQL statement"
Set result = Server.CreateObject("ADODB.Recordset")
result.Open SQLString, db, 3, 3
%>
```

First create a string with the SQL statement.
Then create a variable to receive the result of the SQL statement.
Finally run the SQL statement on a specific database connection. The third and fourth parameters of the "Open" function specify the following:
adOpenKeyset & *adLockBatchOptimistic*
In this compendium these parameters will always be set to 3.

Another method to execute SQL statements is this:

```
<% Set result = db.Execute("SQL statement") %>
```

And of course if the SQL statement's result is not important:

```
<% db.Execute("SQL statement") %>
```

To navigate through a result of an SQL statement there is the following functions:

Move to the first record:
`result.movefirst`

Move to the last record:
`result.movelast`

Move to the next record:
`result.movenext`

Move to the previous record:
`result.moveprevious`

To delete the current record in a result-set:
`result.delete`

To check the beginning and the end of the result-set:
`result.bof`, `result.eof` return true/false

To loop through a result-set, writing a field from each record to the web page:

```
do until result.eof
=result("FieldName")          (or result.Fields("FieldName"))
result.movenext
loop
```

ⓘ To retrieve the value at a field of the current row of the result-set you can either use the field's name or its index position, starting with 0.

4.2 First ASP

Let's now take the first question and try to make an ASP for it:

List all the books (title, mainauthor, isbn) and their publisher (name and country) ordered by publisher, author and book title

To write an ASP file you just need an editor. It is recommended to use ScriptBuilder, because it supports ASP files.

- Create a new file with asp as extension. Save this file at a location were the web server can find it, for example in a directory called ASP under the web server root:
  c:\Netscape\SuiteSpot\docs\ASP\file.asp

- Choose what design you want for your result and write all the static html code for it. If, for example, you want to have the result of the SELECT statement in a table, you can write the html code for the table and the titles…

Here is a possible design:

Book			Publisher	
Title	Main Author	ISBN	Name	Country
Data	Data	Data	Data	Data
.	.	.	.	.
.	.	.	.	.

- Add now some code at the beginning of the file to connect to the database:

```
<% Set db = Server.CreateObject("ADODB.Connection")
db.open "lab63" %>
```

- Now And some code for the SELECT statement:

```
<%
statement = "SELECT book.title, book.mainauthor, book.isbn, publisher.pname, publisher.country
FROM book, publisher WHERE book.publisher = publisher.pname ORDER BY publisher.pname,
book.mainauthor, book.title"
Set resultset = Server.CreateObject("ADODB.Recordset")
resultset.Open statement, db, 3, 3
%>
```

Now we need to go through the result-set and populate the table. To do that we need a loop.

- Find the part of the code that you want to repeat for every row and include it in the loop. For example:

```
<%do until resultset.eof%>
<TR>
<TD>
<%=resultset("title") %>
</TD>
<TD>
<%=resultset.Fields(1) %>
</TD>
.
.
.
<%resultset.movenext
loop%>
```

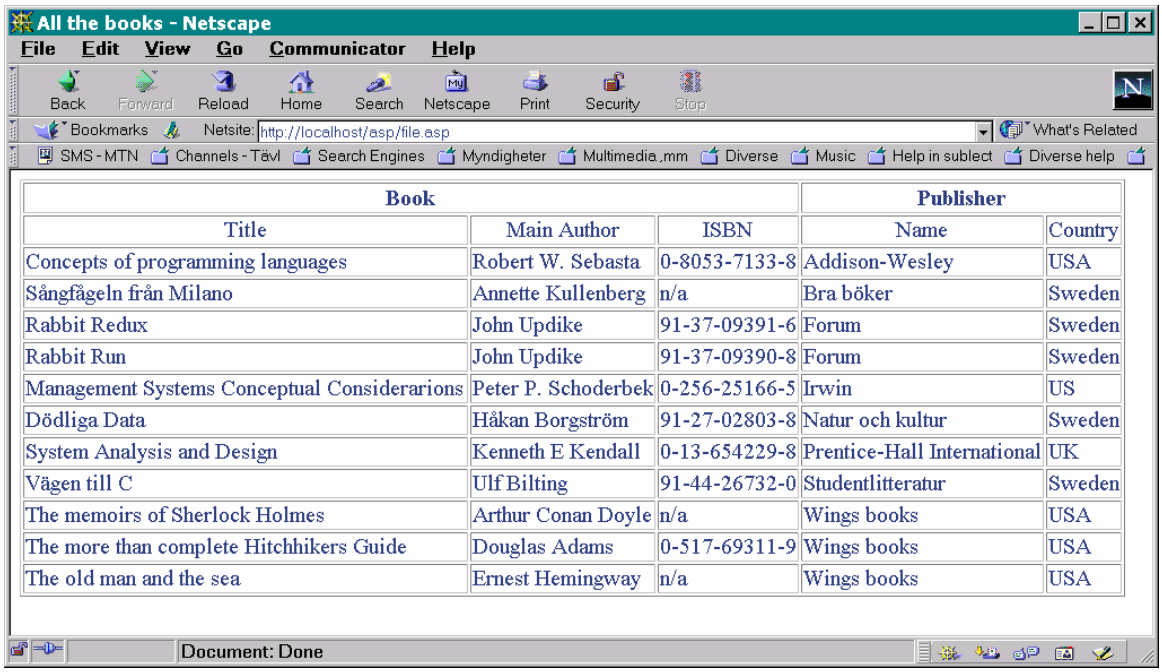
Everything that is included in the loop is going to be the ASP result once for every row in the result-set.

- Make sure that there is matching <% and %> delimiters around the VBScript code blocks. No html content should be within the <% and %> delimiters.

- At the end of the file add some code to close the database connection:

```
<% db.close
Set db = Nothing %>
```

When you call the ASP from a browser you should get something like this:



Book			Publisher	
Title	Main Author	ISBN	Name	Country
Concepts of programming languages	Robert W. Sebasta	0-8053-7133-8	Addison-Wesley	USA
Sångfågeln från Milano	Annette Kullenberg	n/a	Bra böker	Sweden
Rabbit Redux	John Updike	91-37-09391-6	Forum	Sweden
Rabbit Run	John Updike	91-37-09390-8	Forum	Sweden
Management Systems Conceptual Considerarions	Peter P. Schoderbek	0-256-25166-5	Irwin	US
Dödliga Data	Håkan Borgström	91-27-02803-8	Natur och kultur	Sweden
System Analysis and Design	Kenneth E Kendall	0-13-654229-8	Prentice-Hall International	UK
Vägen till C	Ulf Bilting	91-44-26732-0	Studentlitteratur	Sweden
The memoirs of Sherlock Holmes	Arthur Conan Doyle	n/a	Wings books	USA
The more than complete Hitchhikers Guide	Douglas Adams	0-517-69311-9	Wings books	USA
The old man and the sea	Ernest Hemingway	n/a	Wings books	USA

4.3 Updating data

The only difference when updating data is that the UPDATE statement is different every time. In this section we will try to build the same structure as in section 3.3.

First we need a page that shows all the authors and lets the user select one. That can be done in the exact same way as before (with a SELECT statement). The only thing that differs is the parameters that are sent from the one ASP to the next with a form. Example:

If you have a FORM that sends the name of the author as a parameter called “aname” then the receiving ASP can access this parameter in one of the following two ways:

If the FORM used the POST method then the parameters can be accessed like this:
```
Request.Form("aname")
```

If the FORM used the GET method then the parameters can be accessed like this:
```
Request.QueryString(“aname”)
```

Another thing that may come in handy is string concatenation. The concatenation operator in VBScript is the “&” character. Example:
```
<%
fname = “nikos”
lname = “dimitrakas”
fullname = fname & “ “ & lname
%>
```

- Complete this little application so that it behaves like the one in section 3.3.3.

5 Completed Lab requirements

The exercises in section 3 & 4 are compulsory. In addition to that, every group has to create one more set of servlets or ASPs for inserting or deleting data (INSERT or DELETE statement). Every group is free to choose their own statement and their own design.

Before the 22nd of December, you should do a short (oral) presentation of your work. Contact nikos to book time!

6 Internet Resources

VBScript & ASP

http://asp-help.com/

Servlets & JSP

http://www-4.ibm.com/software/webservers/appserv/doc/v20dcstd/doc/index.html

7 Epilogue

When all this is done, you should have a quite good understanding of how to use servlets and ASPs for making databases available on internet.

I hope you have enjoyed this compendium. Please come with feedback!

The Author

nikos dimitrakas