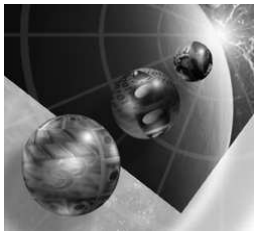


IBM WebSphere Development Tools for AS/400

Building an *e*-business application with RPG using WDT/400

Student Exercises

Course code 47LF

Claus Weiss
weiss@ca.ibm.com

IBM Toronto Laboratory

Version 5 Release 1 Modification 0

homepage: http://www.ibm.com/software/ad/wdt400
newsgroup: news://news.software.ibm.com/ibm.software.wdt400

Third Edition (September, 2001)

The information contained in this document has not been submitted to any formal IBM test and is distributed on an "as is" basis without any warranty either express or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will result elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Comments concerning this notebook and its usefulness for its intended purpose are welcome. You may send written comments to:

IBM Canada Software Solutions

3200 Warden Ave. Markham, Ontario, L6G 1C7

Attention: Claus Weiss, WDT/400 Introduction.

or e-mail to**: weiss@ca.ibm.com**

Overall Lab Guide

The objective of the WDT/400 lab is to have the students work with the WebSphere Studio/400 development environment to create, build and execute an *e*-business application. At the end of the lab, the student should know how to create a simple *e*-business application by creating a browser user interface, using the AS/400 Wizards in WebSphere Studio/400 to create the Servlets to connect to an RPG program that performs the business logic. The Lab also shows how to set up the publishing environment and how to configure the WebSphere Application Server (WAS) using the Publishing Wizard in WebSphere Studio/400.

All the exercises require that the previous labs are successfully completed.

.

Note: *The pictures in these labs show a similar application being built. Some of the names and icons may be different from the environment you are working with.*

Prerequisites

Although not required, familiarity with the RPG language is preferred.

Also, it is helpful if the student is familiar with basic MS Windows operations such as working with the desktop and basic mouse operations such as opening folders and performing drag-and-drop operations. The student should also be familiar with a browser and know how to navigate through the Internet.

Trademarks

IBM is a registered trademark of International Business Machines Corporation.

The following are trademarks of International Business Machines Corporation.

AS/400
Application System/400
VisualAge
DB2/400
ILE
Integrated Language Environment
IBM
OS/400
RPG/400
VisualAge
WebSphere

Trademarks of other companies as shown

'Lotus'	'Lotus Development Corporation'
'Microsoft'	'Microsoft Corporation'
'Windows'	'Microsoft Corporation'

Create a Simple HTML Page with WebSphere Studio

Exploring WebSphere Studio for AS/400

During this lab you will learn more about WebSphere Studio for AS/400. In doing so you will be performing the following:

1. Explore WDT/400 by:

   - Starting **WebSphere Studio for AS/400**
   - Getting familiar with the **page designer** tool
   - Using parts and **page designer components** to create a simple browser interface

2. Publish and test the files by:

   - Using the **Publishing Setup Wizard**
   - Publishing the files to the AS/400 server
   - Invoking the browser and showing the page

What You Should be Able to Do

As a result of this exercise you will create the first page of your *e*-business application. In doing so you will:

- Create a simple Java Server Page (JSP) to be used as the user interface in your application
- Use the Publishing Setup Wizard
- Use the Publishing support in WebSphere Studio/400
- Save your application
- Test the user interface

Starting WebSphere Studio

To start the WebSphere Studio/400 for the first time press the **Start** button on the task bar and choose ***Start→Programs→IBM WebSphere Development Tools for iSeries→ IBM WebSphere Studio for iSeries →IBM WebSphere Studio v3.5***.

When you open Studio for the first time, the program asks if you want to create a new project,

- If you get this **Start up window**
  - Select **Create a new project**.
  - Then proceed to the heading **New Project Dialog**
- If the **Studio** opens with an existing project, **Studio** has been used on this system before and opens with the last used project,
  - Select **New project** from the **File** menu option in the Studio project window.
  - Then you are presented with the new project dialog,

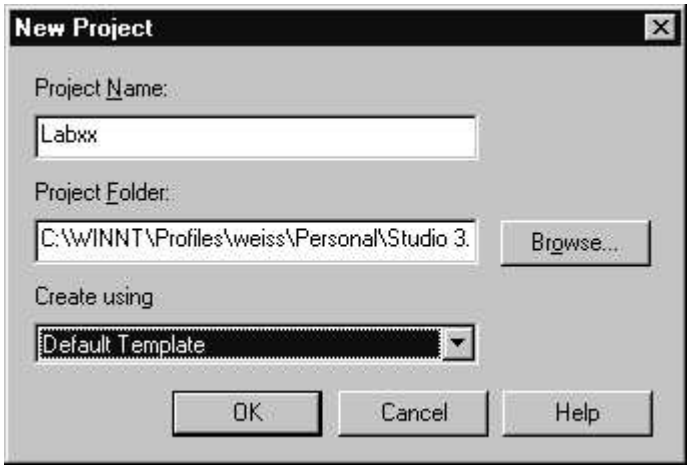
New Project Dialog

At this point the New Project dialog appears. This dialog prompts you to provide more information about your first web site.

- Specify project name **Labxx**, where xx denotes your Lab group number:

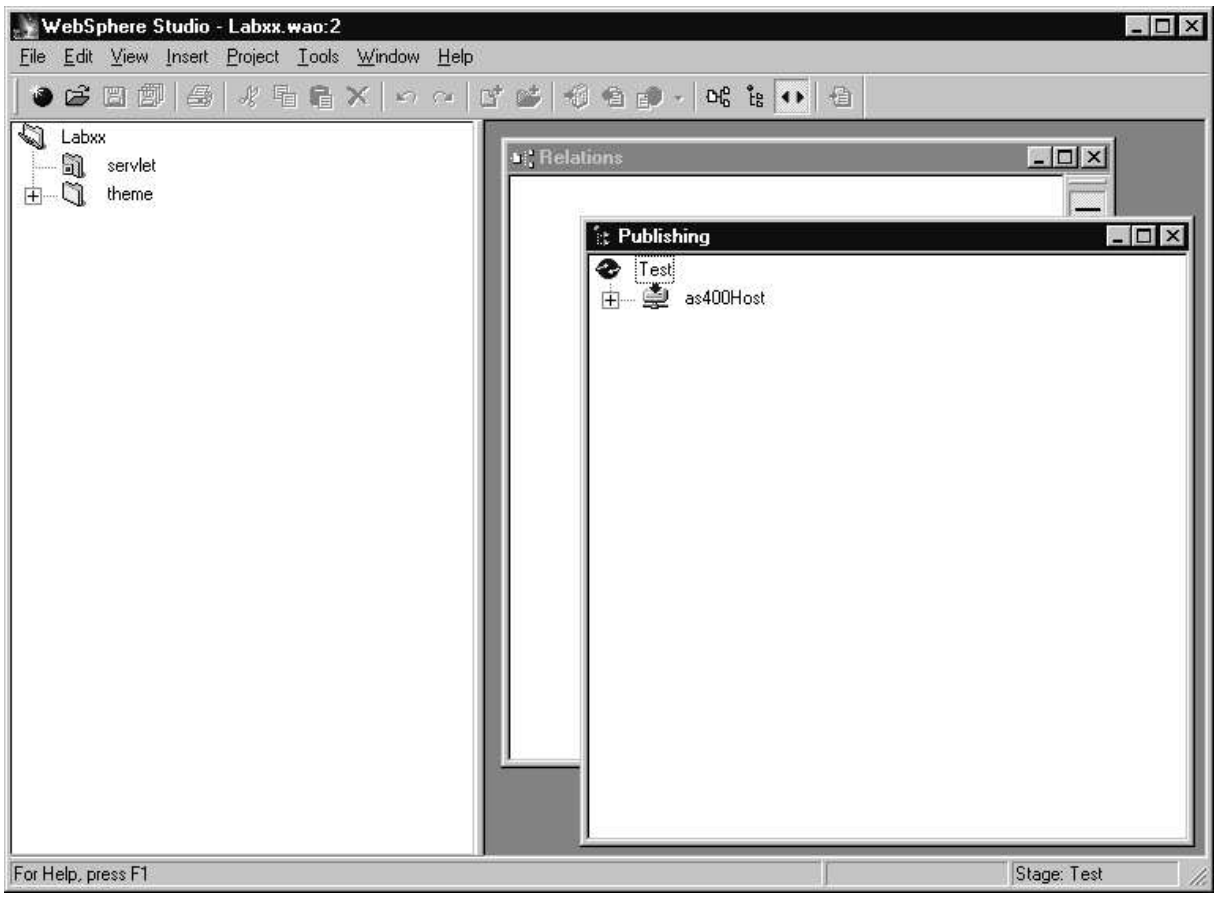
for example, if your are in group 40 use: Lab40

Note: **If you don't have a group number, please ask your instructor**



- Leave the other defaults as they are and press OK
  .

Now you are in the **Main window** of **Studio** as shown in the figure below. You are now ready to build your first web project.



Before you start with your project, just have a look at WebSphere Studio and check out some of its features.

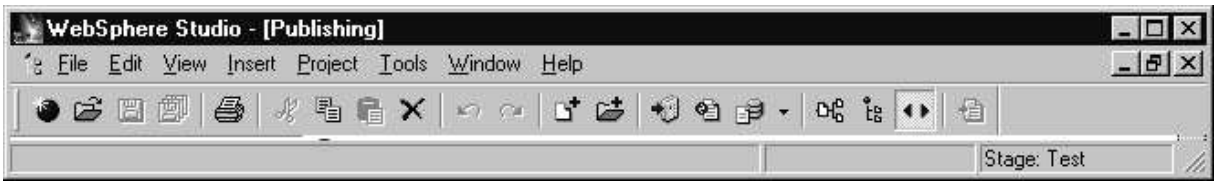
The WebSphere Studio main window

Studio's main window has a **File View** on the **left side**, which represents the directory structure of your source files. You can organize the file and folders any way you like. During the first steps of creating a web page you will work with the left side of the Studio main window, later you will use the right side as well.

Menu bar and tool bar

The menu bar shows the actions available to the user grouped by function, left mouse clicking on any of the menu options on the menu bar will show a drop down menu with detailed functions available for the selected group.

The **tool bar** is a menu of icons. It provides a fast path access to many of the menu items on the menu bar. **Move** the mouse pointer over **the icons** of the tool bar. A **short description** of the icons function is displayed in the form of **'flyover' help** next to the pointer**.**



The Studio toolbar is also context sensitive, as you use different tools inside of Studio the toolbars of these tools will be added.

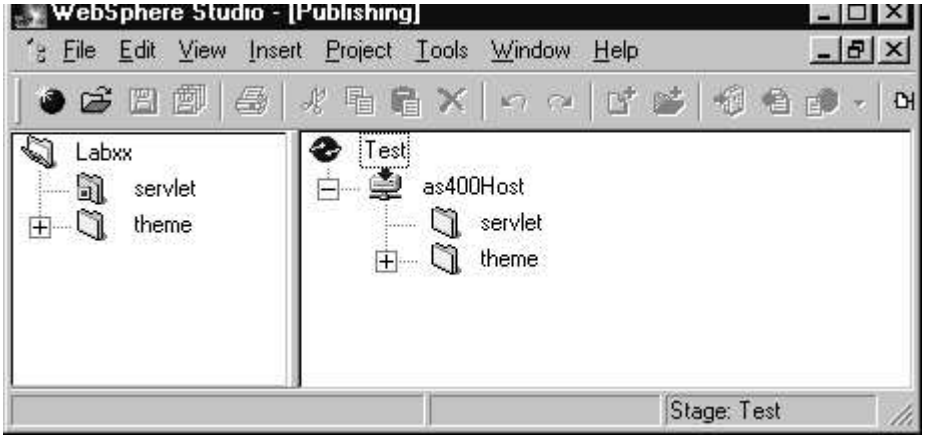
Now you will take a closer look at the right window of the Studio main window, this window allows you to have 2 different views into your Studio project the **Publishing** view and the **Relations view**. This window is also used to launch other tools that are integrated into Studio, for example the **Page Designer** tool that you will be using later.

The Publishing View

The **Publishing View** in the right hand window is used to view the web publishing environment set up for your project. You will have a chance to work with the publishing view later on in this Lab.

> **Note**: The window on your Lab system might show different files/folders than some of the pictures in this document. This is **ok,** don't try to fix it.

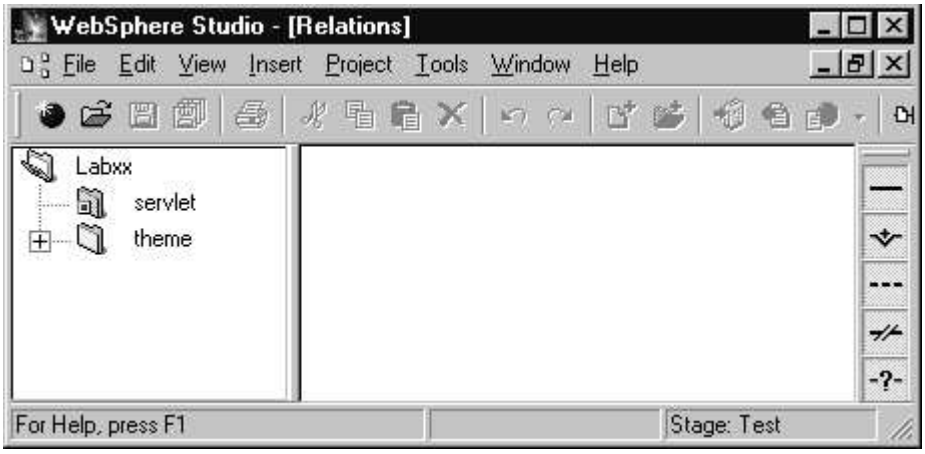
. Below you find an example of a publishing view:



The Relations View

- Click on the View menu option in the Studio main window menu and select Relations view from the pull down menu.

The right side window will change from the publishing view to a Relations view



This view shows the relations between the web pages in your project, since you don't have any web page this view is empty at the moment.

Now that you've had a short tour around the Studio main window, let's start with the project.

Studio Tools - Page Designer

We want you to design your first web page. The tool you'll be using is called **Page Designer** and it comes with Studio. It is like **SDA** for 5250 panels or a **GUI builder** for GUI windows, but as its name indicates it helps with designing web pages.

As in **SDA** or **CODE/400 designer**, instead of keying in source statements in an editor, **page designer** will create the source for you from your design choices. Here, the **source generated** is **html tags**, instead of the **DDS** statements that would be created for a display file by **SDA**.

To invoke **page designer** do the following:

Look at the Studio main window at the window at the left side, you see your **project** with two folders:

1. **servlet**
2. **theme**

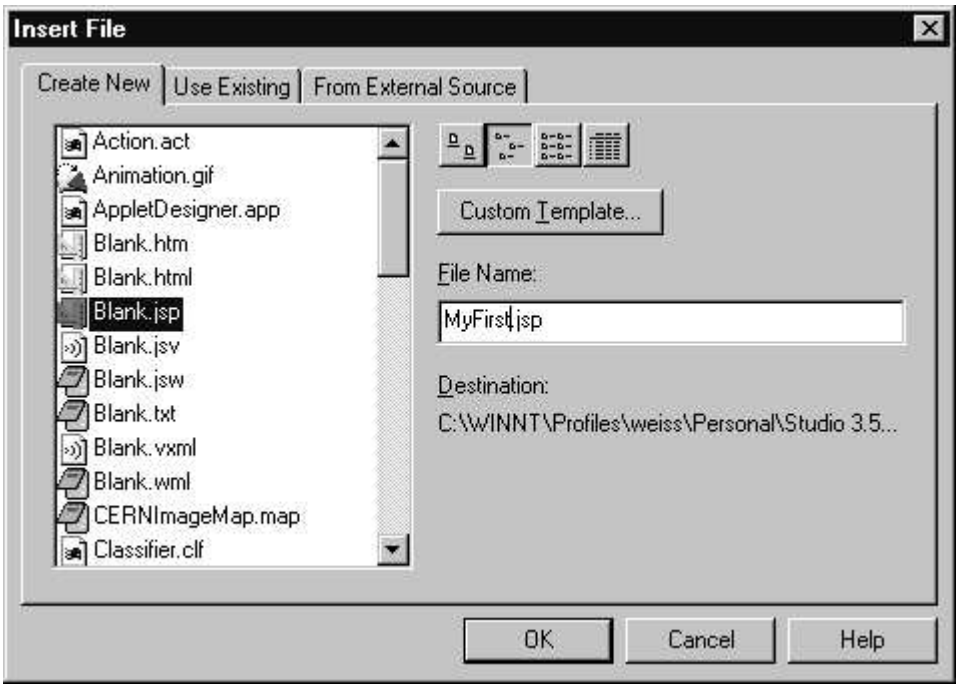
These folders have been created for you by Studio when you created this new project.

We want you to create your first page in the root project folder.

- **Right mouse click** on the root project folder **Labxx** directory
- Select **Insert --> File** from the **pop up** menu

The *Insert File* dialog appears
The dialog contains a list of templates for different file types.

- From this list, select the template, **Blank.jsp**
- Change the **File Name** entry field to **MyFirst.jsp**

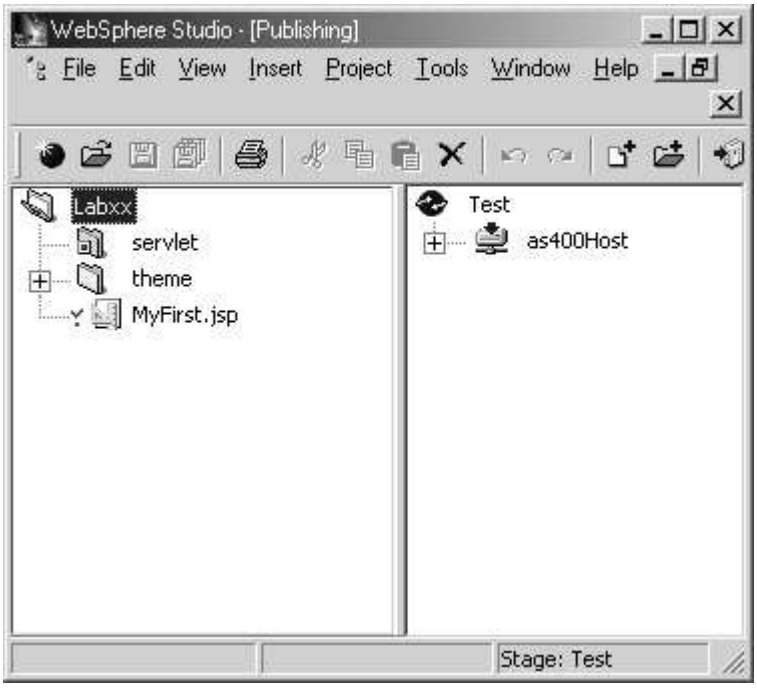


- Press the **OK** push button

Back on the **Studio Main** window:

- Expand the **theme** directory by clicking on the **+** sign beside it

The figure below shows the expanded folder. The Cascading style sheet, Master.css, is one of the default files that gets added when Studio creates a new project.



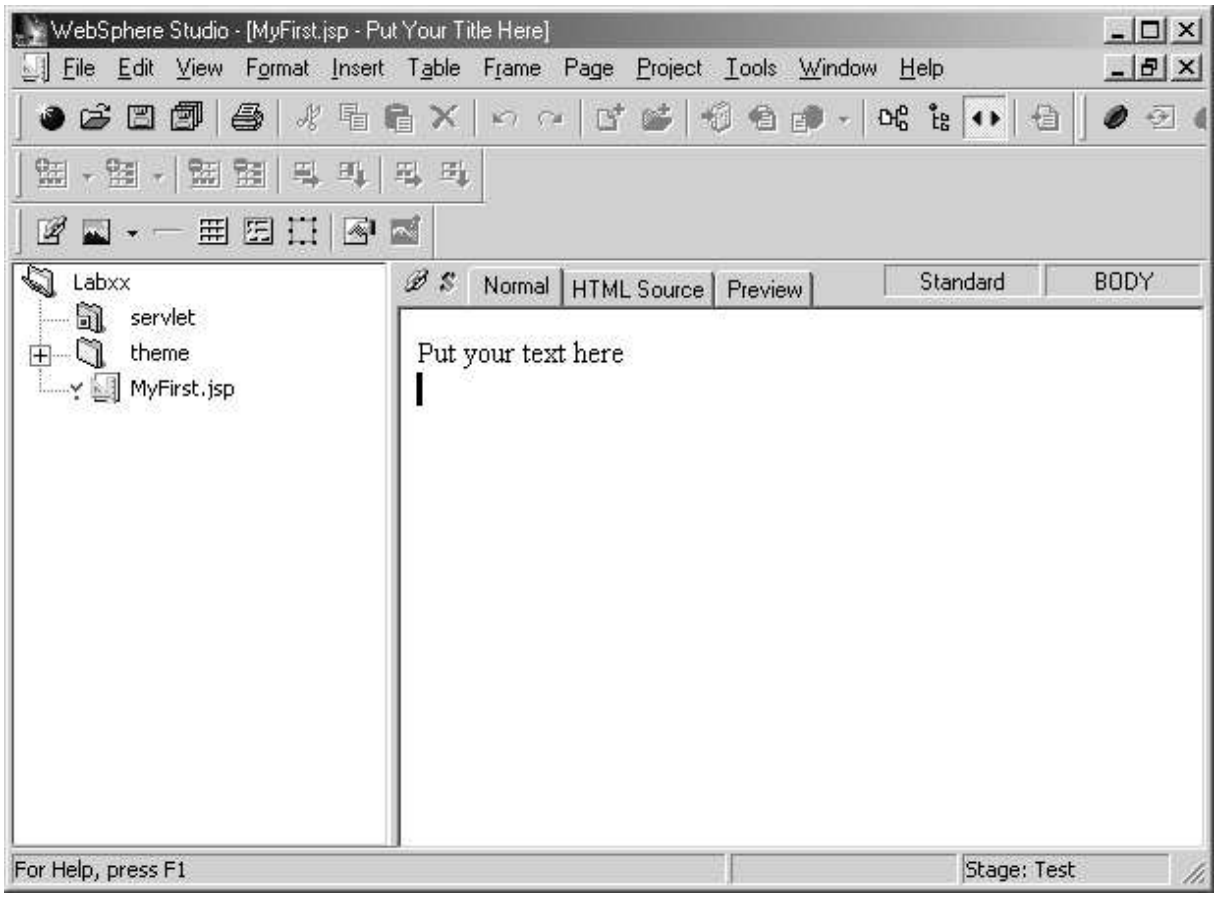
You will notice, the file you just created, is added to the root. You also see a small red check mark next to the file. This check mark indicates that the file is **checked out**. If you would be working in a **team environment** and the file resides on a server, other people working with Studio would not be able to change the file until you **check** this file **in**.

Studio provides full team support for **web development.**

Now you are ready to use **Page Designer**

- Double click on the file icon of your **MyFirst.jsp** file

- The **Page Designer** window will come up on the right side of the Studio main window



This is the window you will be using to design the first web page.

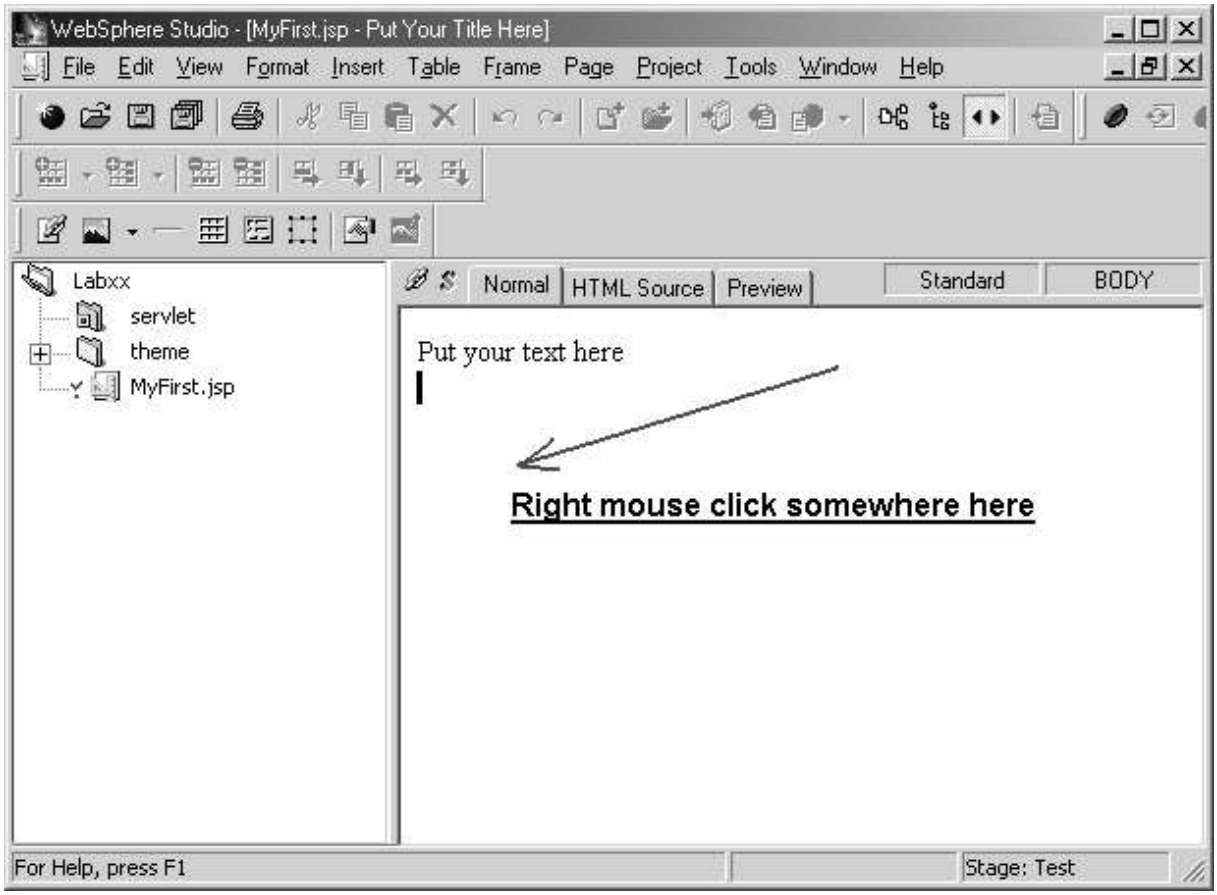
Check the toolbar. You will see that it is expanded and now includes the **Page Designer Toolbar** as well. In the **Page Designer** window itself, you will notice that it allows different views for the file it is currently working with. The initial view shows the normal design window. Tthe following list shows all options available to you.

1. The design window
2. An HTML source editor window
3. A Preview window

Since you are using a blank JSP template file your page is pretty much empty.

Working with page properties

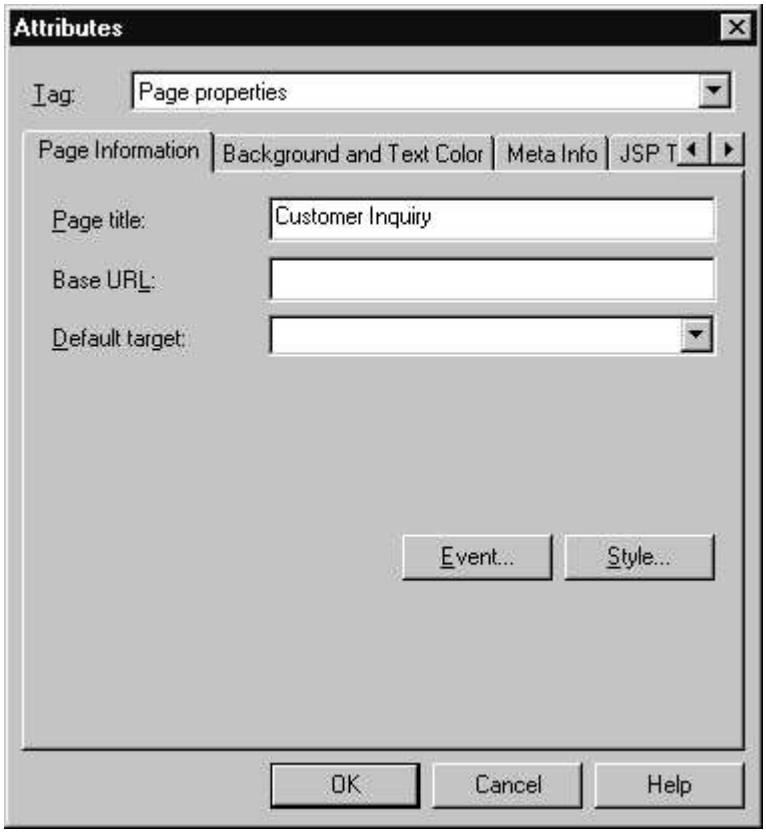
- **Right mouse click anywhere on the white space** in the designer window



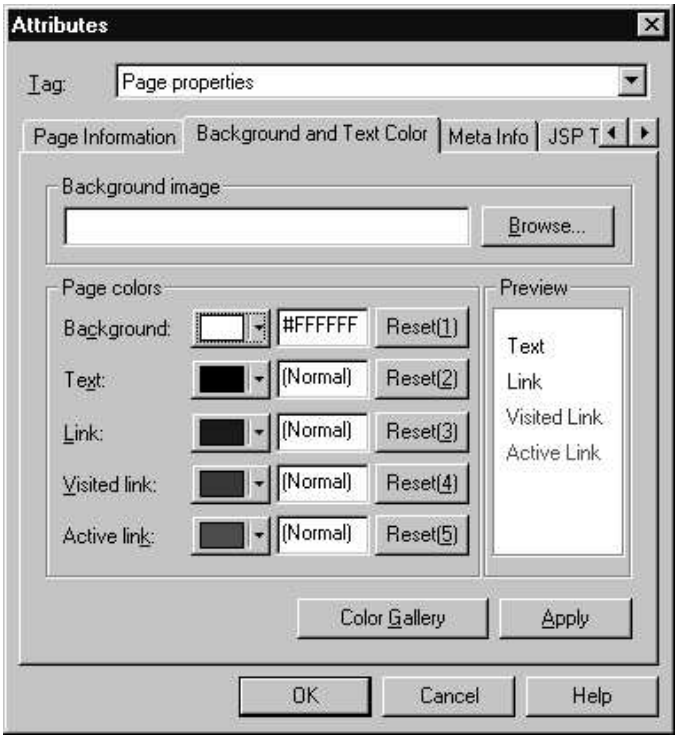
- **Select Page Properties** from the Popup menu

The Attributes dialog will appear

- **Change** the page title entry field to: **Customer inquiry**

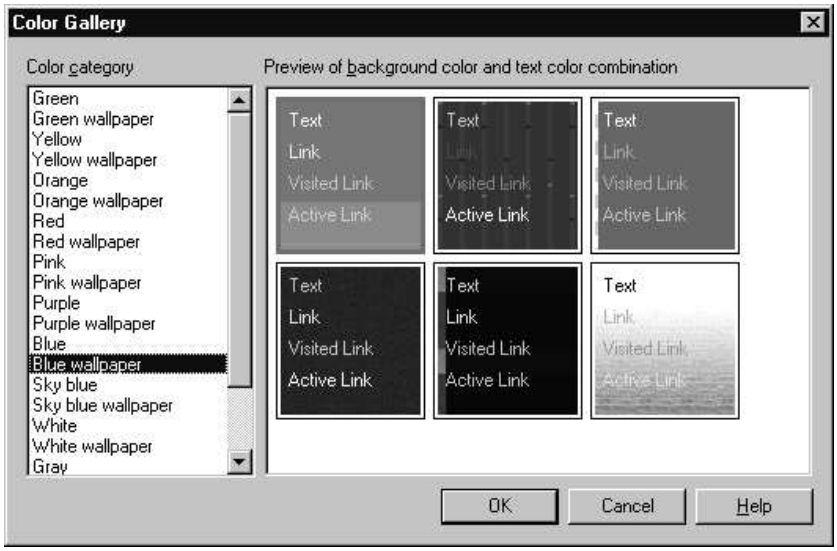


- Select the **Background and Text Colour** Tab



- Press the **Color Gallery** push button

You will get the following dialog:



- Select **Blue wallpaper** from the list on the left side of the dialog

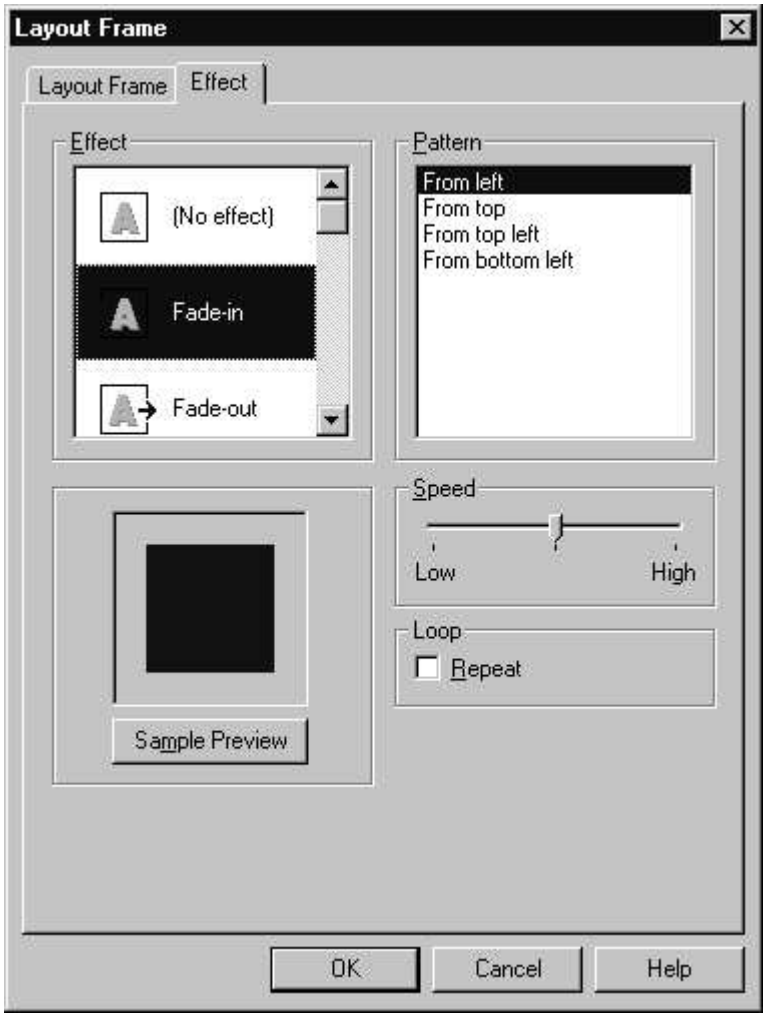
- Select the color choice on the **bottom right** from the **select list** on the right side of the dialog.
- Press the **OK** push button
- Press the **OK** push button to closes the Attributes dialog

Working with a layout frame

Now you will be adding a **layout frame** to your page. A layout frame allows you to specify an area of your page and lay it out as you see it instead of using html tags or tables to do the layout. A **layout frame** also allows some dynamic design to happen on the page when viewed in the browser.

- First Select the text: **Put your text here,** in the design window
- Delete **it**
- Select **Insert** from the **Studio main window** menu bar
- Select **Layout Frame**
- Click on the **Effect Tab**

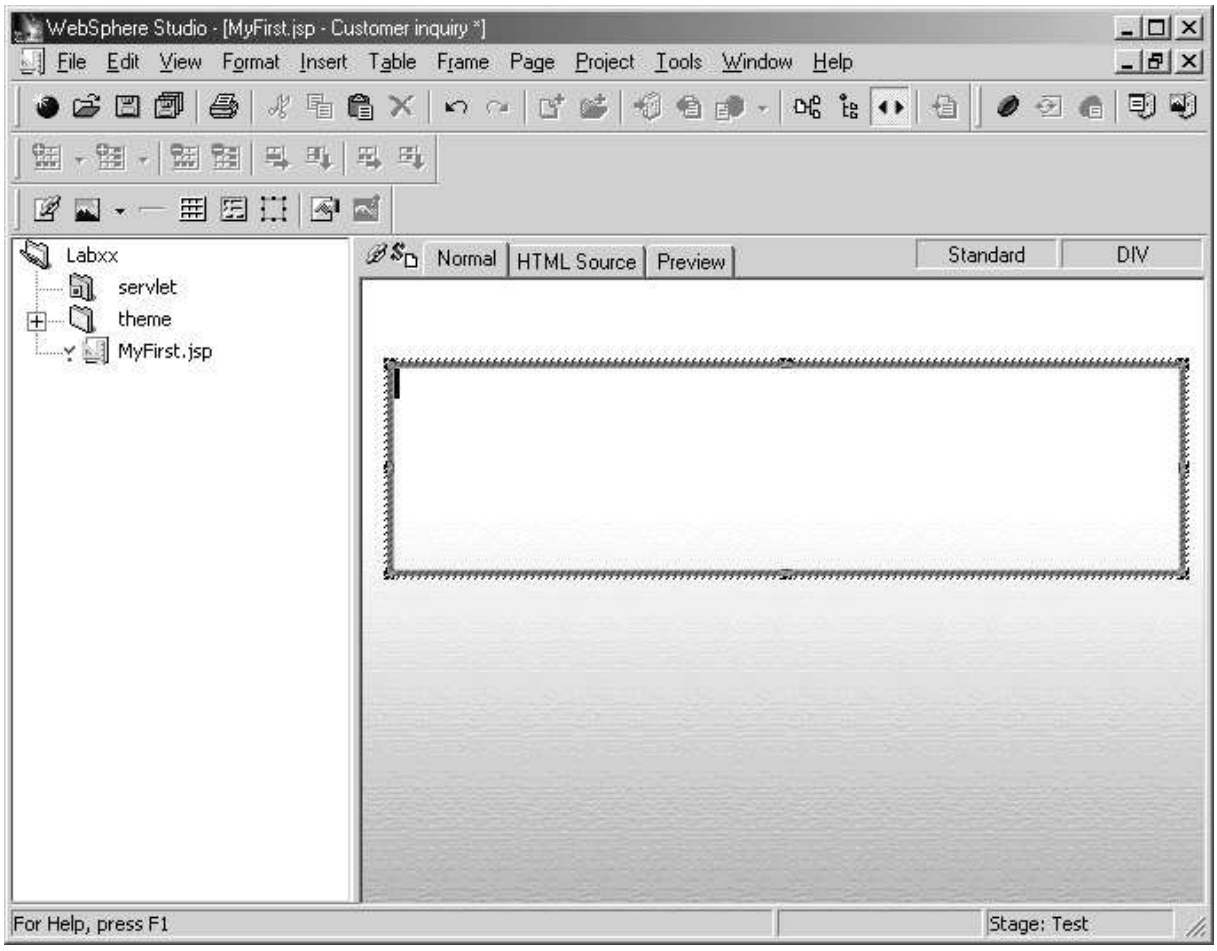
- Select **Fade-in** from the Effect list



- Press the **OK** push button

Your Design window now contains a Frame

- Stretch the **frame by** using your mouse, so it covers the upper quarter of your design window as shown in the figure.

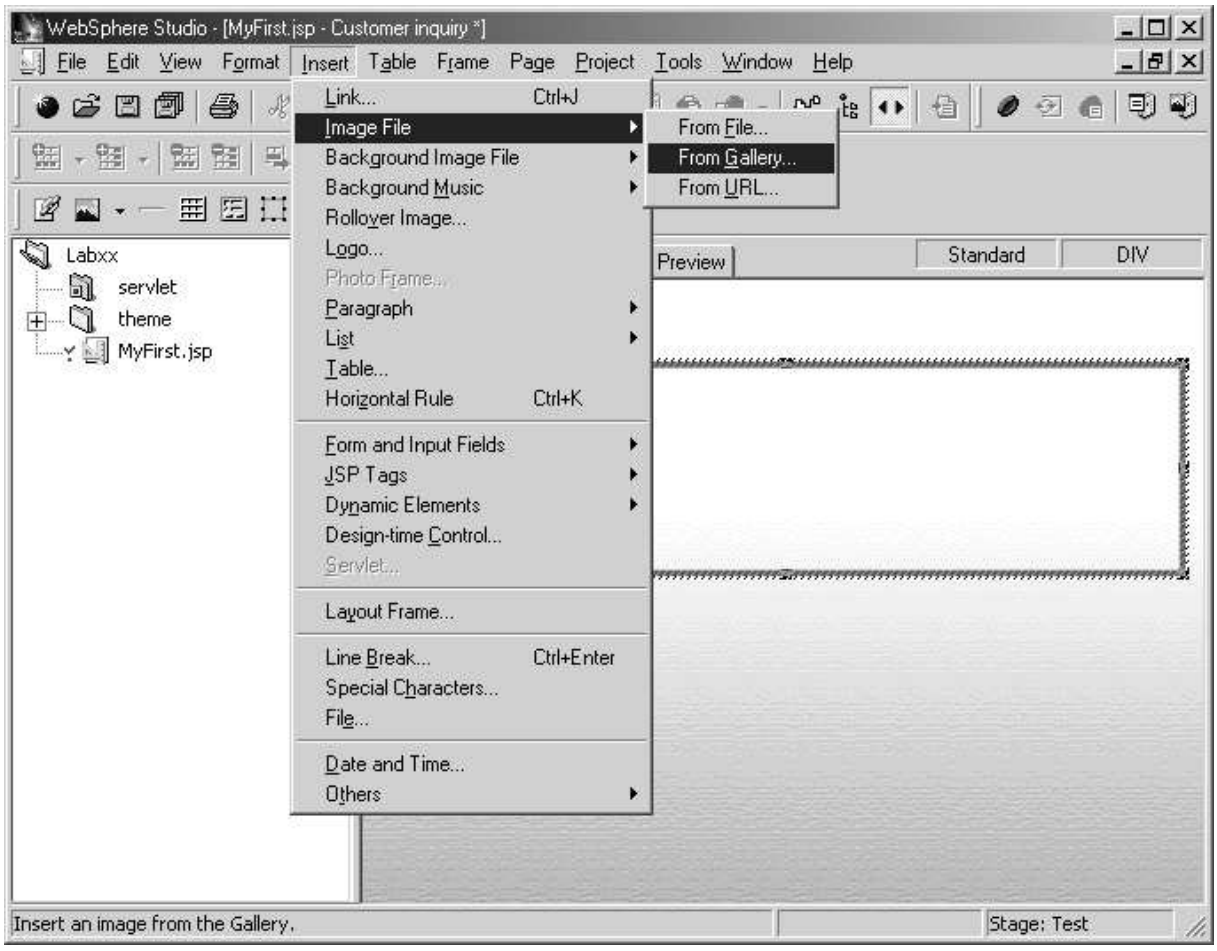


Adding an Animated Image to the Pge

You will now add an **Animated** picture to the frame.

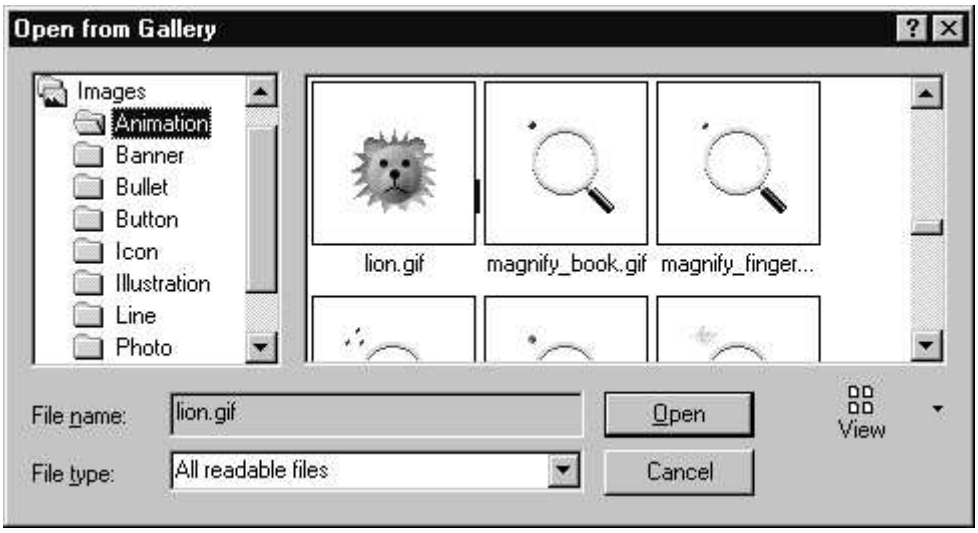
- Left mouse click **inside** the **layout** frame in your design window, so the cursor is positioned inside the layout frame

- Select ***Insert -> Image file -> From Gallery*** from the Studio **main** window menu bar



From the **Dialog** in the **Animation folder**

- Select the **Lion.gif** or what ever image you like best



- Press the **Open** push button

The Lion picture is now part of your page

Adding a Logo to the page using th Logo Wizard

Now you'll add a logo using the Logo Wizard that comes with **WebSphere Studio**

- Click inside the layout frame beside the picture you just added, to position the cursor on the target location for the Logo
- Select ***Insert-> Logo*** from the Studio **main window menu bar**

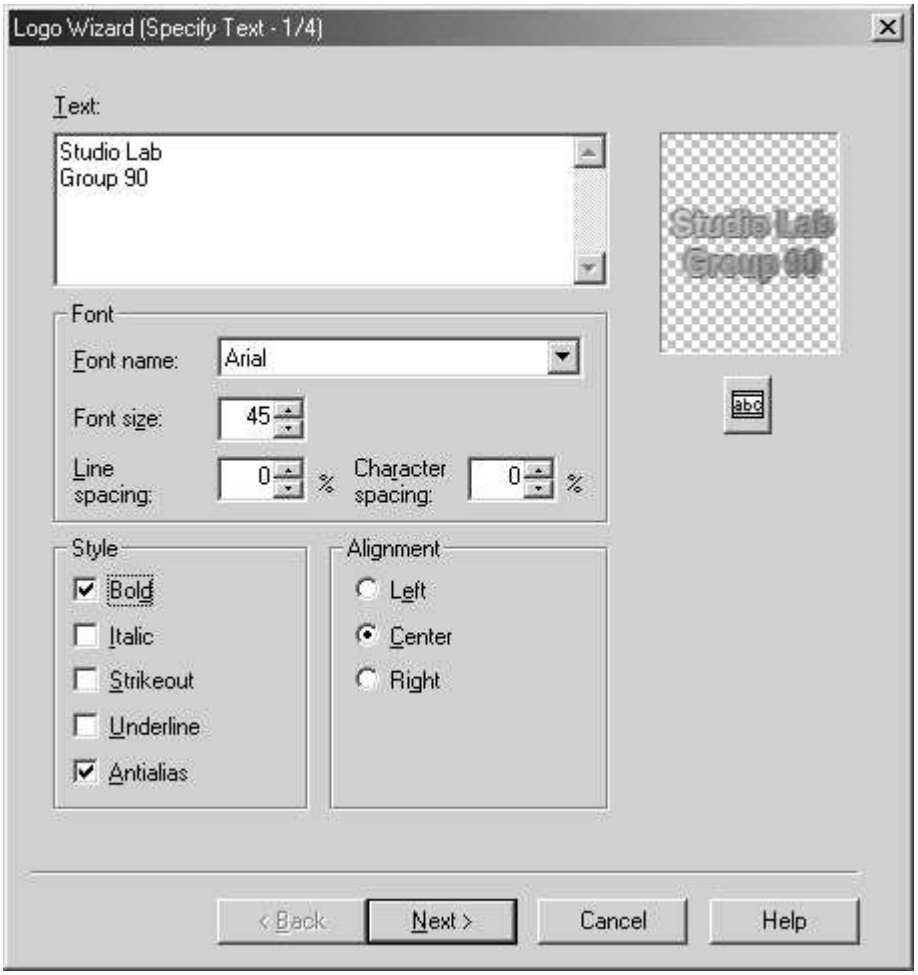
The Logo Wizard dialog appears

In the text field in the top left corner

- **Enter**

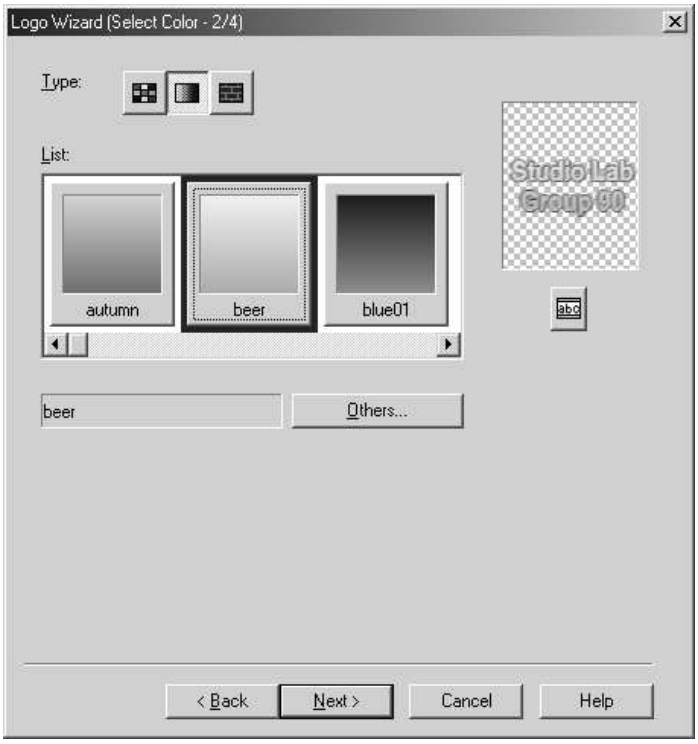
Studio Lab
Groupxx

- **xx** being your Lab group number

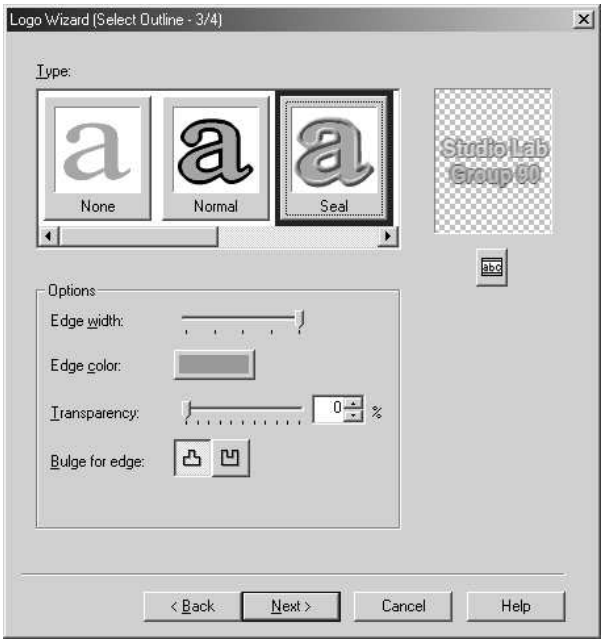


- Select a **Font** that you like
- Select a Font **size**
- Press the **Next**> push button

The Select Color page comes up in the Logo wizard

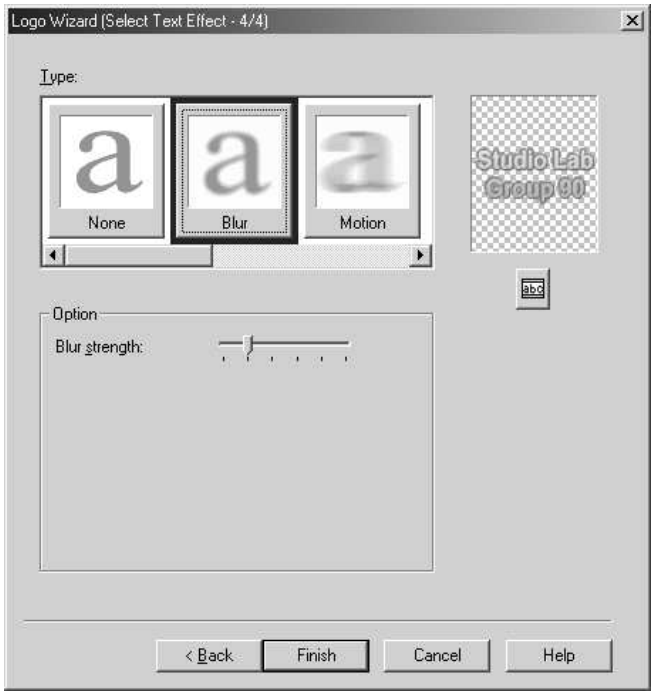


- Select, by clicking on it with the left mouse button, the **middle icon** for Type
- Select**, Beer**, as the color of choice
- Press the **Next**> push button



- Select **Seal** as an outline for your LOGO text
- Press the **Next>** push button

The **Select Text Effect** page comes up



- Select **Blur** for the effect type
- Press the **Finish** push button

You are done with the Logo design. Check your design page

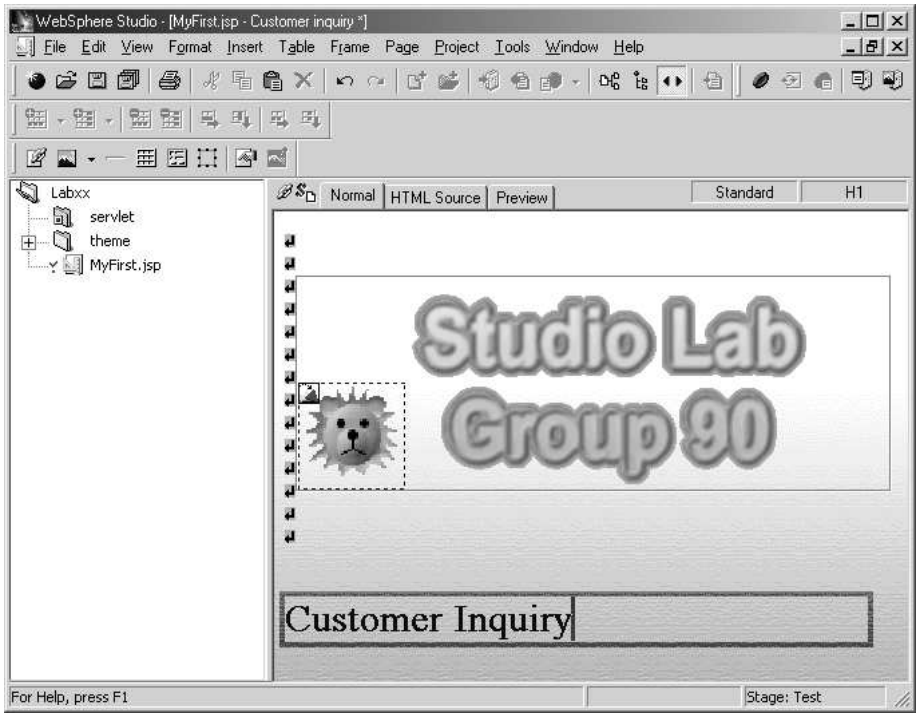
Does it look good? We hope it does.

Creating a heading

Now you will create a heading **underneath** the **LOGO** you just created:

- Press the **left mouse button** above the frame in your JSP page in the **page designer** window. The cursor will be located at the top of the page.
- Press the **Enter** key until the **cursor is located below** the Layout Frame
- Select **Insert --> Paragraph --> Heading 1** from the menu bar on the Studio Main window
- Now in the Design window **key in**

Customer Inquiry



Now you will create an **entry field** and a **heading** for the entry field. This entry field will be used to Enter a customer number. You will send the customer number to an

AS/400 RPG program which will retrieve the customer record and send back the data to a Java Server Page (JSP) to show the data in a browser.

Adding a Form and an Entry Field to the page

In the **second part** of this Lab you will use a **servlet** (a JAVA program that gets generated by Studio/400) to invoke an **RPG** program on the AS/400. In order to invoke this **servlet** you need a **Form** on your web page.

- In the Design window, **click** with your **left** mouse button on the **blue space** underneath the heading that you just inserted.

- **Note:** If the cursor doesn't position underneath the heading, put it **beside** the heading and **press** the Enter key.

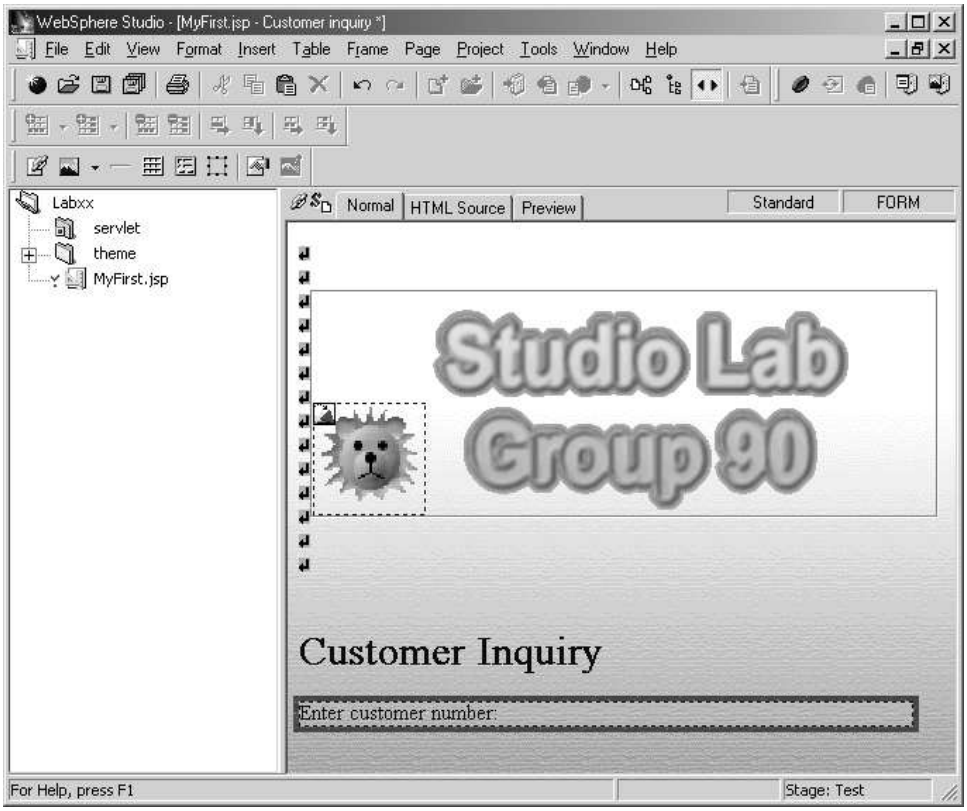
After the cursor is positioned there:

- **Select Insert --> Form and Input Fields --> Form**
  or
  use the **Form Icon** on the Page Designer **Tool bar**

A form that appears as a frame has been added to your design window

- Position the cursor inside the form
- **Key in**

Enter customer number:

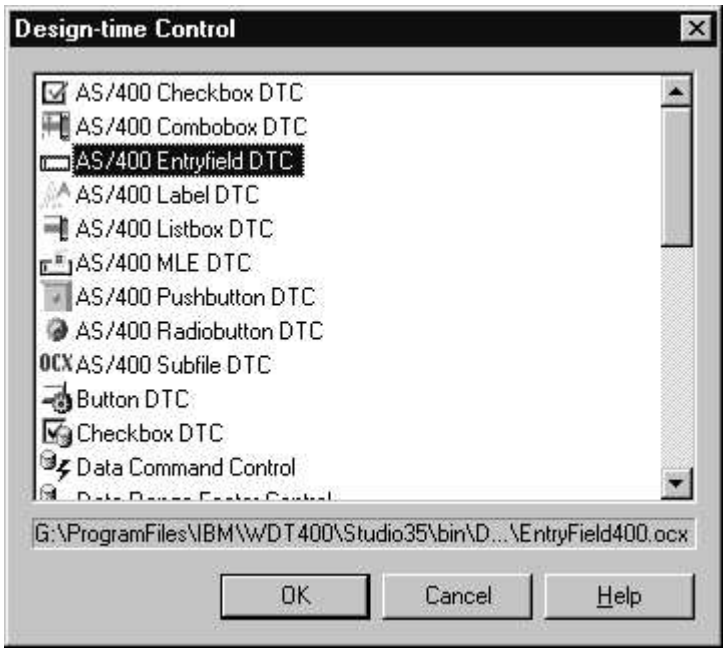


You will now add an **Entry field** to the design page. You will use an AS/400 Entryfield which is implemented as a **Design Time Control** (DTC)

With the cursor located beside the text you just entered

- Select **Insert --> Design-time Control...** from the Studio menu bar
  or
  use the **Design time control** icon on the Tool bar

The figure below shows a list of registered DTC's on a particular PC. It may vary from the list you see on your workstation

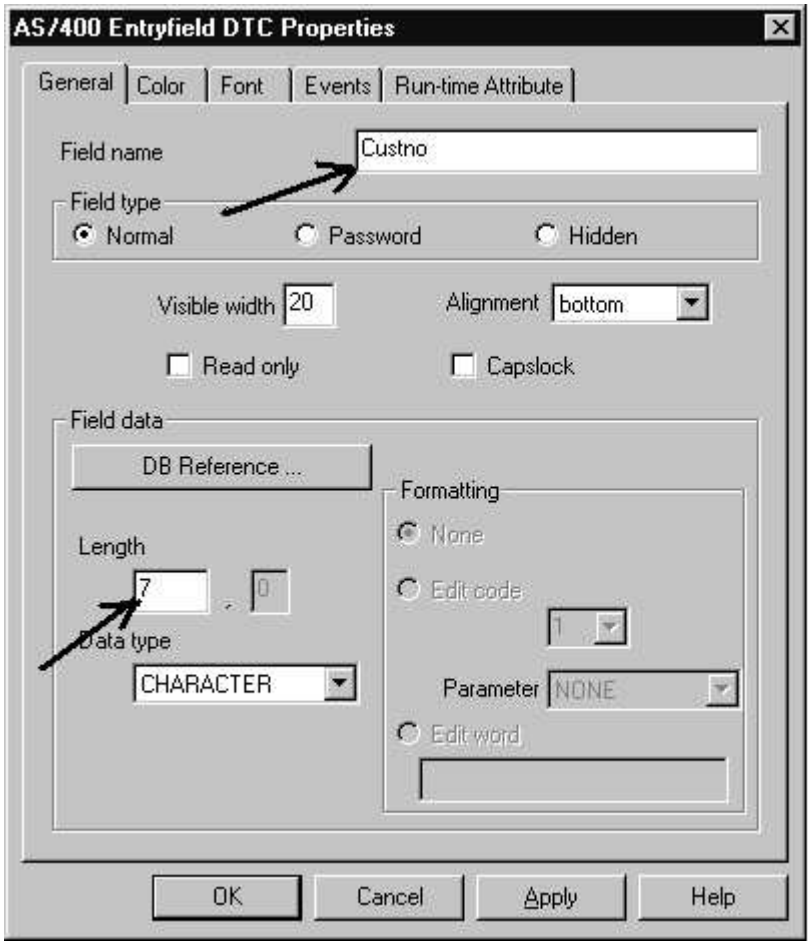


- Select **AS/400 Entryfield DTC** from the list in the Design-time Control dialog
- Press the **OK** push button

On the design page

- Select the **Entry field** you just added
- **Right mouse click** on it
- Select **Control properties** from the **pop up** menu

  On the properties dialog that appears:



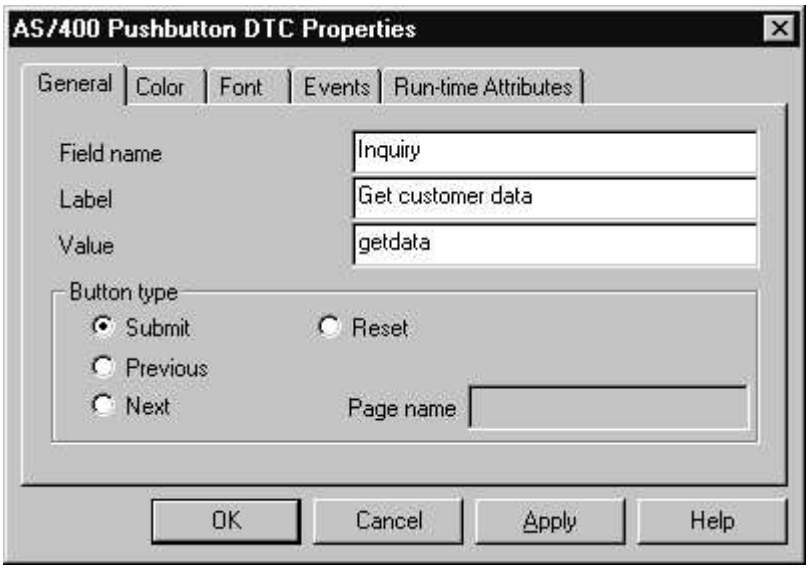
- Enter ***CUSTNO*** for **Field name**
- Enter **7** for the **Length** of the field
- Press the **OK** pushbutton

The last part we need on this page is a push button to send the **submit** request for data to the AS/400 program, so it can **return** some data to the browser.

- Position the cursor **to the right of** the entry field you just added and press **Enter twice**

This positions the cursor below our entry field heading.

- Select on the Studio **main** window menu bar ***Insert --> Design Time Control***
- From the dialog select **AS/400 Pushbutton DTC**
- Right mouse click on the **Pushbutton400** DTC
- Select **Control Properties** from the pop up menu
- Change the **Field name** to ***Inquiry***
- Change the **Label** to ***Get customer data***
- Change the **value** to ***getdata***

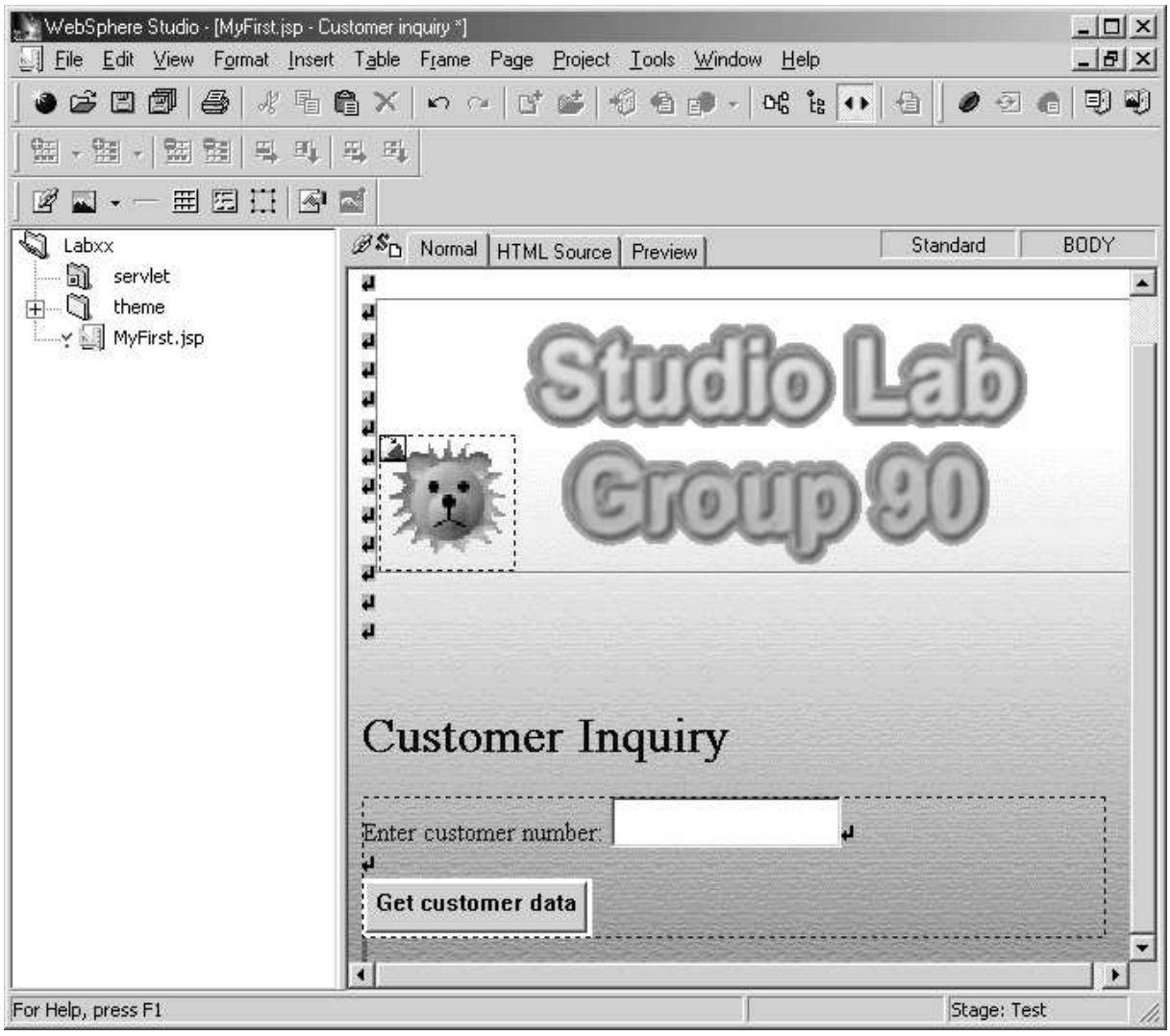


- Press the **OK** pushbutton on the **properties dialog** to change the AS/400 pushbutton properties

Field name is the **control name**,

Label is the text that will appear on the push button

Value is data that is sent to the RPG program to determine which push button has been pressed if you had **multiple** push buttons on a page. This is like the indicator that is set on by different command keys in a **5250** interface.



This completes the user interface design

You might want to check out the additional tabs in **Page designer**

- **HTML Source** shows you the **HTML** tags generated for your design page
- **Preview** allows you to have a look at the page as it would show in a browser.

Close the Page Designer by:

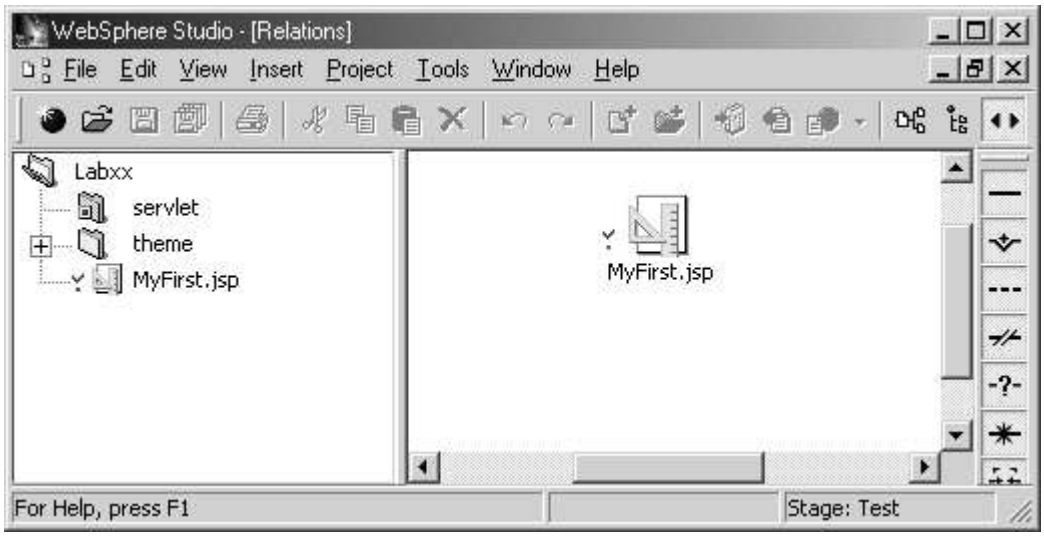
- Selecting the **File** menu option on the **Studio main window**
- Selecting **Close** from the menu pull down

- Press the **Yes** pushbutton to **Save** the JSP File

Relations view

As discussed before, Studio provides a **Relations view.** In this view you can see a graphical representation of **relations between files** in your project.

- Select **View** from the Studio main page menu bar
- Select **Relations** from the menu pull down



- **Click** on the **JSP** file

Since you only have **one JSP** file and a style sheet, this view at the moment doesn't show much for this particular project, but imagine you have hundreds of pages that are inter-linked. This view makes it easy to find missing or wrong links.
Studio is keeping the relations view in sync as you work on your web pages.

Publishing your project

Now you are ready to publish this file to your Application Server. But before you can publish, you need to tell WebSphere Studio the location of your AS/400 Application Server.

In a traditional programming environment you would compile your program source into machine readable instructions to get it ready to run.
In a web environment many of the files you create are **not** to be **compiled**. They are just **rendered** by the browser at runtime. So, they need to be copied to a location where the **HTTP server** or **application server** can access them and send them to the browser.
Until now, as you have worked in Studio, you have stored the files needed for your project on a local disk on your workstation.

Now, you want to make them available to everybody who has access to your web server. You need to copy them to your web server. This process is called **publishing** in the web environment. When you are working with an **application server** you also have to inform the **application server** that the **application** exists and where it can be found.

For **small** web sites you could **publish easily by hand**. For **larger** web sites however, the directory structures become very complicated and it is easy to make mistakes when copying files to their target location. For this reason **WebSphere Studio** lets you set

up your publishing environment once and from then on takes care of getting your web files to the **correct location** when you publish them.

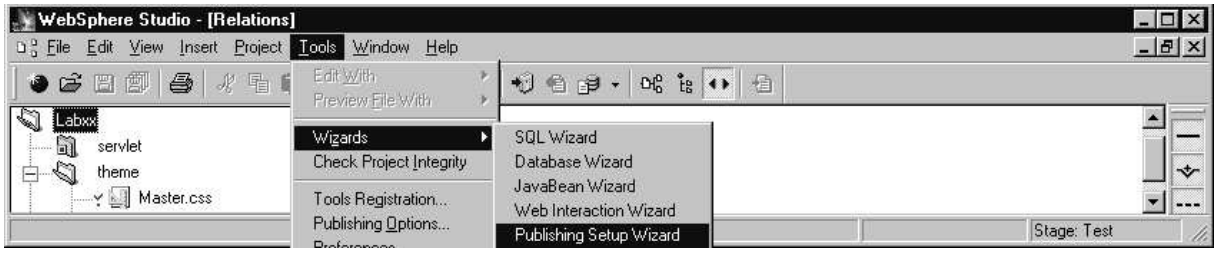
To make it easier to define your application server environment to **Studio** and also to set up the **WebSphere Application Server (WAS)** itself the WebSphere Studio for AS/400 product provides a **Publishing Wizard.**
You will be using this **Wizard** to set up your publishing environment for this project.

Using the Publishing Setup wizard

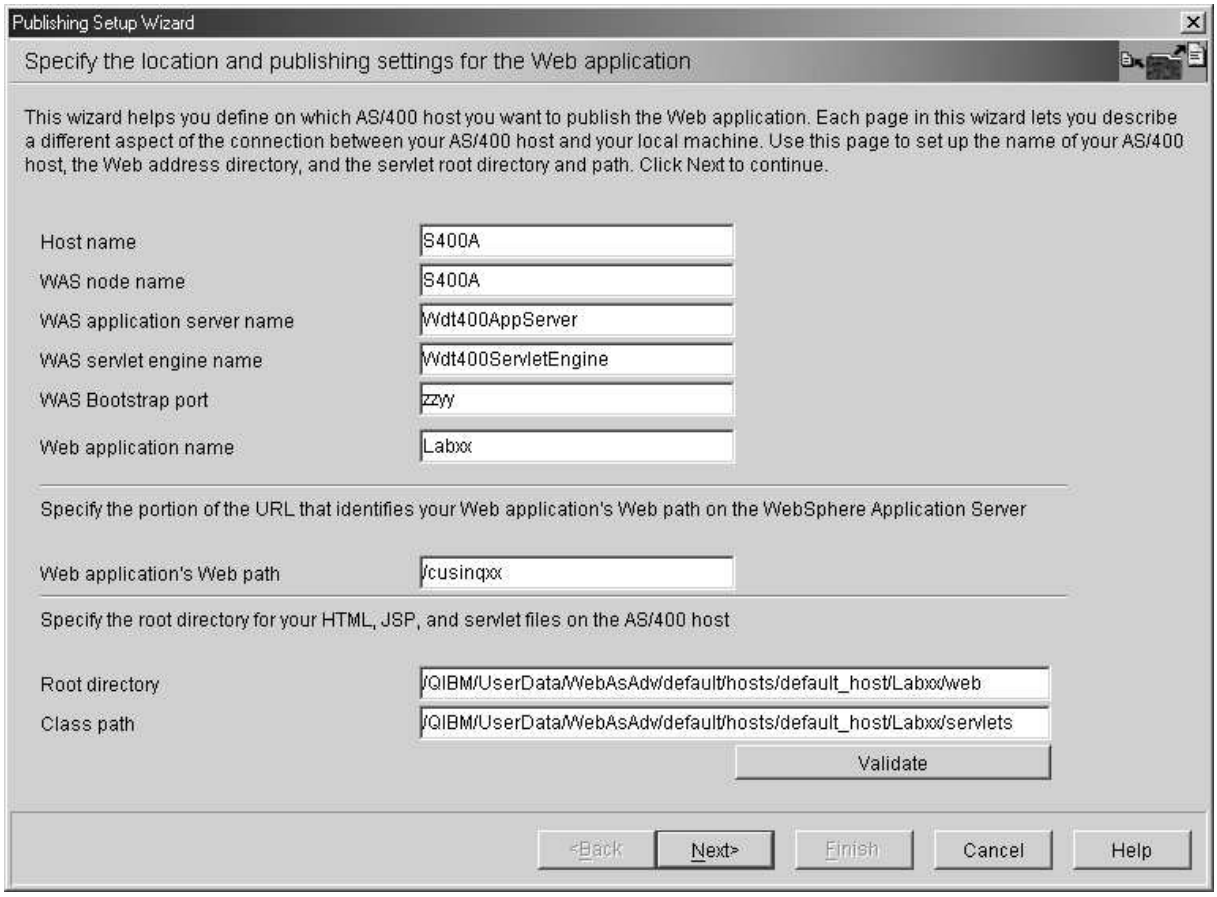
To invoke the **Publishing Wizard:**

- **Click** on your project icon **Labxx** in the **left** window of the Studio main window
- From the menu on the Studio Main window

  Select *Tools --> Wizards --> Publishing Setup Wizard*



The first page of the Publishing Setup Wizard shows and **prompts** you for information about the server environment you want to publish to. Your **web server** will be an **AS/400.** Get the exact name of the server from your instructor



Following is the info what values to use in your particular LAB.

Host name, your AS/400 netserver name -
WAS node name, WAS node name
please ask your instructor for these values !!!!!!

WAS application server name - **LabxxAppServer**
WAS Servlet Engine name: **LabxxServletEngine**

- **WAS Bootstrap port**, **zzyy**
  Note:
  yy identifies the **last 2 digits** of your WAS instance port number your group has been assigned to
  ZZ identifies the **first 2 digits** of your WAS instance port number
  these number are determined when setting up the LAB environment, please
  ask your instructor for both values

- **Project name** is filled in already, **leave as is**
  Web application, web path, key in custinqXX
  Note:
  XX identifies your group number

- **Leave the 2 path entry fields** at the bottom of the dialog as is. The default value is fine for your setup in this Lab

This information will allow the **Publishing Setup Wizard** to send commands to the **WAS** admin instance to get this application registered and to set up the directory structure on the server so it is ready to receive the files to be published.

- Press the **Validate** push button.
  This verifies that the Root directory and Class path exist.

If a Signon dialog appears

- Enter your Userid WDTLABxx

- Enter your password WDTLABxx

Note: **xx** identifies your group number in the userid and Password

You will get the following message, because the directories you just specified do not exist on the server.

- **Press** the **Yes** pushbutton to create the missing directories

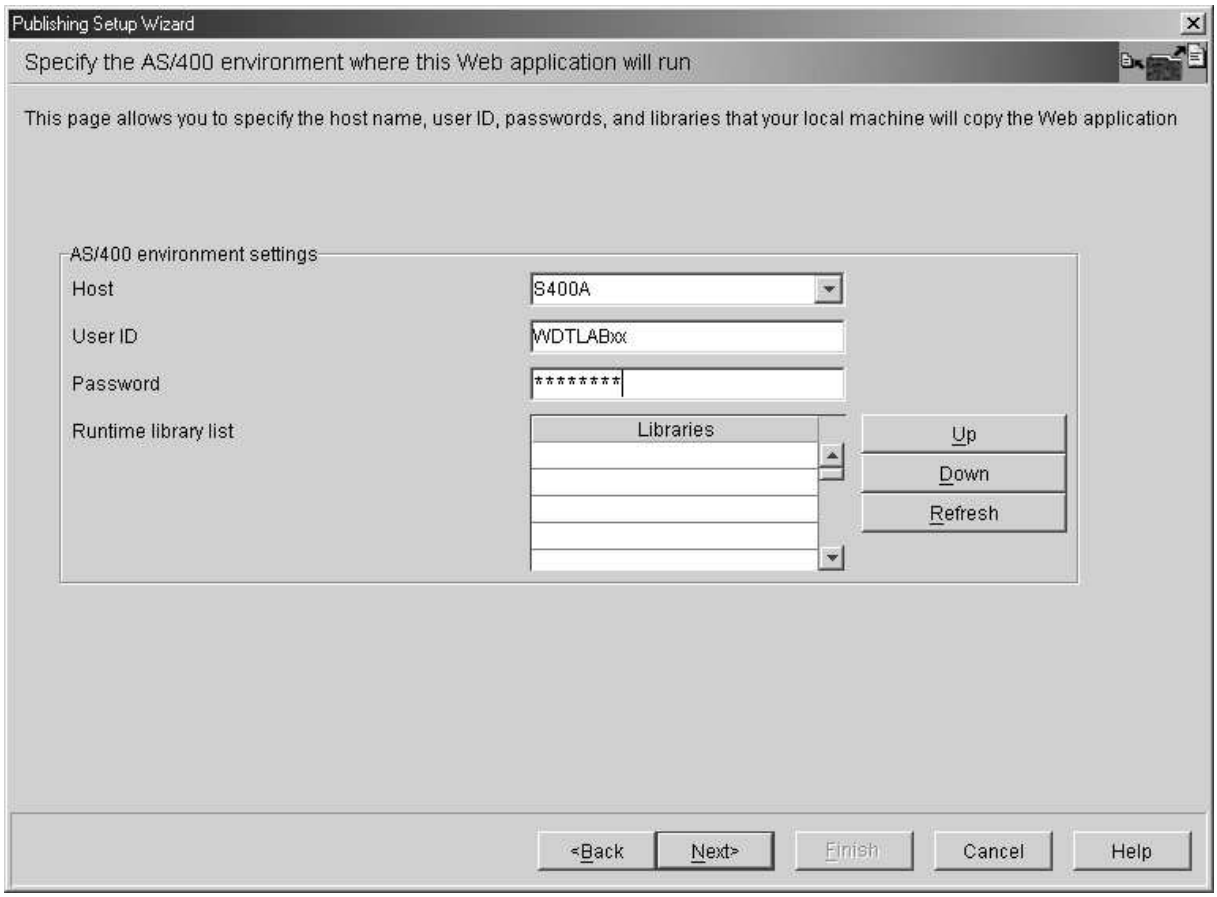
- **Press** the **OK** pushbutton on the confirmation dialog

You are now back on the Publishing Setup Wizard page

- **Press** the **Next**> push button on this page

The next page will allow you to specify your runtime environment on the AS/400. This is not needed for your JSP page that you just built, but the information will be needed

later on. At that time you will use the page you just created to invoke an RPG program. It will fetch related customer data from the AS/400 database.



- **Specify** the **Host name**.

Note:
This **hostname might be different** from the hostname on the **previous page**
Please ask your instructor

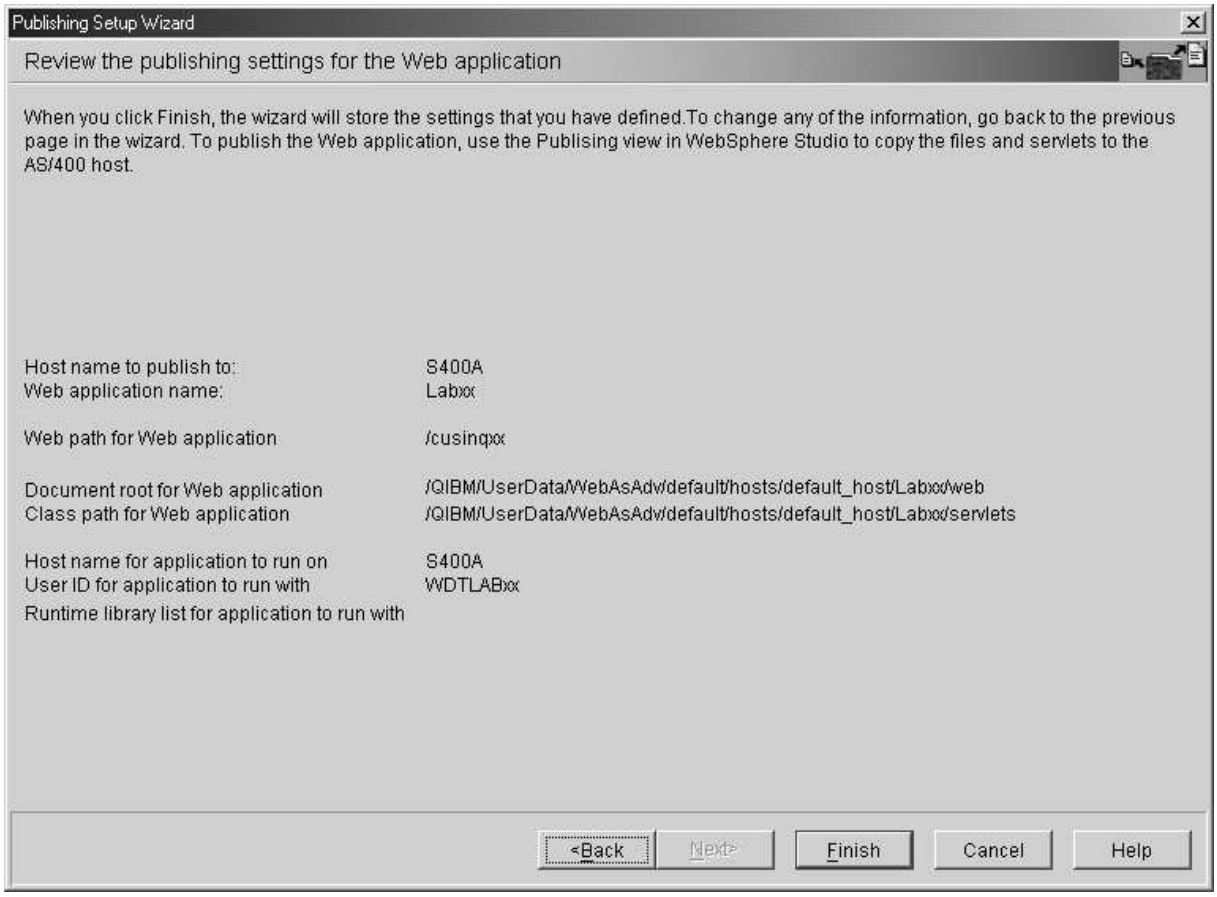
- **Specify Userid and Password** again it is **WDTLABxx**

Note: **xx** being your Lab **group** number

For the **server job** your **RPG** program will run in, you can specify to add libraries to the **library list** at start up of this server job.

- Your library ***WSSLABxx*** is already added to the library list of the job you'll be using, so no need for you to add this library here
- Press the **Next**> push button

A confirmation screen appears



- Press the **Finish** push button

The Wizard now has all the information needed to start the publishing step in Studio.

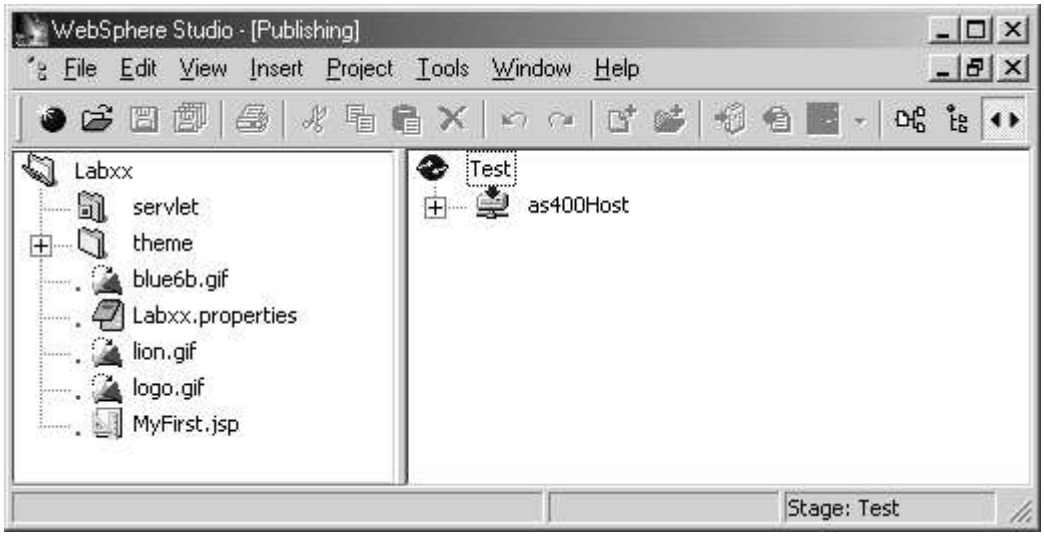
Publishing the project

You are now back on the Studio Main window.

If you do **not see** the **Publishing view** in the right window

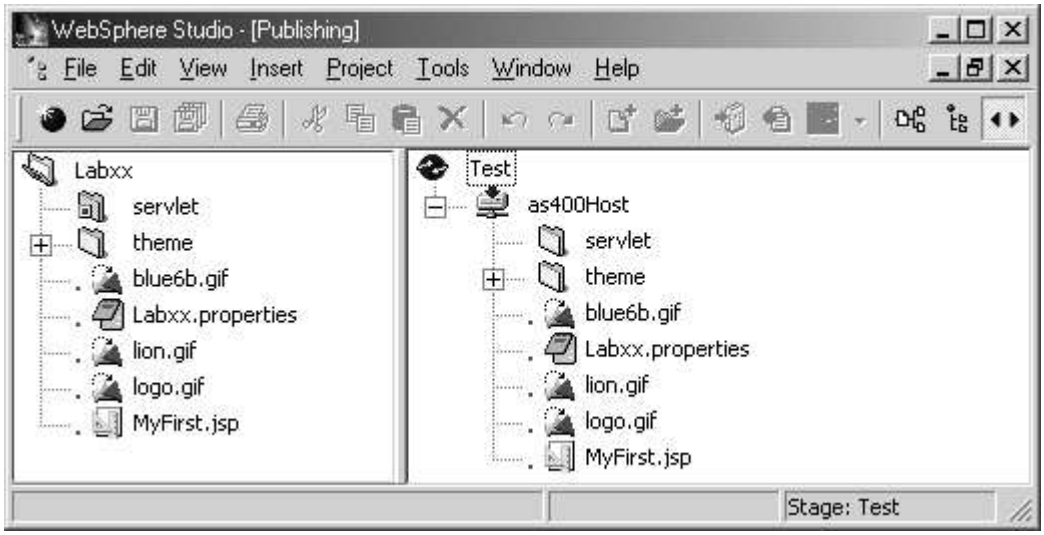
Note: The Studio **title bar** contains information what view is active

- Select **View --> *Publishing*** from the Studio Main window menu bar.



You will notice in the **left window** that an additional file has been added to your project, the file with the name **LABxx.properties**. It has been generated by the **Publishing Setup Wizard**

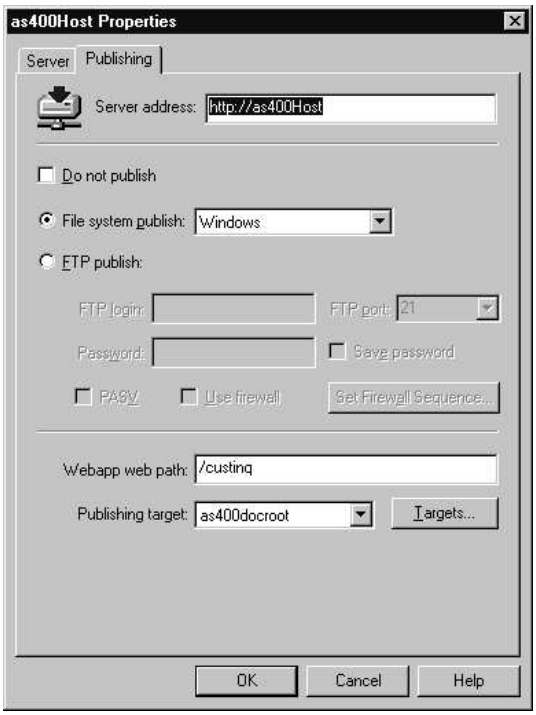
Now with the publishing view active you are ready to publish your Project.



At this time we want you to check the properties of your target **as400Host** server:

- **Right** mouse **click** on the **as400Host** icon

- Select **Properties** from the Popup menu



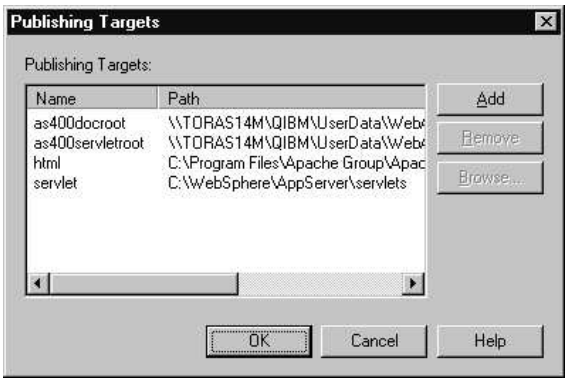
The **publishing setup wizard** has already prepared the publishing target and set up the values for you.

- The publishing will be done through the file system, meaning you need to have the TCP/IP server ***NETSVR** running on your server AS/400 to allow Studio to publish the files

- The Web application path has been established. It is called **/custinqXX** as you specified on the first page of the Publishing Setup Wizard.

To look at the target directories,

- Press the **Targets** push button, beside the Publishing target combo box in the lower half of this dialog



You can see that 2 publishing targets have been added to this Studio project. This information has been added by the Publishing Setup Wizard. The other 2 targets are defaults for the Studio project for a local server.

- Press the **OK** push button, to close the Publishing Targets dialog
- Press the **OK** push button, to get back to the Studio main window

You are almost **set** to **publish**

- Look at the **left** Studio window and **expand** the theme directory (**click** on the **+ beside** it) if it is not expanded already.

You might notice some files that still have **red check marks** beside them, meaning they are still **checked out**,

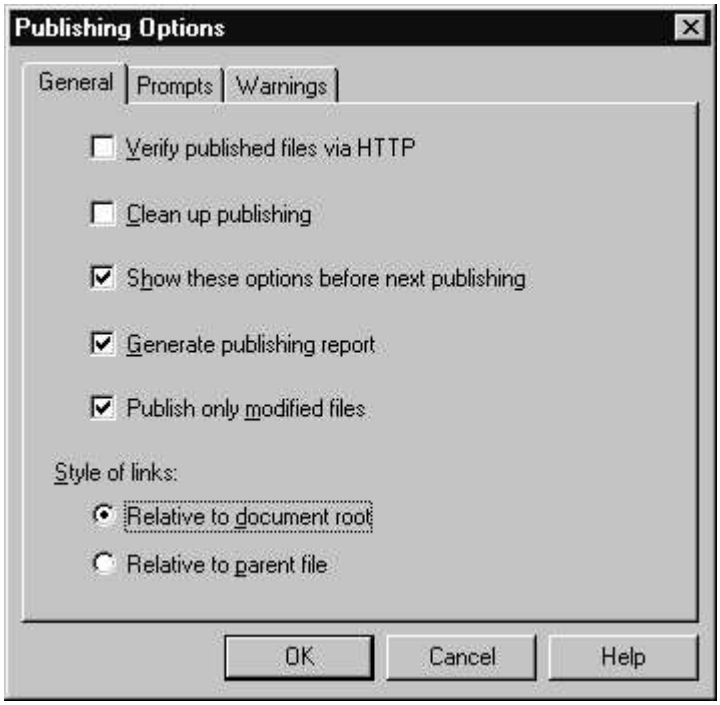
If you see **checked out** files do the following:

- Right mouse click on the **project icon Labxx**
- Select ***Check in*** from the **pop up** menu

The check marks should be **gone**:

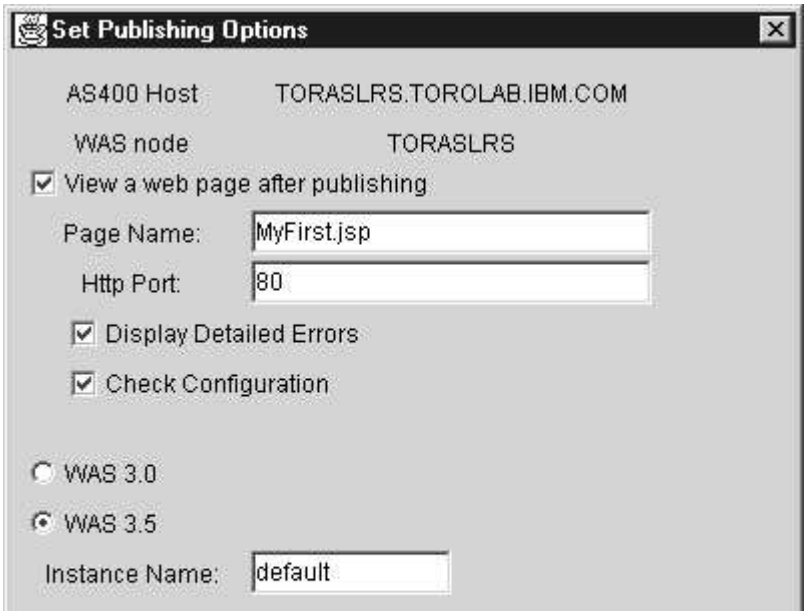
- Right mouse click on the **project icon Labxx**
- Select ***Publish whole Project*** from the **pop up** menu

The publishing options dialog appears



- Accept the defaults by pressing the **OK** push button

A little status window will appear and then you see the **Set Publishing options** dialog



Select the **check box** for ***View a web page after publishing***

- Key in the name of your JSP file (**MyFirst.jsp** if you followed our suggestion)
- Key in the http port number,
  Note: this is different than the WAS port number **41yy**

- If the instructor did not provide the **http port number** please ask

- Also **check** the check boxes for
  - **Display Detailed errors**
  - **Check Configuration**
- Check the **WAS3.5** radio button
- **Key in the WAS Instance Name** *WASyy*

- **_Ask** the instructor, **if you are not sure what name to use**

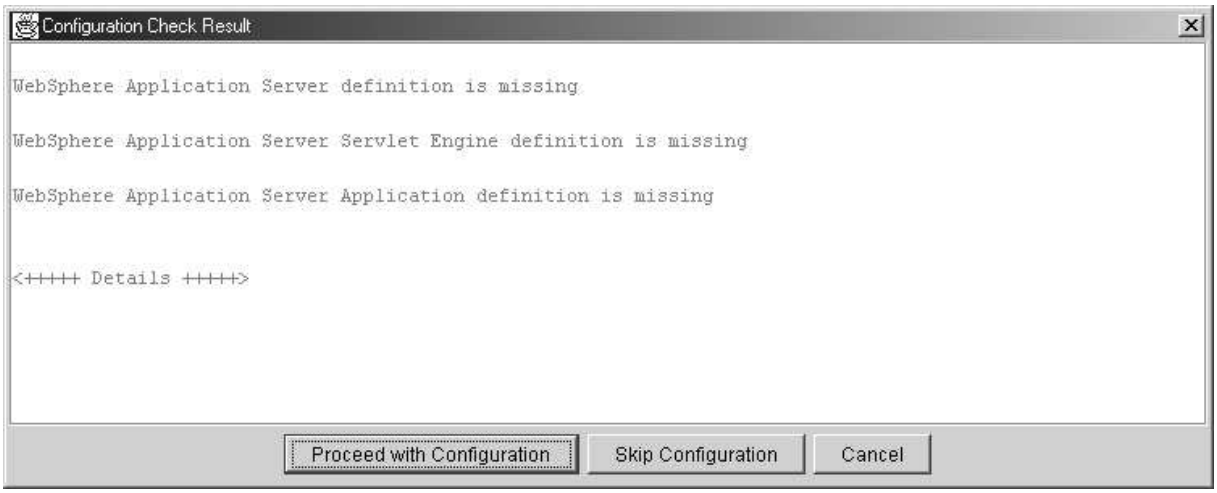
-
- Press the **OK** push button

Since you specified **Check Configuration** on the previous dialog,

the **publishing wizard** will access the **WAS** instance and interrogate the server environment

-

You will see the following confirmation screen



It just acknowledges that the definitions for:

- Application server
- Servlet engine
- WebApplication

are missing in this WAS administration instance.

If the dialog looks a bit different and it states that the **Application server** and/or the **Servlet Engine** are **compatible** that is **fine** as well. Just proceed with the next step.

Note: If the dialog contains **4** messages and the top message states that the **Node** is missing, then you have a problem, you need to ask your instructor for help.

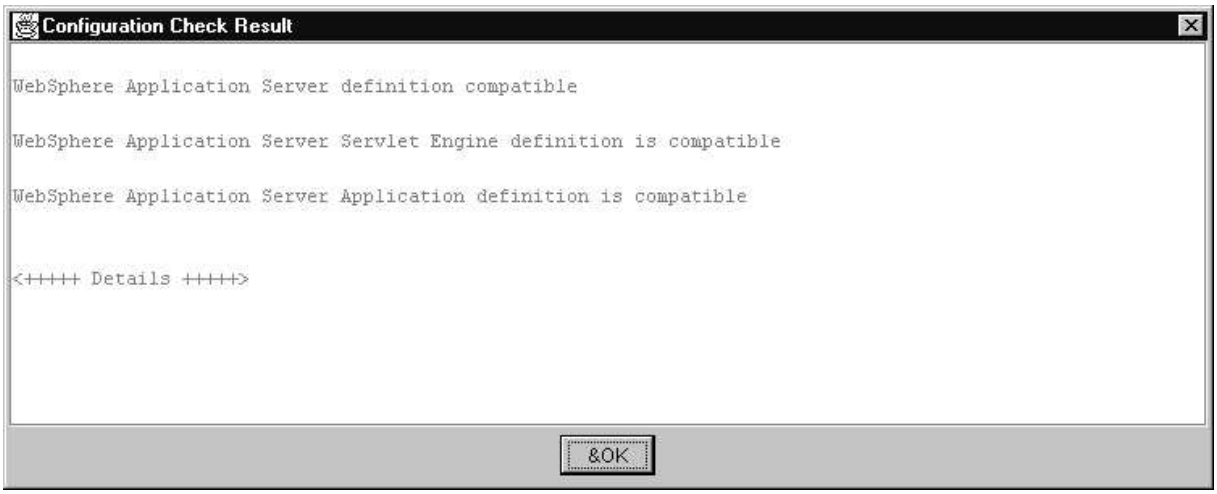
- **Press** the **Proceed with Configuration Pushbutton** to add the missing pieces to the administration server instance, so the application server is aware of this application.

- **Note:** You could compare this step with creating a subsystem to enable jobs to run and adding a library to a job description to enable the system to find the programs in this library without full qualification.

Now the Publishing Setup wizard will send a XML command stream to the **WAS** to add the application to the server definition and then **Stop** and **Start** this **WAS** instance.

This will take a moment.

You should **eventually see** this window



Confirming that the application is now configured in WAS

- Press the **OK** push button

Now the **publishing** step will finish. For **any messages that might come up** during this step:

- Press the **OK**, **Yes**, or **Yes to all** push buttons.

2 browser windows will appear

1. With a **publishing confirmation** page
2. **With the page you just built**

Check the publishing report **and check your web page**.

At the moment **nothing will happen** when you **press** the **Get Customer Data** push button because you **haven't specified** any action to be taken when the push button is **pressed**.

The code for putting some action behind the push button will be **added** in the **next Lab**.

Lab Summary

This concludes the **first** lab.

Congratulations, your first **web page** is completed:

You have learned:

- How to create and design a **web page** using the page designer tool in Studio
- How to create a **dynamic image**
- How to create a **LOGO**
- How to use **AS/400** Design time controls (**DTC's**)
- How to use the **Publishing Setup wizard**
- How to **publish** a web site using Studio's publishing support

Now you are ready to build a web application that accesses data on the AS/400 and returns it to a web page.

Just experience how easy it is.

Go ahead to the next Lab.

Enhance Your Application with logic to access DB2/400 and return data to your web page

Exploring the Studio/400 Web Interaction Wizard and invoking AS/400 programs

During this exercise you will enhance the application that you created in the previous lab. The following tasks will be covered in this lab:

- Adding a Result Window to an existing application with the Web Interaction Wizard
- Referencing fields in a DB2/400 database
- Creating the Program Call Markup Language (PCML) definition of the parameters being passed to/from the AS/400 program
- Write a Service Program or a regular program to interact with web pages
- Generate JAVA code for Servlets using the Interaction Wizard

As a result of this exercise, you will create a web page that is used to display customer information given the input from the web page you created in the previous lab.

What You Should be Able to Do

As a result of this exercise you can use the **WebSphere Studio/400 Web Interaction Wizard** and **CODE/400** to write the necessary program logic to create a dynamic web environment with RPG. You will be able to access an AS/400 from a web browser and:

- Use existing **DB2/400** field definitions to create the necessary PCML information to be able communicate with an AS/400 program
- Use **externally described AS/400 fields** in your web pages
- Get data from the AS/400 by invoking an AS/400 program that accesses the database and returns the data via parameters

- Read and write data from/to a web page
- Build/publish an RPG web application accessing AS/400 data
- Run an RPG web application accessing AS/400 data

Using the Studio/400 Web Interaction Wizard to connect with AS/400 programs

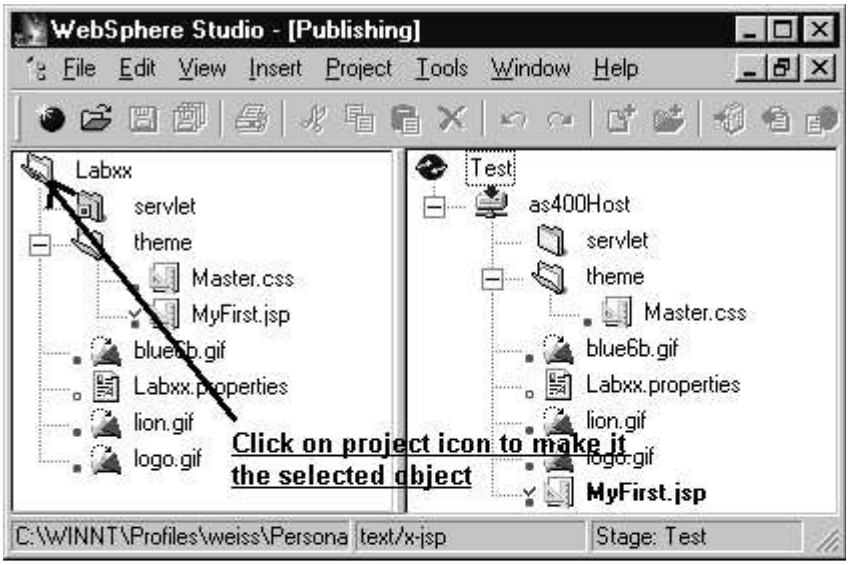
During this exercise you will use the interaction wizard to create the interface description for parameters to be passed to an AS/400 program. You will also generate a web page from the database field definitions of an existing AS/400 database file.

Using the Web Interaction Wizard

If WebSphere Studio is not up, **start it** and specify to use the existing **Labxx** project that you created previously

To invoke the Web Interaction Wizard:

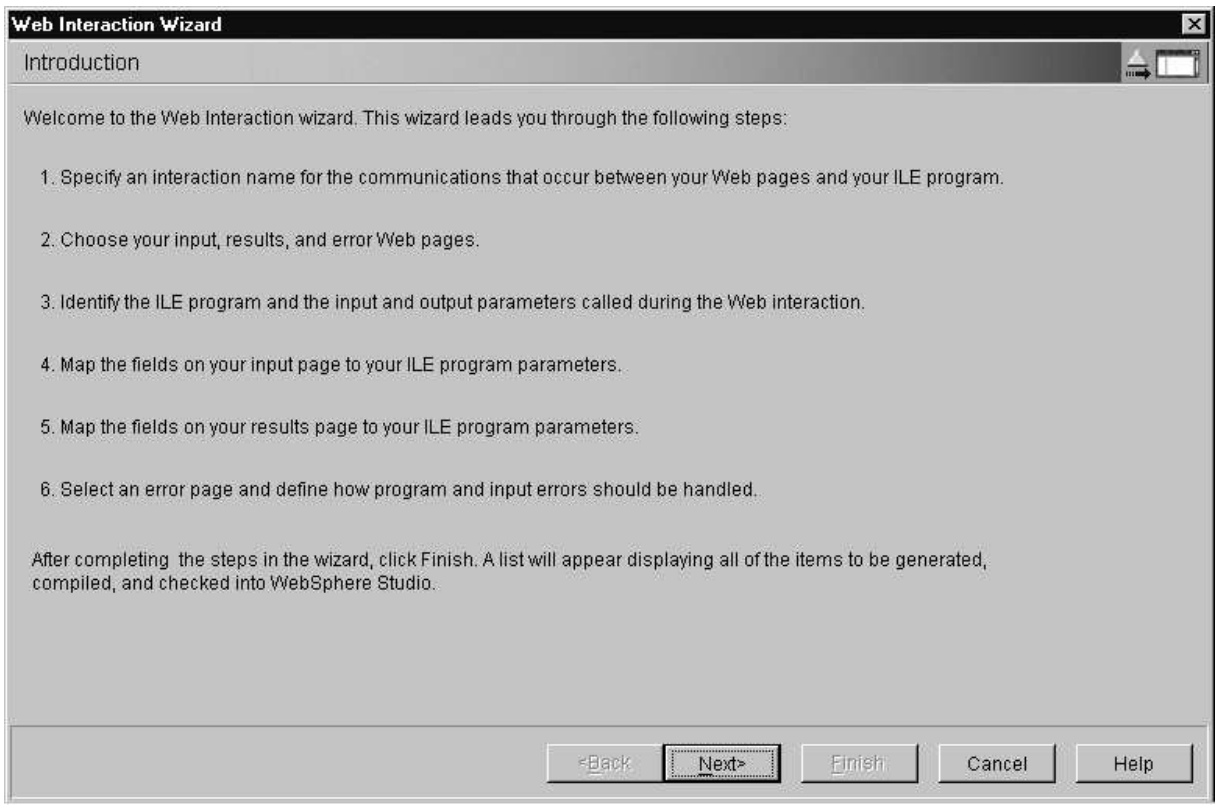
- Left mouse click on the **Labxx project icon** in the **left hand view**



- **Select** from Studio main window menu bar ***Tools --> Wizards --> Web Interaction wizard***

The Web Interaction Wizard introduction dialog will come up,

- **Read** through the step by step introduction to get an idea what the wizard will guide you through:

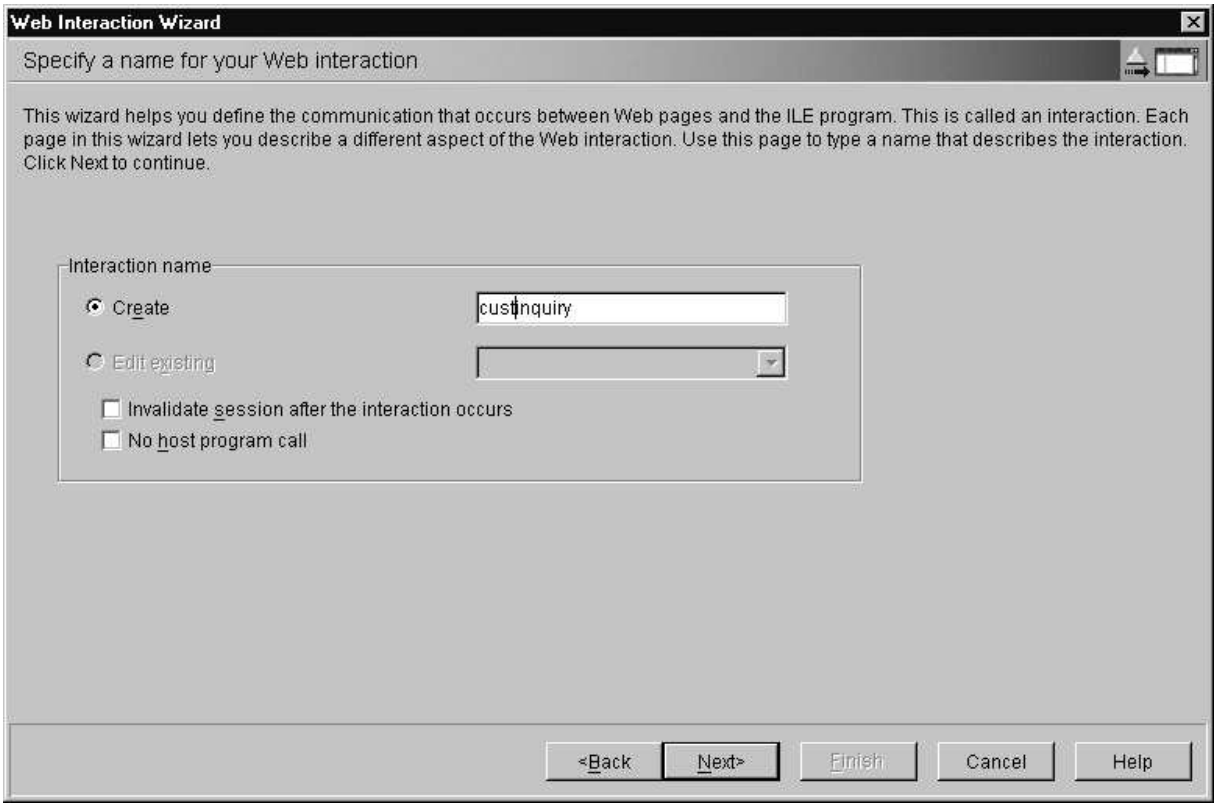


- Press the **Next** push button on this page

Creating a new interaction

An interaction is the **span between** the **submit** of a request from the current web page to the **post** of a new web page. An interaction could be a simple request to the HTTP server to load another page. In this environment an interaction will be made up of a call to an AS/400 program, waiting for the return of the program and then invoking a **JSP** to show the next web page.

First you have to give the interaction a **name** so you can reference it later on. You will see the following screen

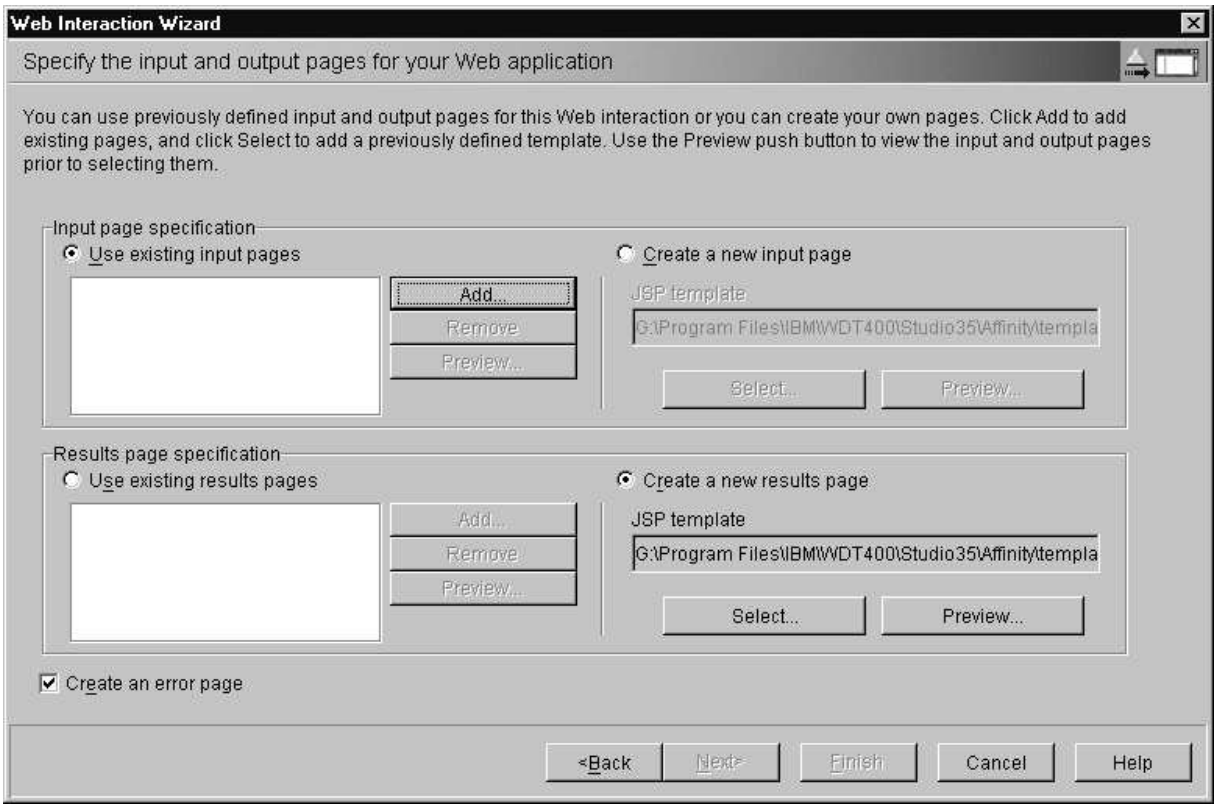


- **Key in** the interaction name: ***custinquiry***
- **Don't check** the check boxes
- **Press** the **Next**> push button

Defining the web pages to work with in this interaction

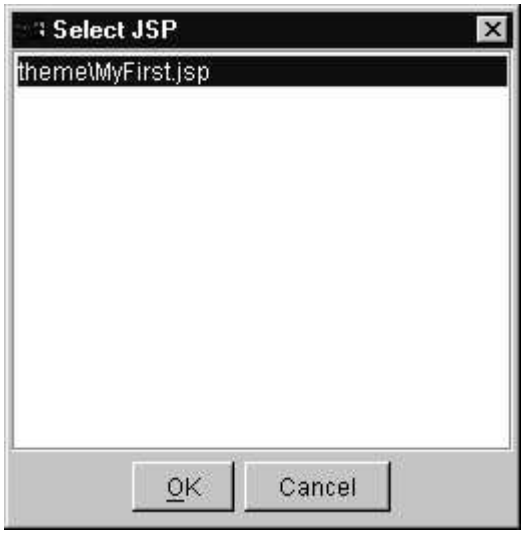
The next page of the interaction wizard asks for information about the web pages that are involved in this interaction. You have an **input page** that invokes the interaction and an **output page** that completes the interaction and will be shown in the browser at

the end of the interaction.

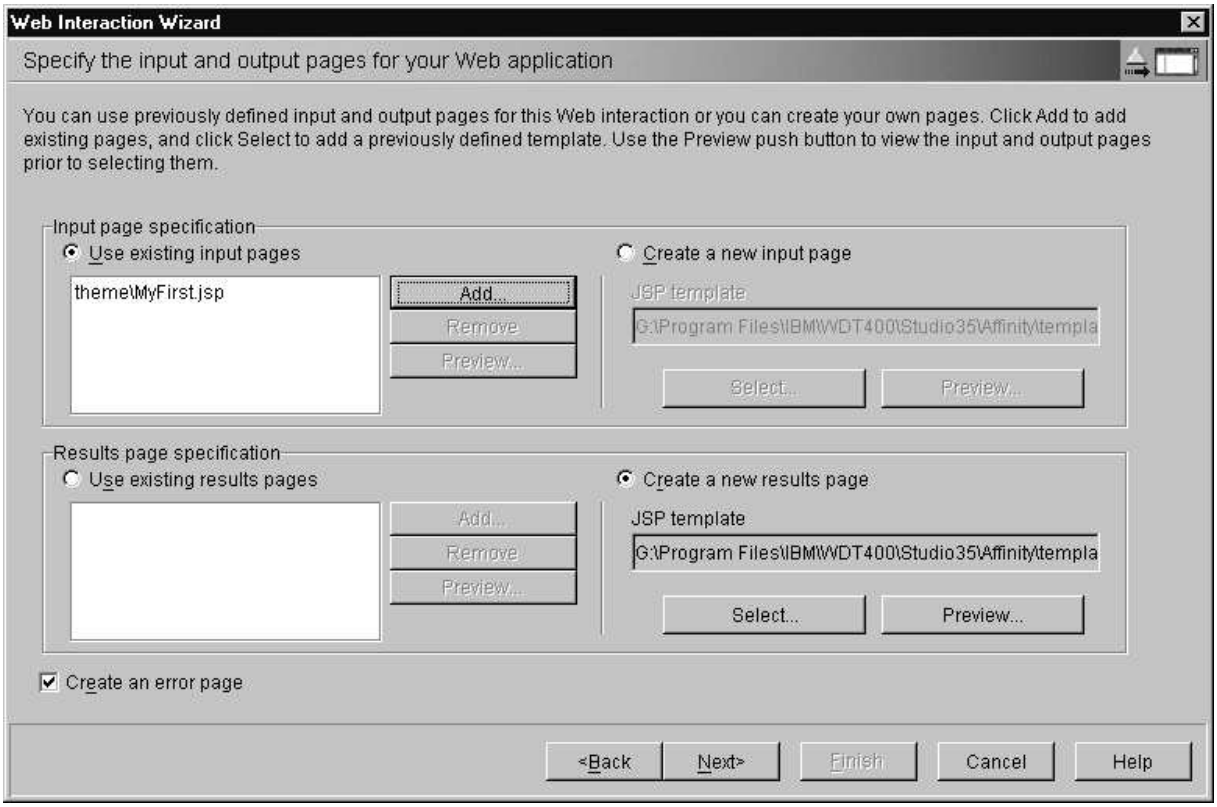


- Check the Radio button for: ***Use existing input pages***
- Press the **Add** push button inside the **Input page specification** group box

This brings up a list of pages that can be used in this project, since you only have one page created in this project, only **one page** is shown in this list.



- Select your **MyFirst.jsp** file, as the input page to be used in this interaction
- Press the **OK** push button



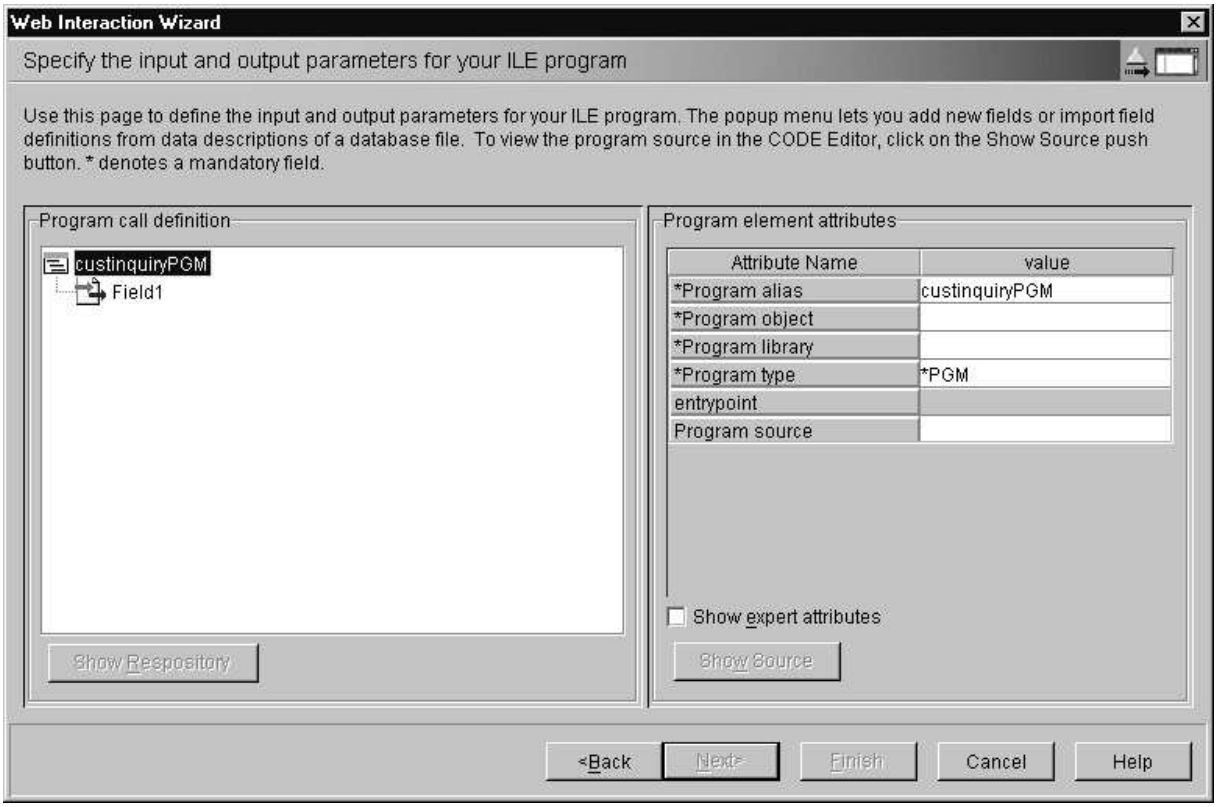
Specifying the output page

You don't have a result page to be shown when this interaction is completed, so the wizard gives you a choice to generate a page for you. Compare this to **DFU**, where you can select database fields to work with and DFU creates the user interface for you.

- Leave the Radio button: ***Use existing Output page***, **unchecked** to indicate that the wizard should generate it.
- Press the **Next** push button

Defining the AS/400 program invocation and parameters

The next page will ask you to define the parameters you want to pass between the Web environment and the AS/400 program that serves the requests from the web.



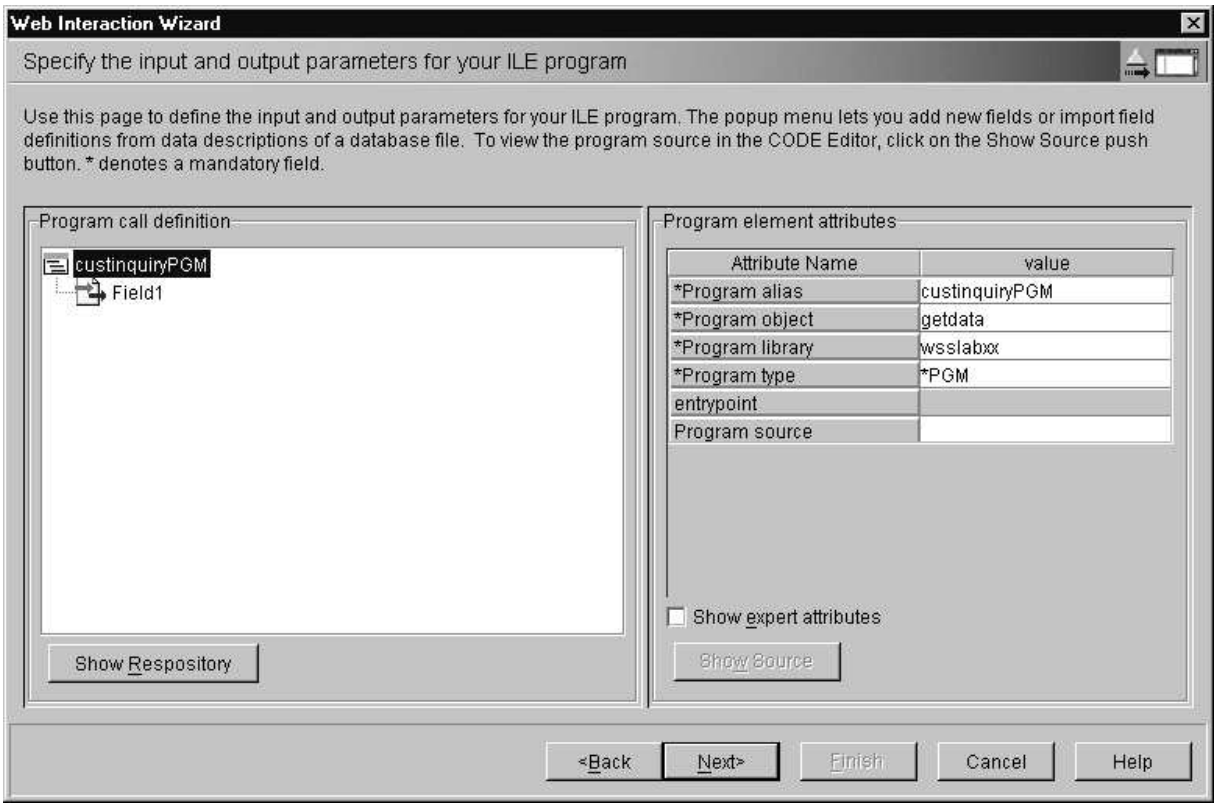
First define the program that you want to call on the AS/400

- If the program icon in the left list is not selected, **left mouse click** on it.

Now **fill in the entry fields** in the right side of the dialog

- Leave the program alias name as defined by the wizard
- Enter the **program object name** **getdata**

- **Enter** the **library name *WSSLABxx***



Now you have to make a choice for entry field **program type**

- If you feel **comfortable** in working with **Service programs** and **procedures** in Service programs **then follow the next steps**
- If you would **rather** work with a **regular program** than leave this entry field as is and **skip the next page**.

Extra steps for Service program invocation

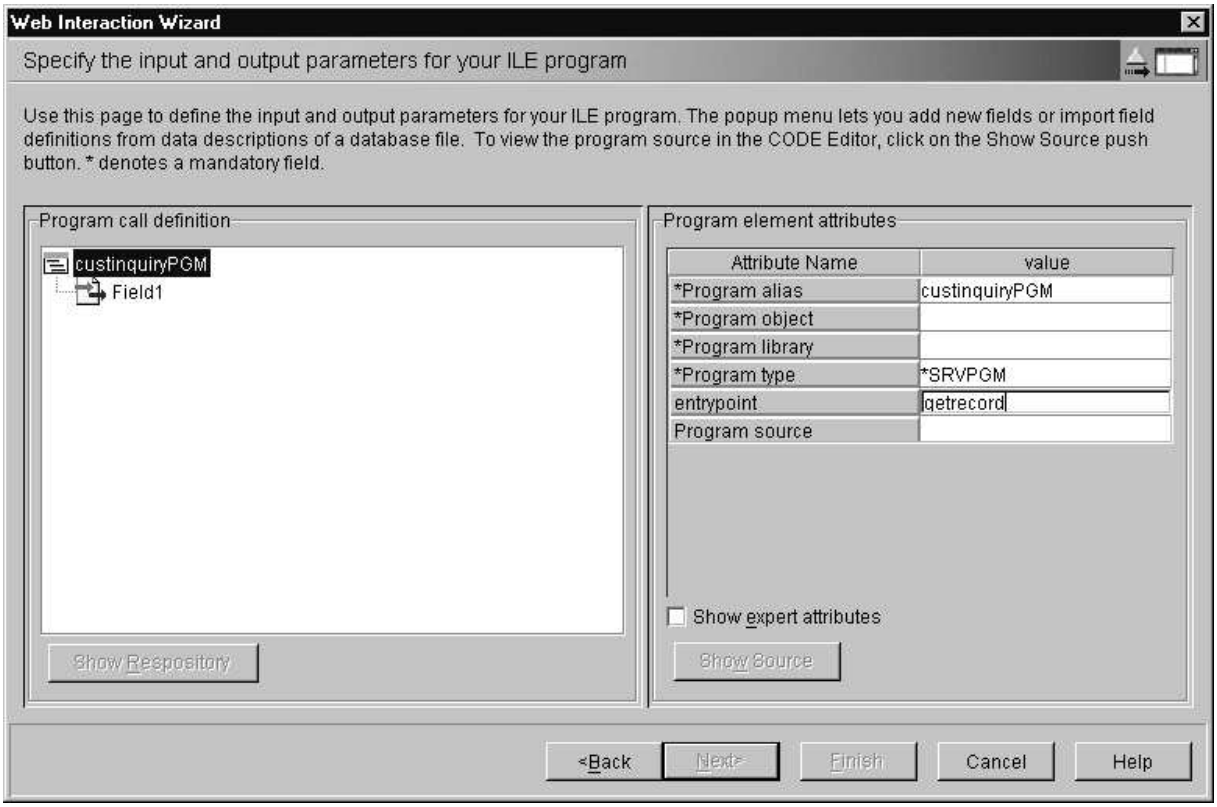
Instructions for groups that want to work with procedures in service programs,

if you want to invoke a regular *PGM from your web page just skip to the next page.

The Service program you call will have a different name, so first change the program name:

- Select the **Program object** entryfield
- Change it to ***getdatas***
- Select the **Program type** entry field
- Select ***SRVPGM** from the drop down list

- Enter **entry point** name ***getrecord*,** this is the procedure name in your service program



The special instructions for calling Service programs end here.

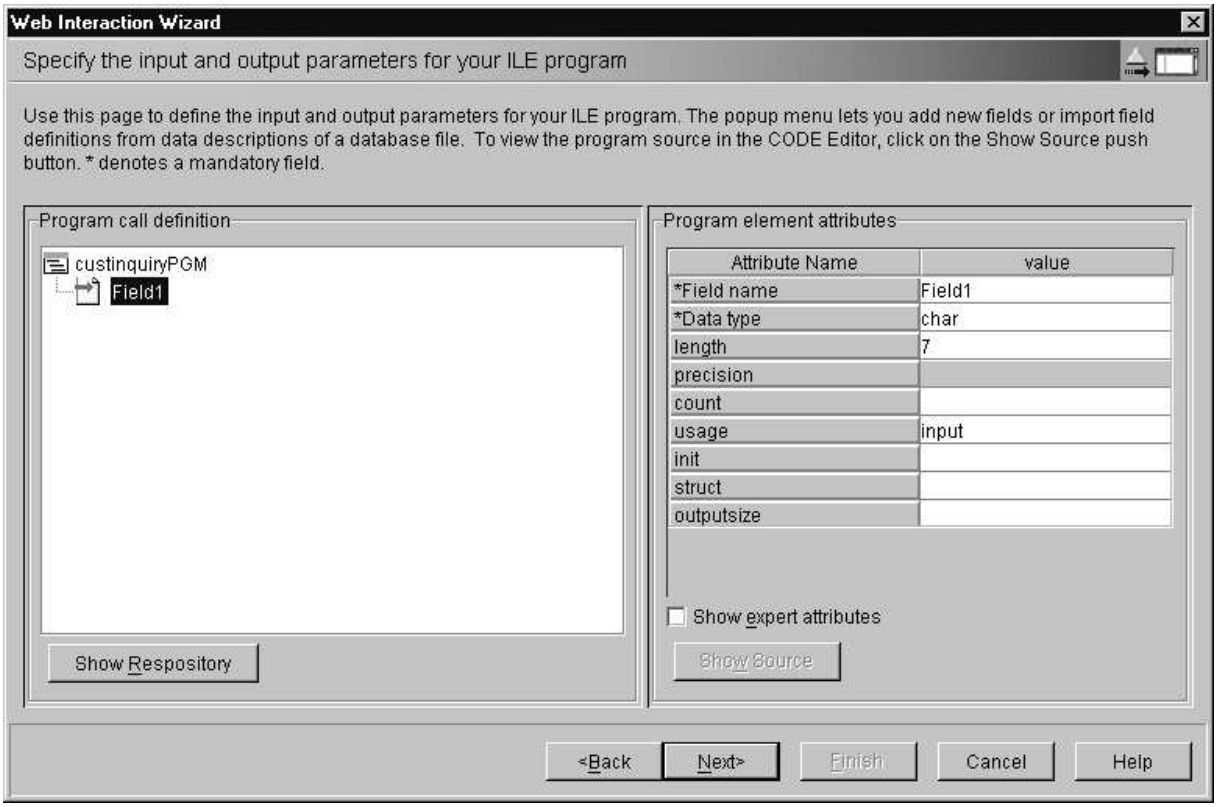
Proceed here for both *PGM and *SRVPGM invocation

Defining the parameter information

- Now click on the **Field1** icon in the list on the left of the dialog

The context entry fields for this field will appear on the **right side** of the dialog, replacing the context entryfields for the program.

You will use this **parameter** as the **Input parameter** from your Input page MyFirst.jsp to pass data to your AS/400 program



You will change the **length of the parameter** so it matches the length of the CUSTNO entry field on your page, and you will **change** the default InputOutput usage to **Input** only.

- Enter a **length of 7**

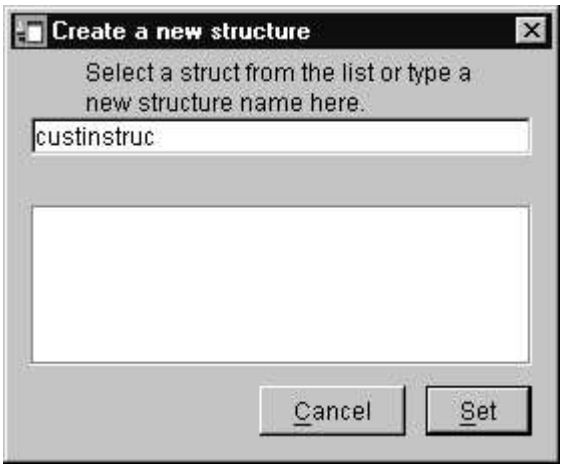
- Click on **usage**
- Select **Input** from the usage drop down list
- **Leave** the other fields as is

You just specified that the input parameter you want to pass to your program on the AS/400 is a character field with a length of 7

Now you will specify the output parameters that contain the result from the database read by your AS/400 program.

You define these fields in a **structure** and you use existing database fields to define the field in this structure.

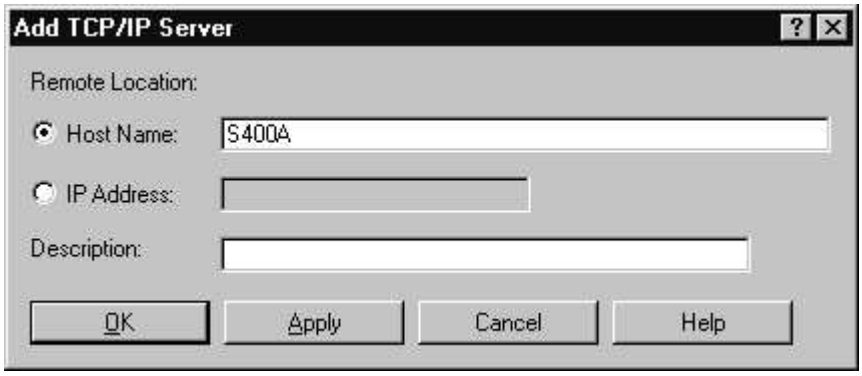
- Right mouse click on the **program icon** in the list on the left side of the dialog
- Select ***Add --> Structure*** from the pop up menu
- Enter the name of a new structure: ***custinstruc***



- Press the **Set** push button
- Right mouse click on the newly created **Structure icon** at the top of the window
- Select ***Add Fields from DB Reference*** from the pop up menu
- Select an **iSeries** from the list of servers

If the list doesn't contain servers

- Click on ***Start --> WebSphere Development Tools for AS/400 -- Communications --> Define TCP Server list***
- Press the **Add** push button
- Specify the **server name**
- **Ask the instructor for the server name**, most likely the same server you used in the Publishing Setup Wizard



- Press the **OK** push button

You will now have to go back and invoke the server list in the Interaction Wizard again

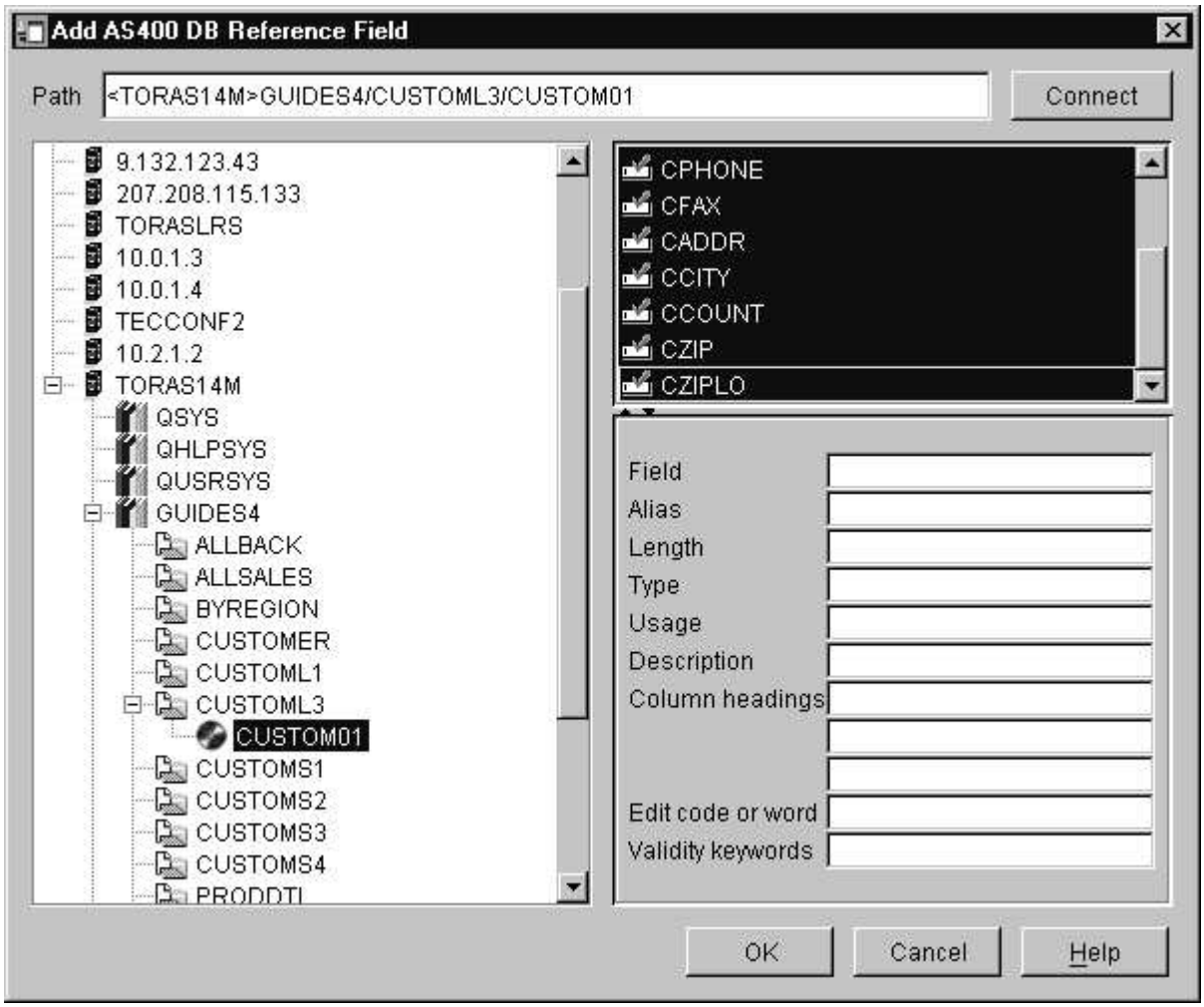
End of defining a server if none shows up in the list

- Select the **correct server** from the list by **double clicking** on its icon

> **Note**: You may be prompted for Userid and Password

- Select your **group library** from the list of libraries by **double clicking** it.
- Select file ***CustomL3***
- Expand it by **double clicking** to show the **record formats** in it

- Double click on the **record format** shown



You will see a list of fields on the right side of the dialog

- Select **all fields** by scrolling to the end of the field list, **press shift and hold shift**, **select** the **last field** with your mouse cursor, all fields should be selected
- Press the **OK** push button

You now see the newly created structure, the Web Interaction wizard always adds one default field. You see this default field on top of the structure, you don't need this field, delete it with the following steps:

- Right mouse click on the **default field**
- Select **Delete** from the pop up menu

This structure will be used to transfer data from the data base to your result web page. It will not be used to transfer data to the AS/400. By default the structure is defined as Input/output, you will use it as Output only. To change it.

- Click on the **structure icon on** top of the list
- Select the **usage** entryfield
- Select **Output** from the drop down list.

The **custinquiry** program in the list has now a structure called **Field3**, this is the default name given by the Interaction Wizard, you should rename it to **Custstruc**.

- Left mouse click on **Field3** in the left list
- Select the **Field name entry field** on the right side of the dialog
- Change the **name to** **Custstruc**

One more field is needed for this interface, this field will contain a return value to indicate whether the action on AS/400 was successful. If it was not successful the return value will contain a message number and a value for a substitution variable.

- Right mouse click on the **pgm icon** in the left list in the Web Interaction Wizard
- Select ***Add --> Field*** from the pop up menu

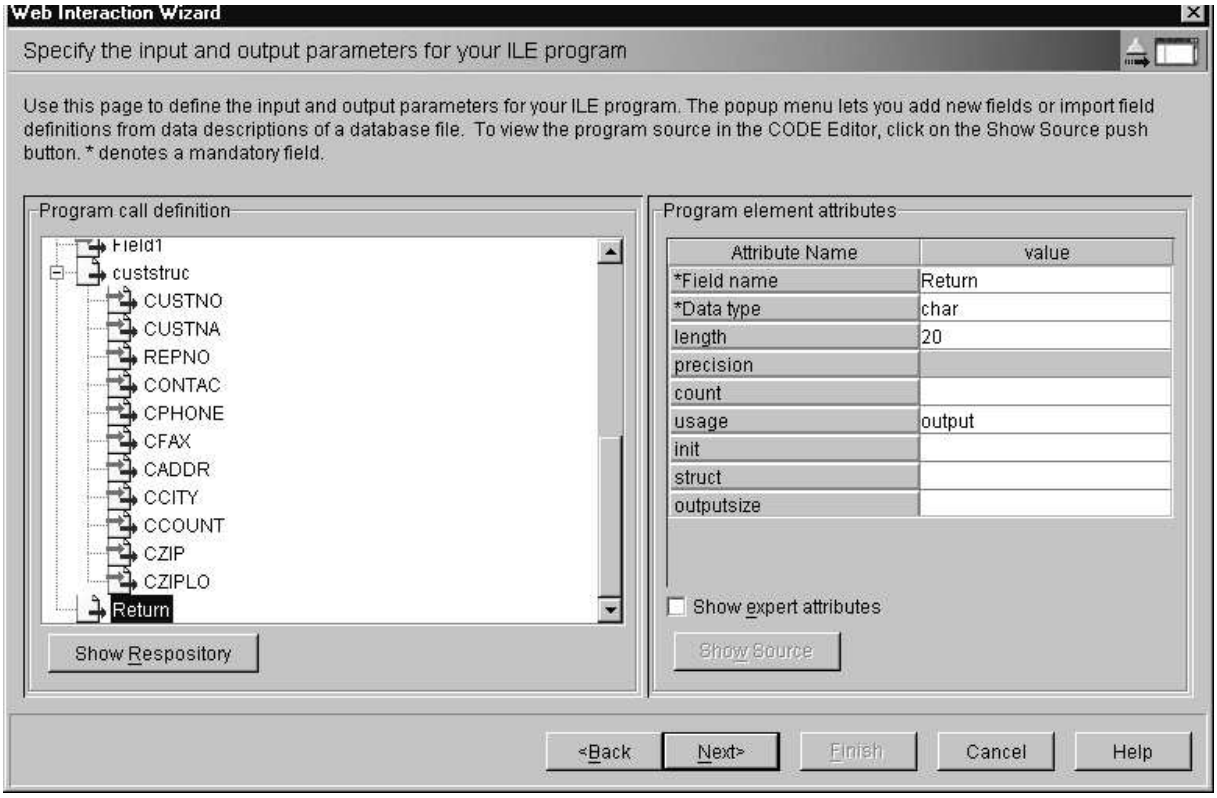
A field gets added at the end of the parameter list

- Select **this field** by left mouse clicking on it

On the right side dialog

- Change its name to **Return**
- Change the length to **20**

- Change its usage to **Output**



The interface between the **servlet** and the **AS/400 program** that **you will write** is now **defined**.

Now you have to specify which **controls on your web pages** are related to the fields in your program interface.

- Click the **Next** push button on the Wizard dialog.

The input parameter list page appears. Here you have to link the first parameter **Field1** to the **input** field **CUSTNO** on your **MyFirst.jsp** page.

- Select **Field1** from the left list (Input parameters)
- Select **CUSTNO** from the right list (Input fields)
- Press the **Link** push button

The input parameter linking is done. Proceed to the output parameter dialog .

- Press the **Next>** push button on **Web Interaction wizard** dialog

You see a list of your output parameters. Since the Interaction Wizard will generate the page layout by itself, you don't have do the linking of parameters to fields. It will be done by the wizard. You have to tell the wizard however which fields you want to include on the page to be generated. The default is to include all fields.

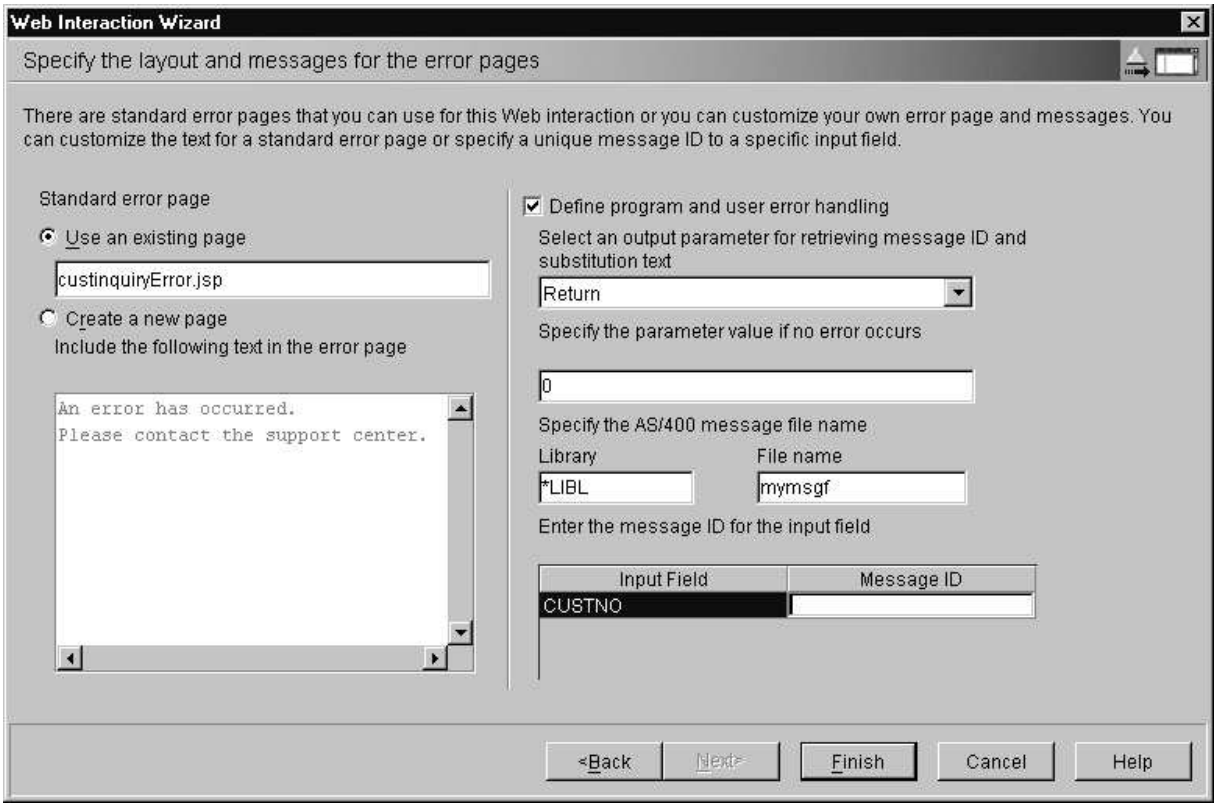
You want to deselect field **Return** from the list, the return value should not be visible to the endusers.

- Select **Return** in the Output Parameters to include list
- On the right side deselect the **Include** check box

All other fields are to be displayed and you can proceed without changing anything else on this page.

- Press the **Next**> push button

The **last page** of the Web Interaction Wizard is displayed. It allows you to specify error handling.



You will use the **RETURN** parameter to send feedback from the **RPG program** to the **SERVLET.**

- Select the **Define program and user error handling check box**
- Click on the **little arrow** in the combo box for the output parameter
- Select **RETURN** from the list

If everything goes well in the AS/400 program it will return a **0**

- Specify **0** in the entry field for the **parameter value if no error occurs**

Now you need to specify the **AS/400 message file** containing the **message text** for the message id you are returning in parameter **RETURN**

- Enter **MYMSGF** in the message file name entry field
- Leave ***LIBL** in the Library name field
- Specify **CUS0001** in the Message ID field

- Press the **Finish** push button

A confirmation screen appears.

- Press the **OK** push button
- **Press** the **Yes To all push button**, if you get a warning message that some files already exist.

A confirmation screen is being displayed,

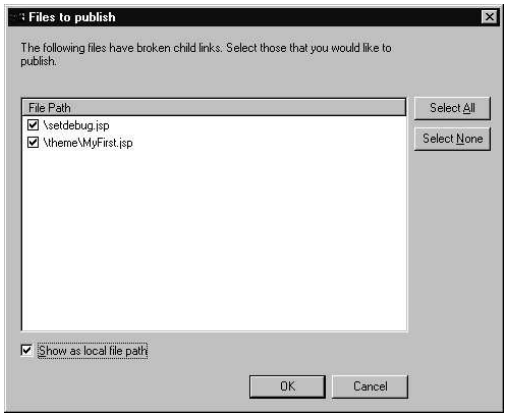
- Press the **What's next push button**

You will see a browser window being displayed with instructions for the next 2 steps

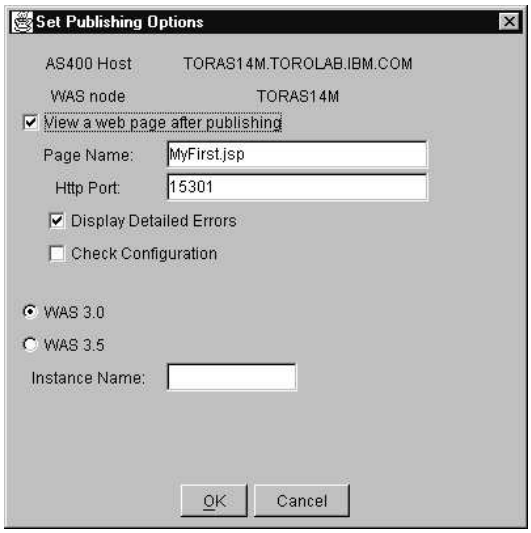
Step 1. Create publishing settings for the AS/400 host
Step 2. Publish your Web application to the AS/400 host server

Step 1 might sound familiar to you because you have done this step during the first part of this Lab. However you will have to re-publish your project since you have added files to the project when working with the Web Interaction Wizard.

- Close the **browser window**
- Right mouse click on the project icon **Labxx** in the left view of the **Studio main window.**
- Select **Publish whole project** from the pop up menu
- Press the **OK** push button on the Publishing Options dialog
- **Press OK** if you get a dialog mentioning broken Child links



- Press **OK** on the Set Publishing option dialog



Before you can test your web application you have to create the AS/400 program containing the business logic to return data to your result page.

You can write the logic with SEU if you wish to. These Lab instructions will guide you through using the **CODE/400** environment to **edit** and **compile** your AS/400 program.

Adding the AS/400 program to your Web Application

Before we start, here are some useful tips on using the CODE/400 workstation editor.

To invoke the CODE/400 editor

- Press the Windows **Start button** on the bottom of the desktop
- Select ***Program --> IBM WebSphere Development Tools for 400 --> IBM CODE/400 --> CODE editor***

That brings up the **editor window** and you are ready to write your program.

We give you 2 choices writing a regular **RPGIV** program, or an **RPGIV Service** Program,

depending on the program type you chose before in the Web Interaction Wizard,

- **proceed to the correct section** in this hand out,

They are titled

Creating the *PGM object using RPGIV
Creating the *SRVPGM using RPGIV

The next section explains some CODE editor features. If you are already familiar with CODE skip the section on CODE editor features.

Some CODE editor Features

Now that you have added the second action subroutine let's look at some of the features of the CODE editor, so you can later easily find your way around and utilize them.

- **ALT+L** will mark a line
- **ALT+U** will unmark a line
- **ALT+D** will delete a marked line
- **ALT+C** will copy a marked line
- **ALT+M** will move a marked line Press Enter to insert a new line

Token Highlighting

The CODE editor highlights the **Tokens** (specification fields) of your RPG program source, providing a separate color for each improving readability.

When you make changes to a line, the token colors updated only after you move your cursor off the line.

To see how token highlighting works:

3. Move the cursor to the **Calculation** statement you just created to **set on LR**.
4. Position the cursor to **column 7** (right next to the 'C').
5. Type an asterisk (*****).

6. Move the cursor off the line and watch what happens. The line where the '*****' was typed has become a comment line and its color changes accordingly.

7. Move the cursor back to column 7 and **remove** the '*****'. Move the cursor off that line of code and the statement is tokenized.

8. The token highlighting changes to reflect that this is a non-commented '**C**' specification.

Displaying Types of Lines

Using the CODE editor. it is possible to have only particular types of source lines displayed at a time. To do this:

3. Choose ***View→Show*** from the CODE editor menu bar. The options available on the pull down menu lists the types of line selections that can be made.

4. Choose ***Comments***. The CODE editor now contains only those RPG statements that are comments.

5. Choose ***View→ Show all***. All statements types are displayed.

In addition, the choices in the **View** pull-down can be used to include only control specifications, user subroutines, and action subroutines. The **Include** choice will allow the selection of only those lines containing a particular character string. As the CODE editor is fully-programmable, it is possible to create macros to include/exclude all types of source statements, based on specific criteria.

Syntax Checking

Syntax checking is available for RPG code, and by default will be active. The syntax of the RPG code is checked automatically when a change is made to a line, and the cursor is moved off the line.

When errors are found, they are displayed following the statement with the error.

Using Format Lines

The format line is at the top of the CODE editor window, just above the first statement. A format line is used to help keep track of the columns in a particular specification line. The content of the format line can vary to reflect the particular type

of specification being keyed such as **F** specs, **C** specs, **D** specs and so on.

To display a format line:

Make a line active by positioning the cursor to it by clicking on the line with the left mouse button, or by using the arrow keys and clicking the left mouse button. The format line gets updated as you leave a line, to refresh the format line without leaving the current statement

- Choose ***View→Refresh format line***.
- The format line changes to reflect an RPG specification, since the cursor was on a C-specification when the format line was requested.

> **Tip** *The format line can also be refreshed by pressing **Ctrl+R**. with the cursor on a statement.*

- You can move the cursor right or left with the arrow keys to go from character to character, or with the Tab key to go from field to field. An indicator on the format line moves with the cursor to show in which column the cursor is positioned.
- To select a format line for any specification you want, choose the ***View*** menu bar choice, then choose ***Select format line***. The **RPG Format Line Selection** window appears

  Now you can started and create the RPG **program** or **service program**.

•Creating a *PGM object with RPGIV

To make life easier for you we already prepared most of the **RPGIV** source for you.

To **open** this existing source follow these steps:

- Select **File --> Open** from the Edit Window menu bar
- From the **Select file dialog**, **click on the + sign** beside the name of the AS/400 server that you used to publish to in the previous exercise
- If you get a sign on dialog, use
  - Userid **WDTLABxx**
  - Password **WDTLABxx**

You will see the libraries in your jobs library list

- Expand the **WSSLABxx** library by clicking on the + sign beside its name
- Expand source file **QRPGLESRC**
- Select the **GETDATA** member from the list on the right hand side
- Press the **OK** pushbutton

The member **GETDATA** will get loaded into the edit window

Go through the source which is pretty simple

It contains an **F spec** to access file **CUSTOML3** which contains the customer data sorted by customer number

The **D spec**s defines the data structure **CSTRUC** that you pass back to your web page and the ***CUSTNOI* variable** that gets passed from the web page to this program.

As well the **Return** variable is defined as a 20 length character field.

The ***Entry** parameter list is defined as the

- **CUSTNOI** field (this is the input parameter)
- **CSTRUC** structure (this the data structure for the customer data)
- **RETURN** field to indicate no success for file access

The code you have to write here fetches a customer record by chaining to the file with the **CUSTNOI** that gets passed into the program

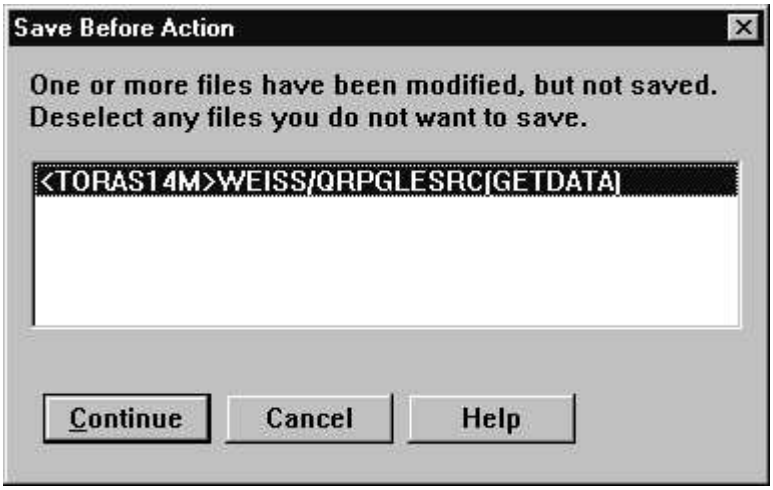
Add the code to get the customer record and check whether the Chain was successful

```
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq
C     CUSTNOI       CHAIN     CUSTOML3                              5050
C
C                   If        *IN50
C                   EVAL      RETURN='CUS0001 ' + CUSTNOI
C                   ELSE
C                   EVAL      RETURN='0'
C                   ENDIF
```

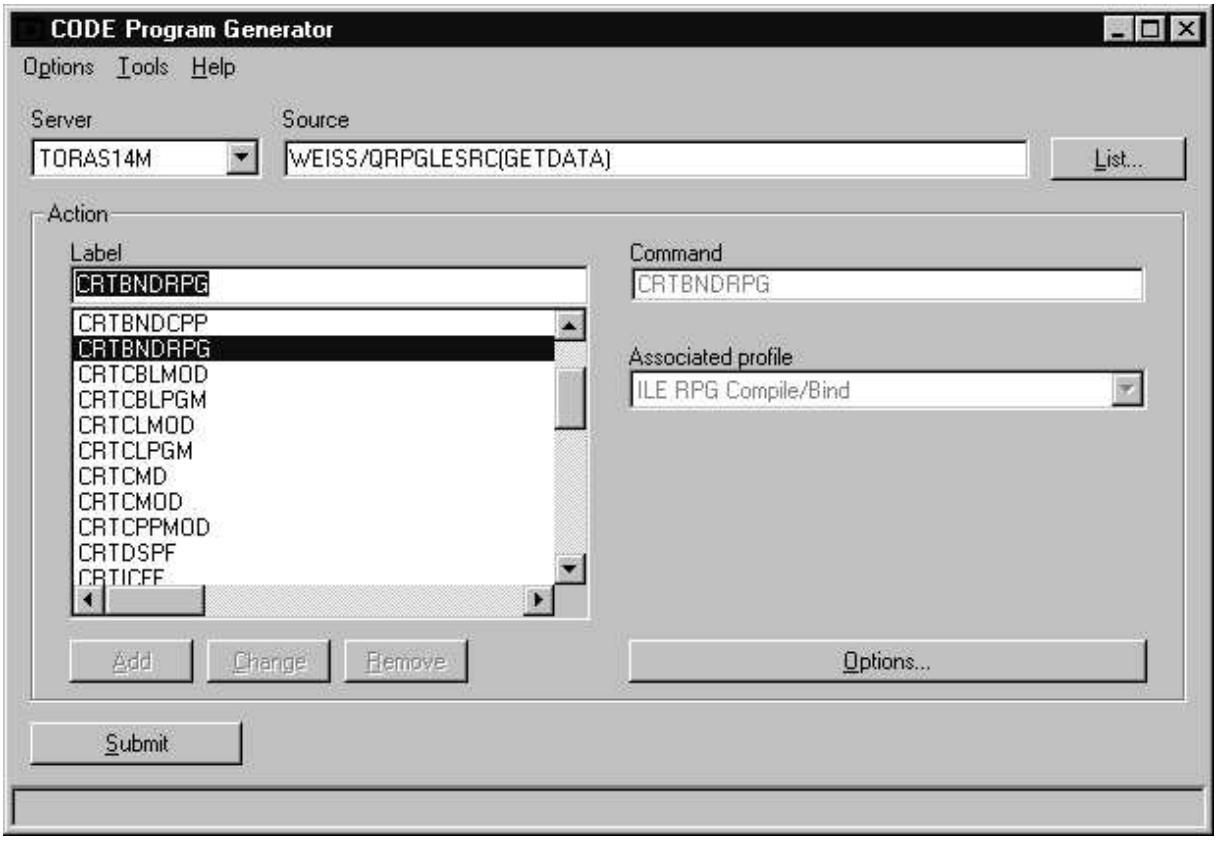
Done with coding so far, **lets Create** the program

- Select ***Actions*** **-->** ***Compile*** **-->** ***Prompt*** from the Edit window menu bar

If you haven't saved before you will get a dialog asking you whether you want to save



Press the **Continue** push button



In the **CODE Program Generator** dialog,

- Select **CRTBNDRPG** from the list of choices
- Select **the Options** --> ***Settings*** **menu option** from the CODE Program Generator dialog

- Select the **Interactive** Radio button
- Press the **Ok** push button

On the **CODE Program Generator dialog**

- Press the **Submit** push button

If you are lucky you get a message that states your **compile has been completed** and the program object has been created.

Now you are **ready to run** your web application.

Proceed to the section **Running the web application** and skip the creating a *SRVPGM part.

If you get a message that there have been **errors found** in your program,

- **go through the edit compile fix cycle**.

Edit compile fix cycle

you will see a dialog that asks you whether to proceed with the command window.

- Press the **No** pushbutton from this dialog

An error list window will appear, **check the errors by:**

- Double clicking **on the error**

That will position you to the Edit window to the statement that is wrong

Fix all errors until the error list only contains error messages with a **check mark**

- Select **Actions --> Compile -- no prompt** from the edit window menu bar

The CODE editor saves the compile options for each member, so you don't have to re-specify that you want to use the CRTBNDRPG command.

- **Continue this cycle until you get a clean compile**
- then proceed to

Running the web application

• Creating a *SRVPGM with RPGIV

To make life easier for you we already prepared most of the **RPGIV** source for you.

To open this existing source follow these steps

- Select ***File*** **-->** ***Open*** from the Edit Window menu bar
- From the **Select** file dialog, click on the **+ sign** beside the name of the **AS/400** server that you used to publish to in the previous exercise
- If you get a **sign on** dialog, use
  - Userid **WDTLABxx**
  - Password **WDTLABxx**

You will see the libraries in your jobs library list

- Expand the **WSSLABxx** library by clicking on the + sign beside its name
- Expand source file **QRPGLESRC**
- Select the **GETDATAS** member form the list on the right hand side
- Press the **Ok** push button

The member **GETDATAS** will get loaded into the edit window

- **Read** through the source which is pretty simple

It contains an **F spec** to access file **CUSTOML3** which contains the customer data sorted by customer number

The **D spec**s define the data structure **CSTRUC** that gets passed back to your web page and the **CUSTNOI** variable that gets passed from the web page to this program.

Also variable **RETURN** is defined as 20 character, to return message id and customer number in case the customer record was not found.

The prototype for procedure getrecord with the parameters defined as the

- **CUSTNOI** field (this is the input parameter)

- **CSTRUC** structure (this is the data structure for customer data)
- **RETURN** field (returns error or success)

> **Note**: The external name is case sensitive, this has to exactly match what **you specified in the Inter Action wizard**

The procedure interface defines the parameters the same way the prototype did.

Inside the procedure after the procedure interface you need to add the RPG code

You have to add the following code

```
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq
C     CUSTNOI       CHAIN     CUSTOML3                              5050
C
C                   If        not *IN50
C                   EVAL      cust1=cstruc
C                   EVAL      RETURN='0'
C                   ELSE
C                   EVAL      RETURN='CUS0001 ' + CUSTNOI
C                   ENDIF
```

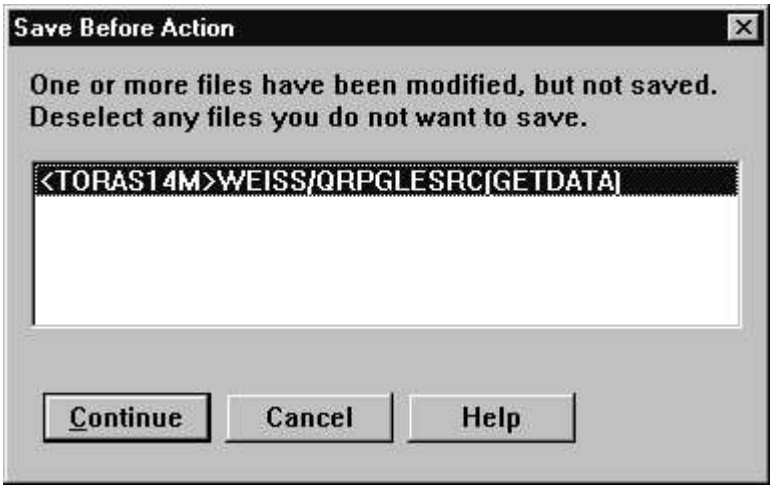
You do a CHAIN to the data base and move the data base fields to the local procedure structure. If the database access fails move the MSGID and Customer number to the RETURN variable.

Create an RPG module

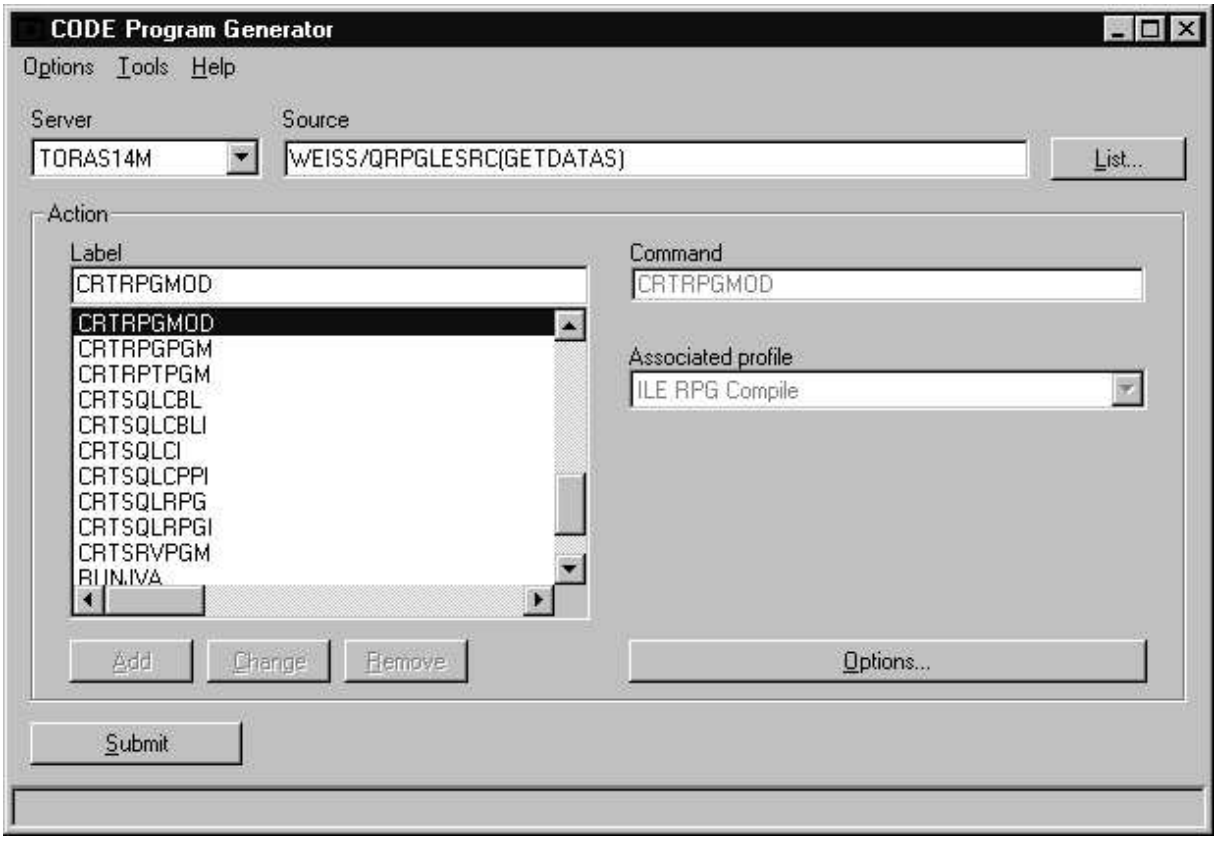
Done with coding so far, lets Create the program

- Select **Actions --> Compile --> Prompt** from the Edit window menu bar

If you haven't Saved before you will get a dialog asking you whether you want to save



- Press the **Continue** pushbutton



In the CODE Program Generator dialog,

Select **CRTRPGMOD** from the list of choices

- Select the **Options --> Settings** menu option from the CODE Program Generator **dialog menu bar**, don't use the **Options pushbutton** at this time

- Select the **Interactive Radio button**
- Press the **Ok** pushbutton

On the **CODE Program Generator dialog**

- Press the **Submit** push button

If you are **lucky** you get a message that states that your **compile has been completed** and the program object has been created.

Now you are ready to **create the service program** for your web application go the section Create service program.

If you get a message that there have been **errors found** in your program, go through the edit compile fix cycle.

Edit compile fix cycle

you will see a dialog that asks you whether to proceed with the command window.

- Press the **No** push button from this dialog

An error list window will appear, check the errors by:

- Double clicking on the error

That will position you to the Edit window to the statement that is wrong

Fix all errors until the error list only contains error messages with a check mark on the side, errors that have been fixed get a check mark in the error list

The CODE editor saves the compile options for each member, so you **don't** have to re-specify that you want to use the **CRTRPGMOD** command.

Go **back up** to **Edit Compile fix cycle**

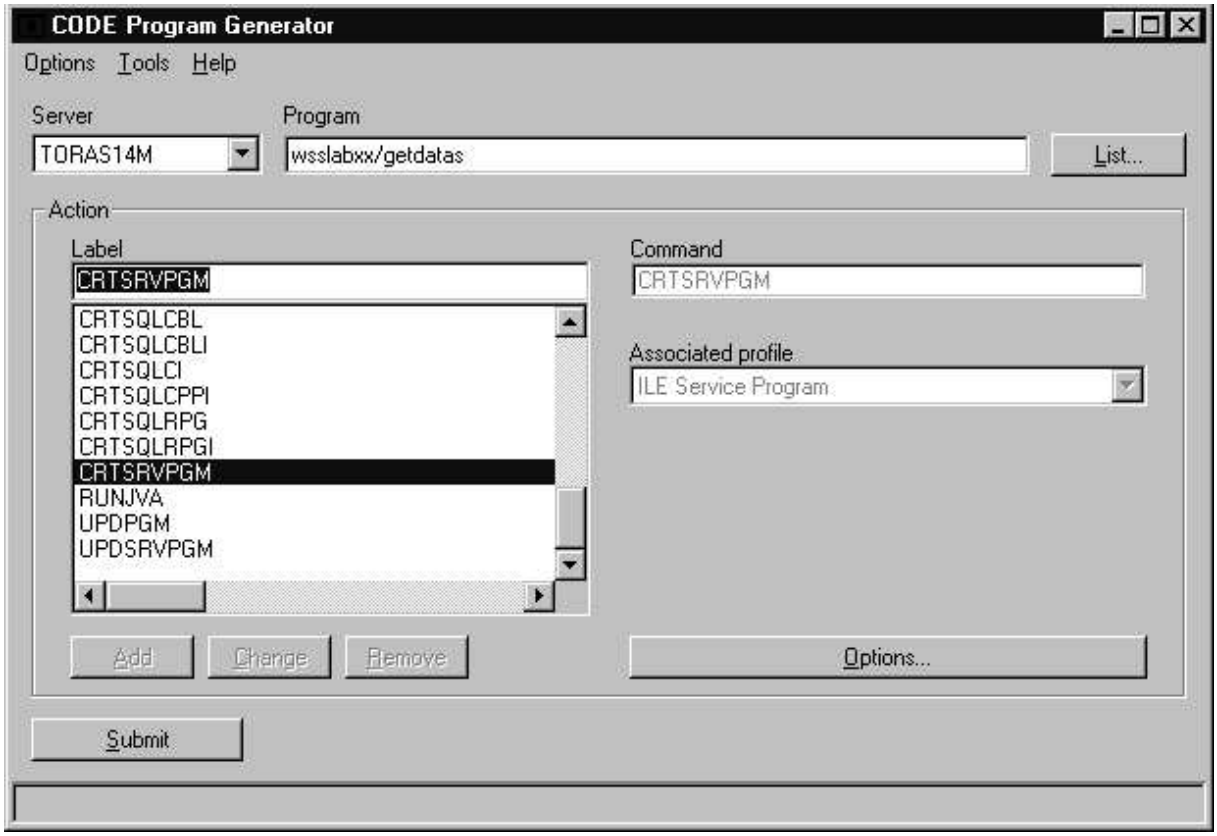
Continue this cycle until you get a clean compile

- then proceed **down to** the next step

Create a service program

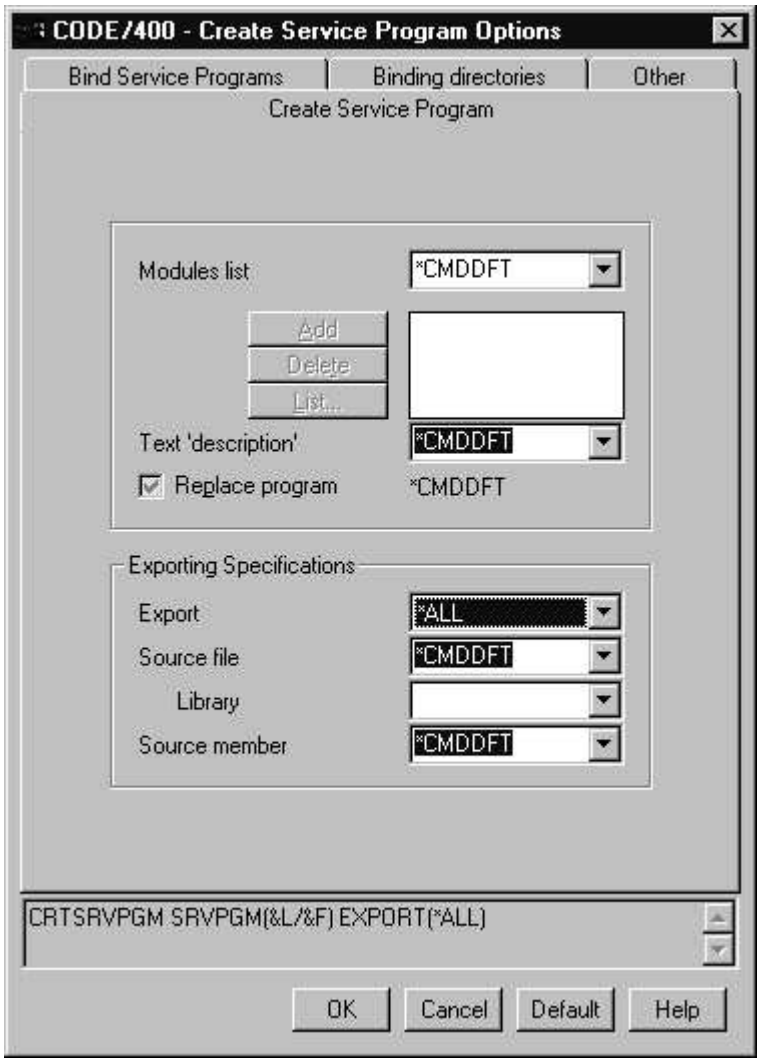
Create a Service program by

- Select **Actions --> Compile -- no prompt** from the edit window menu bar .



- Select the **CRTSRVPGM** command from the command list

- Enter **wsslabxx/getdatas** in the Entry field for the program name on top of the dialog
- Press **the options** push button (**now it is the pushbutton**)



On the CODE Service Program Options dialog, on the **Create Service Program page**

- Click on the **little arrow** in the Export **combo box**
- Select ***ALL** from the drop down list
- Press the **OK** push button
- Press the **Submit** push button on the CODE Program Generator Dialog

Hopefully you get a message that your Service program has been generated

- Press the **OK** pushbutton in the confirmation message box

Finally you are ready to run your web application

Proceed to the section **Running the web application**

Running the web application

Let's see if the application runs:

Go to your browser window and **load MyFirst.jsp**,

The first window should be displayed.

- **Type**

0010100

as the **customer number**.

- Press the **Get customer data** push button. This should bring up the second screen with the customer information filled in.
- If you type in a **wrong** customer number, an error message should appear.

WDT/400 Lab Summary

See how easy it is to create a web application by using your RPG skills. If you need to produce browser based applications the WDT/400 Wizards help you getting the job done.

This concludes the WDT/400 hands on lab. We hope you **enjoyed** the experience. Using what you have learned from this lab you can build on this experience and create your own **RPG web** applications by exploring many of the other features in **WDT/400** that were not covered here.

Remember WDT/400 includes these 5 complete products:

1. **CODE/400**
2. **VisualAge RPG**
3. **VisualAge JAVA/400**
4. **WebSphere Studio/400**
5. **WebFacing**

For more details, tips and techniques, and service and support information, be sure to visit the WDT/400 web site at

www.ibm.com/software/ad/wdt400.

Have fun with WDT/400!

