EGL Workshop for IBM i

RDi-SOA – RDi version 7.5.0 with RBD version 7.5.1.3

Featuring, Enterprise Generation Language

Rational® software

Joseph Chang, Howard Chen

Contact: changhs@tw.ibm.com

Date: 19-OCT- 2009, 20-OCT-2009, 21-OCT-2009

UNIT

RBD Workbench

Topics:

- **EGL and the Workbench terms and concepts**
- A Workbench Walk-Through

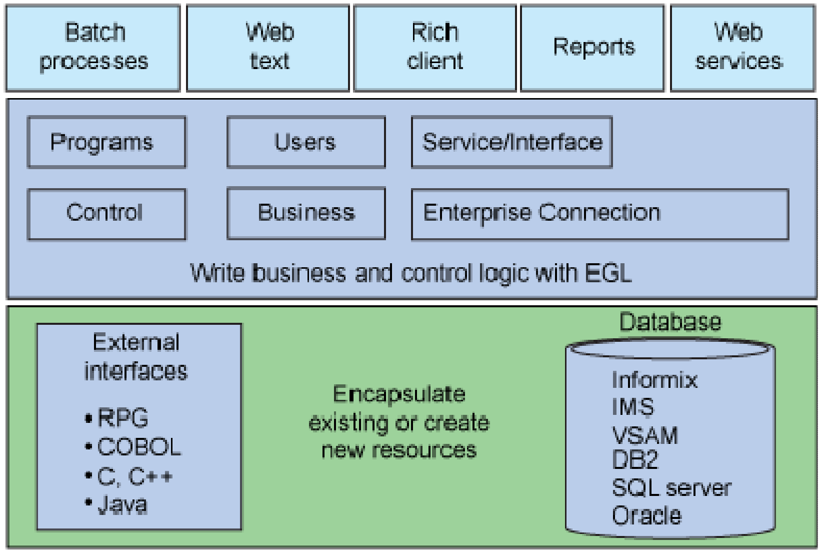
What is RDi-SOA ?

- RDi-SOA stands for Rational Developer for System i for SOA Construction

- IBM® RDi-SOA is a new software bundle that was made available in the first quarter of 2008. It combines IBM® Rational® Developer for System i and IBM® Rational® Business Developer.

- This combination offers a complete solution for modern Web development and use of Web services to rapidly extend existing RPG and COBOL applications to a Web or service-oriented architecture (SOA) environment.

  - IBM Rational Business Developer provides an Eclipse integrated development environment (IDE) for EGL. (EGL is a modern language designed to shield i5/OS application programmers from the technical complexities of Web and SOA middleware and standards.)



EGL is designed to address a full spectrum of business application requirements

What is SOA ?

SOA is defined by IBM's SOA foundation as follows:

"Service-Oriented Architecture (SOA) is an architectural style for creating an enterprise IT architecture that exploits the principles of service-orientation to achieve a tighter relationship between the business and the information systems that support the business."

SOA has the following characteristics:

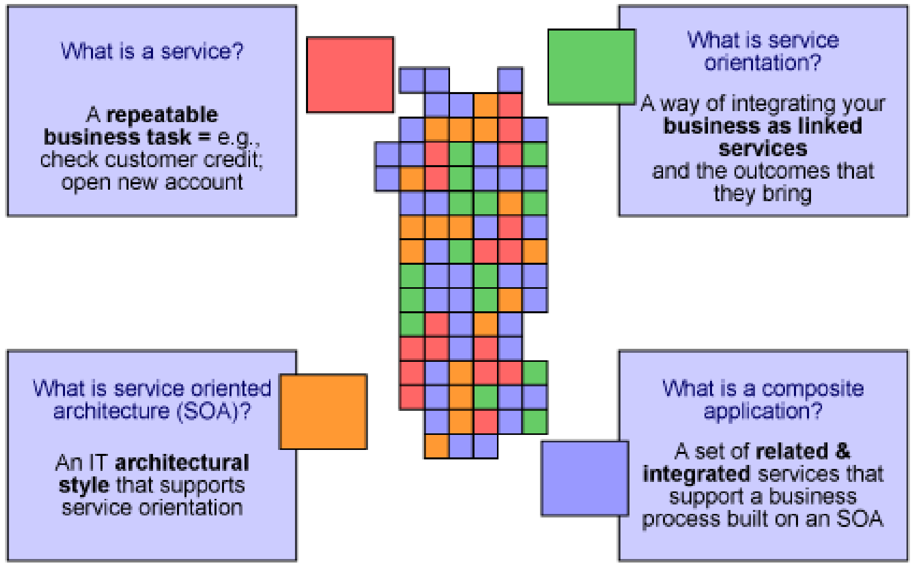
- It enhances the relationship between enterprise architecture and the business.
- It allows the building of composite applications as a set of integrated services.
- It provides flexible business processes.

Service-Oriented Architecture implies new roles in the enterprise, new ways of collaborating, new supporting frameworks and new types of software artifacts in an evolutionary (as opposed to a "revolutionary") way.

Definitions of a SOA

What is a service?

- A service is a repeatable logical manifestation of a business task.
- A service is about a part of a business process -- not about software or IT.


What is a service?
A **repeatable business task** = e.g., check customer credit; open new account
What is service orientation?
A way of integrating your **business as linked services** and the outcomes that they bring
What is service oriented architecture (SOA)?
An IT **architectural style** that supports service orientation
What is a composite application?
A set of **related & integrated** services that support a business process built on an SOA

What is Service Orientation?

- Service orientation is a way of integrating a business as a group of linked services.
- Service orientation is not about technology; It is about a new way of looking at business and how it operates.

What is SOA?

- SOA is an architectural style that supports the service orientation.
- SOA is an enterprise-scale IT architecture for linking resources on demand.

- IBM, Software Group

What is SOA About ?

“SOA is about flexible business processes. It supports the flex-pon-sive company by merging technology, business insight, and thought leadership to create an environment in which innovation can thrive.”*

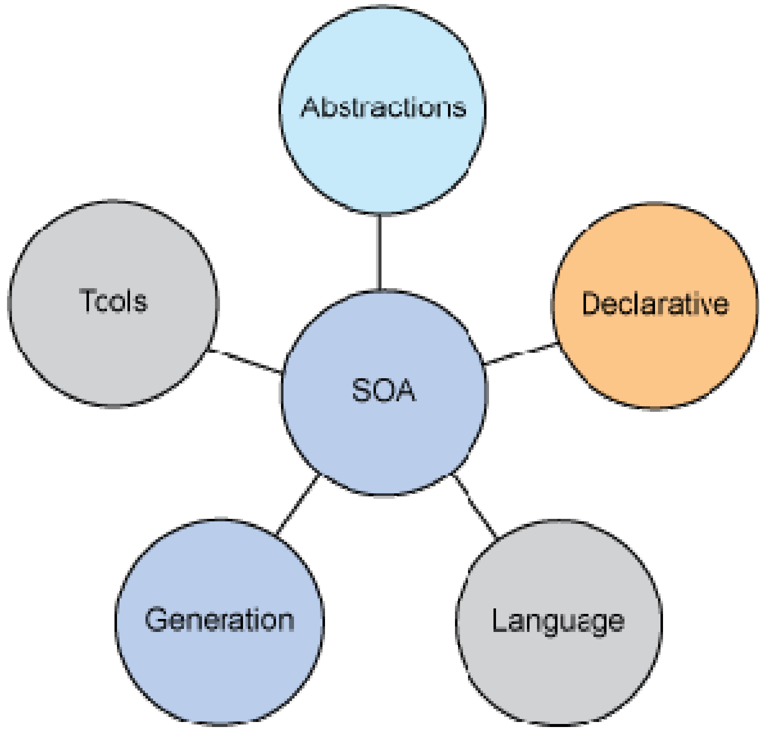
The top 10 don'ts for your flex-pon-sive* journey are:

- Don't expect maximum business without SOA.
- Don't just do technology; it's a transformation of the way you do business.
- Don't throw everything out.
- Don't bite off big projects all at once.
- Don't forget to set expectations.
- Don't expect to do this without a culture modification through governance.
- Don't forget the right skills.
- Don't expect flexibility without open standards.
- Don't do this alone - leverage partners with experience.
- Don't do it without a strong plan because the first step is the most important.

- Sandy Carter, Vice President of SOA and WebSphere Strategy, IBM Corporation

What makes EGL powerful?

EGL supports a variety of SOA-related capabilities that help business-oriented developers become extremely productive in a short time.



RDi-SOA and EGL

RDi-SOA and EGL provides several ways for developers to increase their productivity for i5/OS application development:

- **Abstraction.** EGL provides concise and powerful notations that help to eliminate the tight coupling. It reduces the amount of coding required to interface systems and middleware. This abstraction simplifies and speeds up development work.
- **Declarative programming.** EGL includes a certain level of declarative specifications to help reduce repetitive and error-prone coding. For example, validation rules associated with a data item trigger validation to be run whenever the item is used in a Web page or a 5250 screen.
- **Language.** EGL is a comprehensive but easy-to-learn language. It is modern, modular, and readable. It has a rich library of built-in functions to boost your productivity for commonly required operations, such as date and time math, string manipulation, and so forth. In addition, the language is extensible and offers full interoperability with other languages, including EGL interfaces to native Java, as well as seamless invocation of RPG or COBOL programs or any ILE procedures.
- **Generation**. Though simplified, the EGL development technology must still guarantee optimal deployment to the runtime platforms to take advantage of their Qualities of Service (QoS) and to allow native management and monitoring of the systems in operation. This is accomplished through a code generation engine available as part of the Rational Business Developer Extension that transforms the EGL specification into native Java or COBOL source, and creates any other required deployment artifacts.
- **Tools.** RDi-SOA contains a rich set of Eclipse-based capabilities, including EGL source-level debugging, powerful smart editing, visual construction, graphical navigation, and automated generation of Create/Read/Update/Delete (CRUD) applications from Unified Modeling Language (UML) models or from data schemas.
- **SOA.** EGL is designed to facilitate services development and deployment. A simplified SOA development paradigm is built into the language itself and complemented with the tools. You can create services without the need to know Web service protocols and standards, such as Web Services Description Language (WSDL), Simple Object Access Protocol (SOAP), or Extensible Markup Language (XML).

- Linda Cole, Manager of IBM Rational Business Ecosystem

The Value of RDi-SOA and EGL

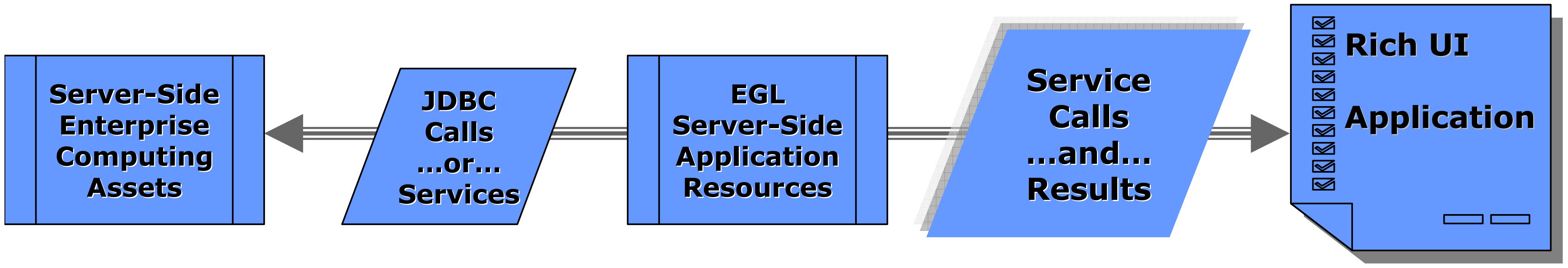
Organizations that need to deliver modern, powerful Web and SOA solutions can enjoy the following benefits:

- Increased programmer productivity
  - 300% to 1000+ percent
- A newly empower class of developers delivering modern solutions and SOA
- Increased flexibility and responsiveness by eliminating skills silos
  - EGL developer can be RPG + Java + SOA + JSF/RUI developers
- Reduced training costs
  - Easy to learn
- Reduced errors and deliver higher quality
  - Generates Java, JavaScript and Web Services
- Simplified legacy integration
  - Call RPG programs
  - Encapsulate RPG programs into Web Services
- Reduced development and maintenance costs via a flexible choice of deployment options, including:
  - All major IBM Operating Systems and runtimes
  - Cross platform solutions

SOA and Web 2.0 Rich UI

- Web 2.0 Rich UI makes extensive use of **services, and SOA** – Service Oriented Architecture, which is a way to modularize and deploy code so that it can be consumed anywhere in the world using any language.

- There are two types of Web Service calls used by Web 2.0 Rich UI
  1. **RESTful service calls** – A call made through the HTTP service-interface. Once the call is made, a result is passed back to the requestor in XML or JSON format.
  2. **SOAP service calls** - A type of service call that is more popular in enterprise. It requires the exchange of XML messages between the client and host system.


Server-Side Enterprise Computing Assets
JDBC Calls ...or... Services
EGL Server-Side Application Resources
Service Calls ...and... Results
Rich UI Application

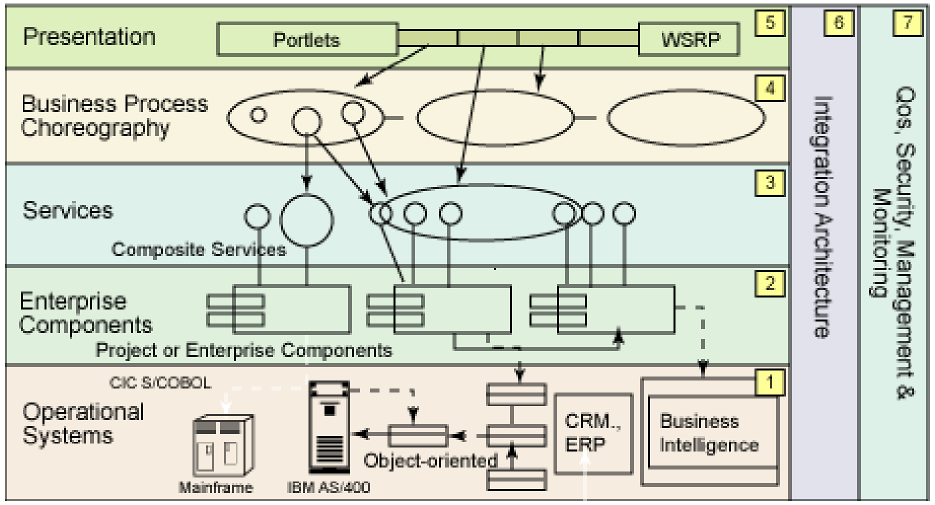
- By utilizing web services you build modular, scalable systems.

SOA Challenge

- SOA helps create greater alignment between IT and line of business while generating more flexibility

- According to a survey of six hundred senior executives around the world, the #1 barrier that companies are seeing to adopting SOA is shortage of skills.

The Layers Of A SOA



- Successful SOA adoption is done incrementally stressing the importance of starting small, and scaling appropriately.

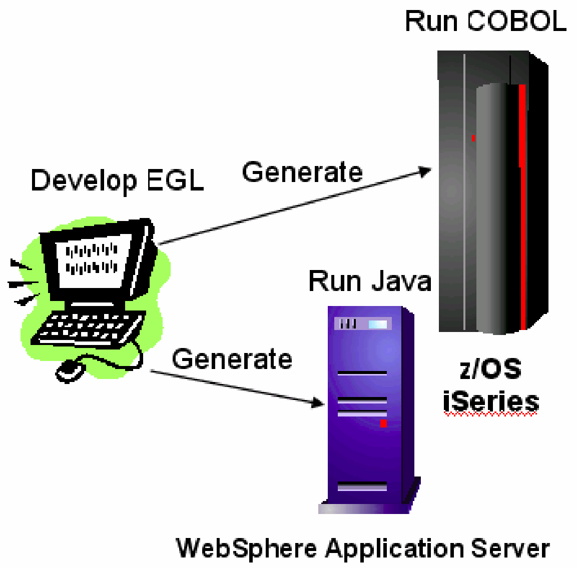
- Legacy applications are frequently at the core of your IT environment. But many times, these essential applications are isolated and inaccessible to common skill sets. Without the right skills and tools, it can be difficult to integrate these core investments with the rest of your IT environment.

What Are Your Skills ?

- IBM i Skills
  - ❑RPG ILE
  - ❑RPG/Free
  - ❑Embedded SQL
  - ❑Synon/2E
  - ❑RDi
- Web Development Skills
  - ❑Java Server Faces
  - ❑Java
  - ❑JavaScript
  - ❑AJAX
  - ❑SOA, WSDL, SOAP
  - ❑REST, XML

What is EGL?

- EGL is **Enterprise Generation Language**
- High-level programming language for developing leading-edge business applications
- Independent of implementation
  - Hides technology complexities
  - Presentation and persistence layer API fully generated
- Enables Rapid Application Development
- Simplifies development of Services (SOA)
- Run-time code generated for appropriate platform
  - Java for Windows, Linux, System i, and so forth
  - COBOL for System z and System i
- Development and Generating Tools included with RBD
- Migration path for CSP, Visual Age Generator, and Informix 4GL customers
- Integrates with leading-edge software technologies
  - Eclipse, JSF, Application Servers, etc.



Three Main Benefits of the EGL Language

PORTABLE

- Java, SOA, COBOL, System Z/I/P, Browser, ...
- Optimal native generation to any platform
- Easy inter-operability with legacy

SIMPLE

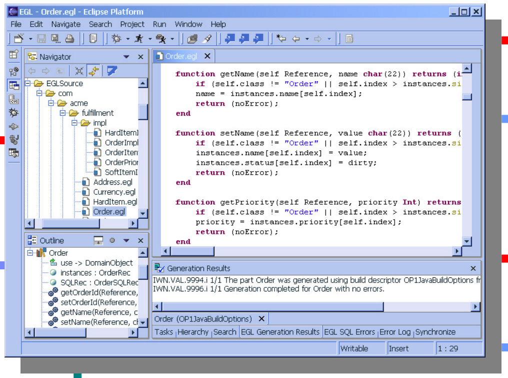
- High productivity with equal flexibility
- Language simplicity and robustness
- Immediately useable by business developers
- Scalable software architectures

ABSTRACT

- Effectively hide technical complexity
- Use declarative approach to everything
- Support emerging standards

EGL Portability - RBD/EGL Platform Coverage

EGL Business Logic Common Repository


EGL - Order.egl - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Navigator
Outline
Order.egl

```
function getName(self Reference, name char(22)) returns (i
    if (self.class != "Order" || self.index > instances.si
    name = instances.name[self.index];
    return (noError);
end

function setName(self Reference, value char(22)) returns (
    if (self.class != "Order" || self.index > instances.si
    instances.name[self.index] = value;
    instances.status[self.index] = dirty;
    return (noError);
end

function getPriority(self Reference, priority Int) returns
    if (self.class != "Order" || self.index > instances.si
    priority = instances.priority[self.index];
    return (noError);
end
```

Generation Results

```
IWN.VAL.9994.i 1/1 The part Order was generated using build descriptor OP1JavaBuildOptions fr
IWN.VAL.9996.i 1/1 Generation completed for Order with no errors.
```

Order (OP1JavaBuildOptions)

Writable Insert 1 : 29

Java / COBOL →

System z

- WebSphere
- USS
- Linux
- Batch
- CICS
- IMS

Java / COBOL →

Windows, Linux, Unix – System P

- WebSphere
- Other J2EE Application Servers
- Native i5OS
- Native i5OS

Java / COBOL →

System i

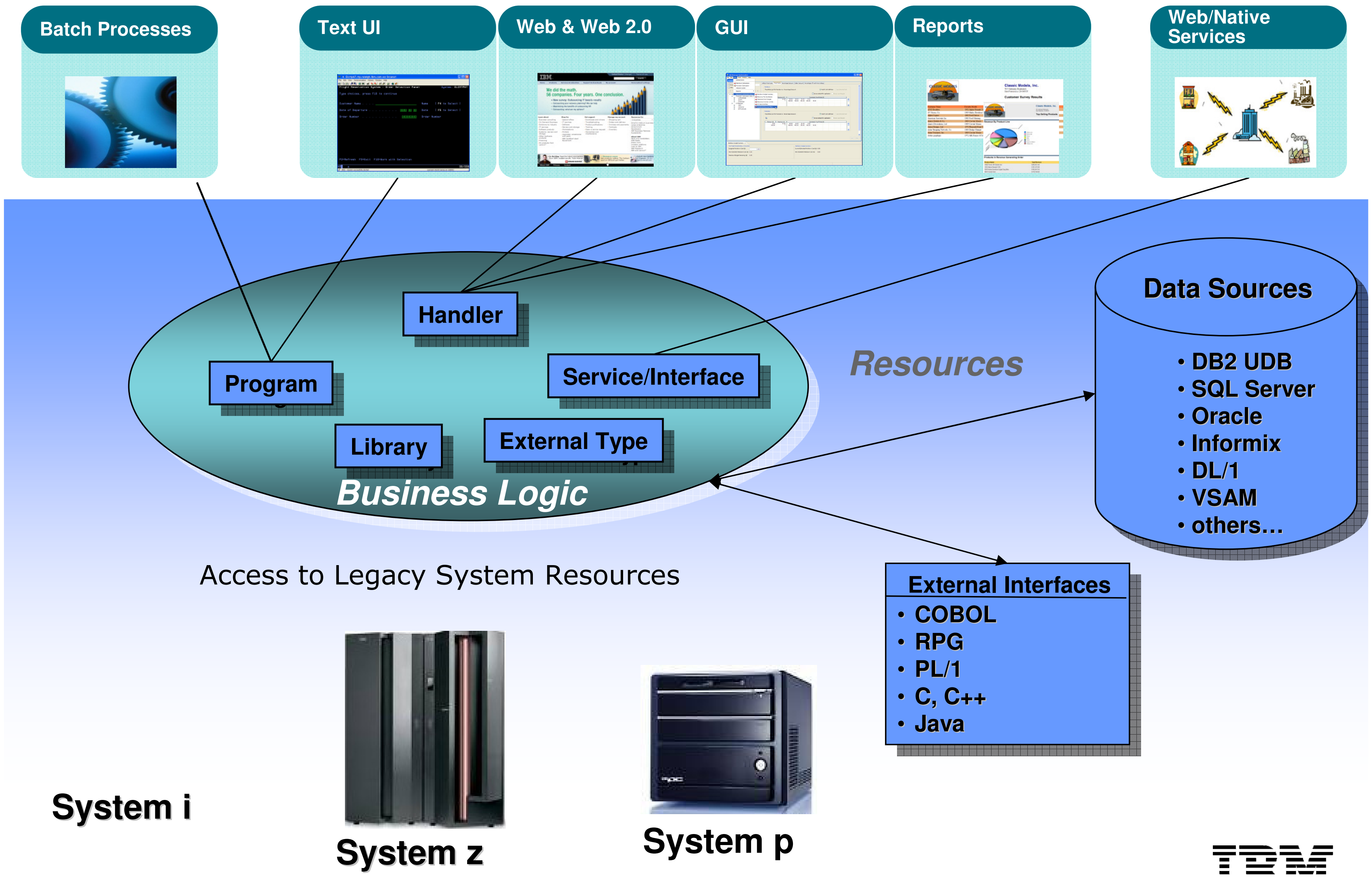
- WebSphere
- Other J2EE Application Servers
- Native i5OS
- Native i5OS

JavaScript Web 2.0 →

Browser

- IE *
- Firefox *
- Safari *

Portability – RBD/EGL Technology Support



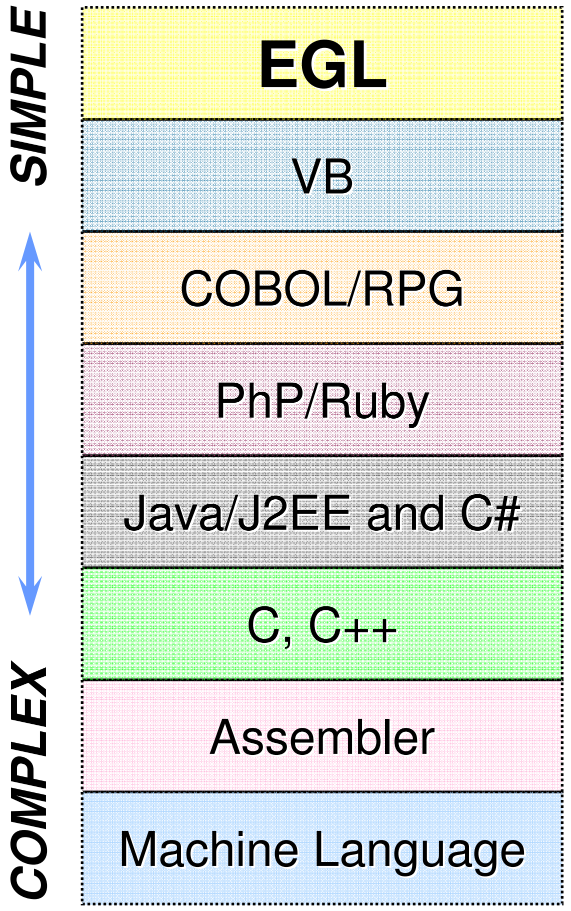
RBD/EGL – Higher Level Abstractions

- Extend the natural evolution of programming language as an abstraction of business expression

"Re-training COBOL developers to Java/J2EE costs over $50K each, and only 12% may actually succeed" – Gartner Group

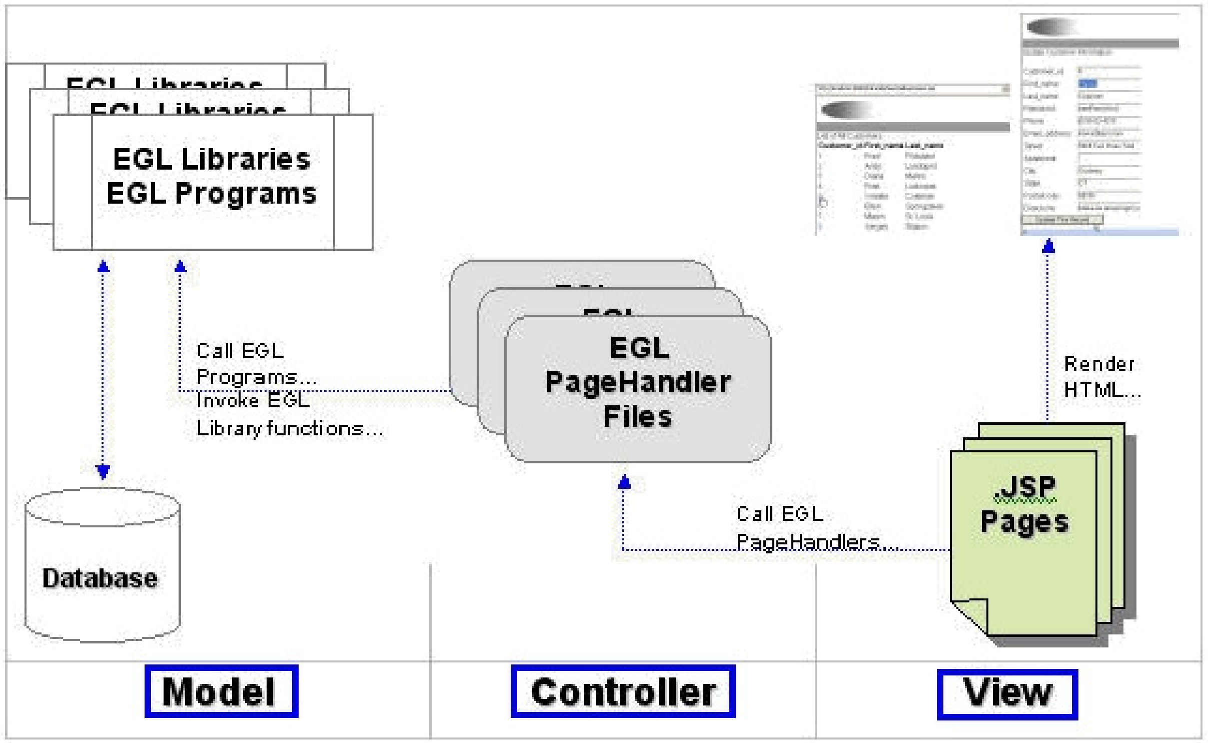
"The task force had an initial meeting and identified challenges to teaching Java, based on the literature and our experiences." – ACM Education Board Java Task Force http://www.sigcse.org/topics/javataskforce/

An insurance company spent approximately $250,000 to train 12 RPG developers. One out of 12 succeeded. This person had a MS in Computer Science and 8 years of C programming experience. – Bob Cancilla



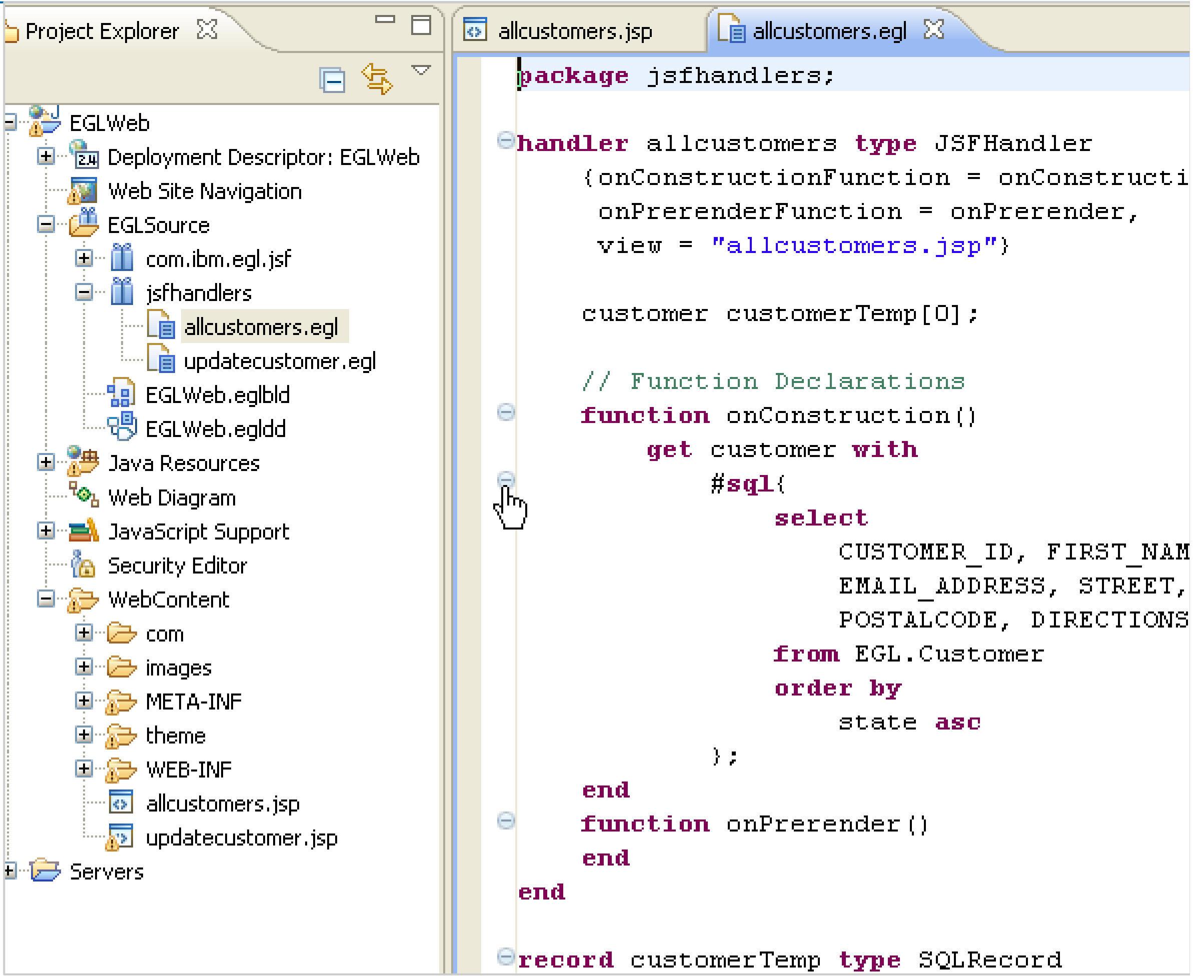
EGL and the Model-View-Controller Framework

- The model-view-controller framework (referred to as MVC or "model 2") has many benefits and is often considered a best practice for developing Web applications.
- Use EGL to develop applications that are divided into these three functional layers:



What is Rational Business Developer (RBD)?

- One of several products that allows you to do develop using the EGL language. Other products include:
  - RDI-SOA
  - RDz
- Member of the Rational Software Development Platform (SDP)
- Support for wide range of platform, middleware, and technologies
- Rich Eclipse-based tooling



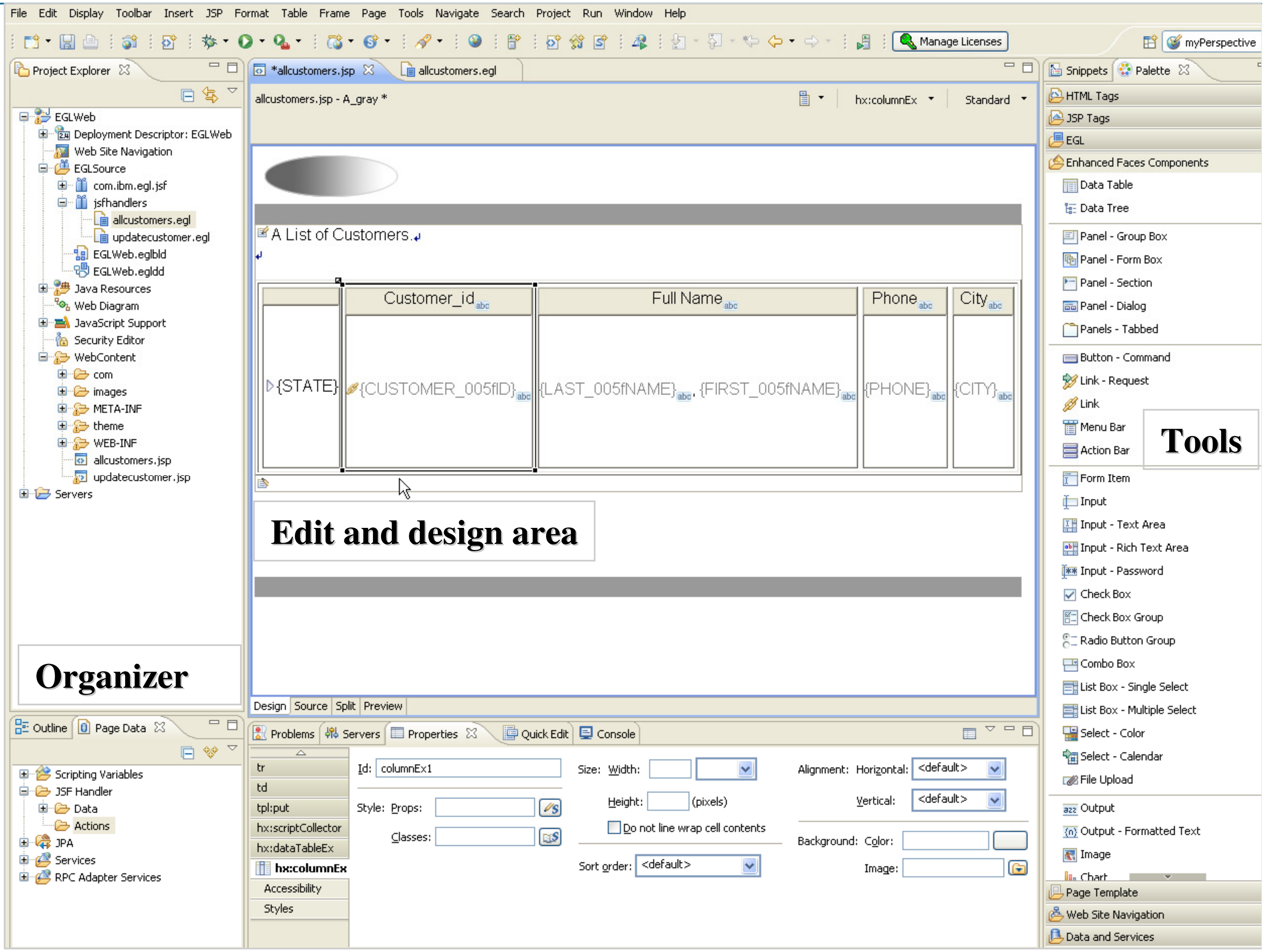
```
package jsfhandlers;

handler allcustomers type JSFHandler
    {onConstructionFunction = onConstructi
     onPrerenderFunction = onPrerender,
     view = "allcustomers.jsp"}

    customer customerTemp[0];

    // Function Declarations
    function onConstruction()
        get customer with
            #sql{
                select
                    CUSTOMER_ID, FIRST_NAM
                    EMAIL_ADDRESS, STREET,
                    POSTALCODE, DIRECTIONS
                from EGL.Customer
                order by
                    state asc
            };
    end
    function onPrerender()
    end
end

record customerTemp type SQLRecord
```

What is the RBD Workbench?

The RBD Workbench is the software that enables you to create, test, modify, run and deploy your EGL applications

It organizes and maintains your software development resources

It provides access to tools like editors, design tools, compilers and folders for managing your application's contents



Id: columnEx1

Style: Props:

Classes:

Size: Width:

Height: (pixels)

Do not line wrap cell contents

Sort order: <default>

Alignment: Horizontal: <default>

Vertical: <default>

Background: Color:

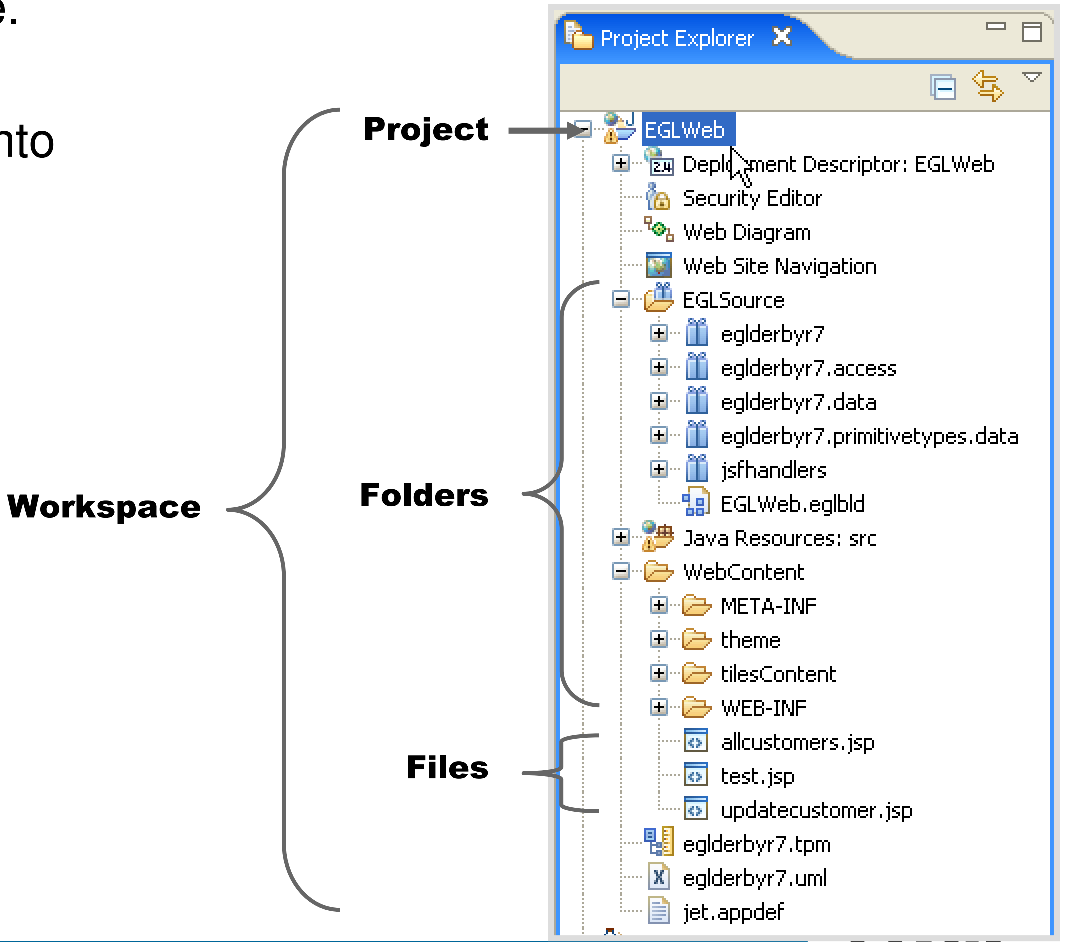
Image:

How do I develop with the RBD Workbench?

- You will work with a variety of resources: EGL code, Web pages, Graphic images, Data files. The Workbench allows most of the resources associated with a project to be stored in a workspace – which is the highest-level folder that contains everything you can access during a development session.

- Your Workbench session begins with the opening of a Workspace. Everything you have access to is inside this Workspace.

- Workspace resources are organized into
  - Project(s)
    - Folders
      - Files



EGLWeb
Deployment Descriptor: EGLWeb
Security Editor
Web Diagram
Web Site Navigation
EGLSource
eglderbyr7
eglderbyr7.access
eglderbyr7.data
eglderbyr7.primitivetypes.data
jsfhandlers
EGLWeb.eglbld
Java Resources: src
WebContent
META-INF
theme
tilesContent
WEB-INF
allcustomers.jsp
test.jsp
updatecustomer.jsp
eglderbyr7.tpm
eglderbyr7.uml
jet.appdef

What is in a Project?

- **Workspace projects** organize and manage related application resources. They can be designed along lines of:
  - Batch – or – Online applications
  - Different business applications (Accounts Payable, Inventory, Claims, Part Assembly, etc.)
  - Common (shared) projects consisting of data and record definitions that can be reused

- Projects also contain **configuration data**, such as *build files*** and **generation options** for EGL Java and/or COBOL generation

- Projects may be further divided into **folders**

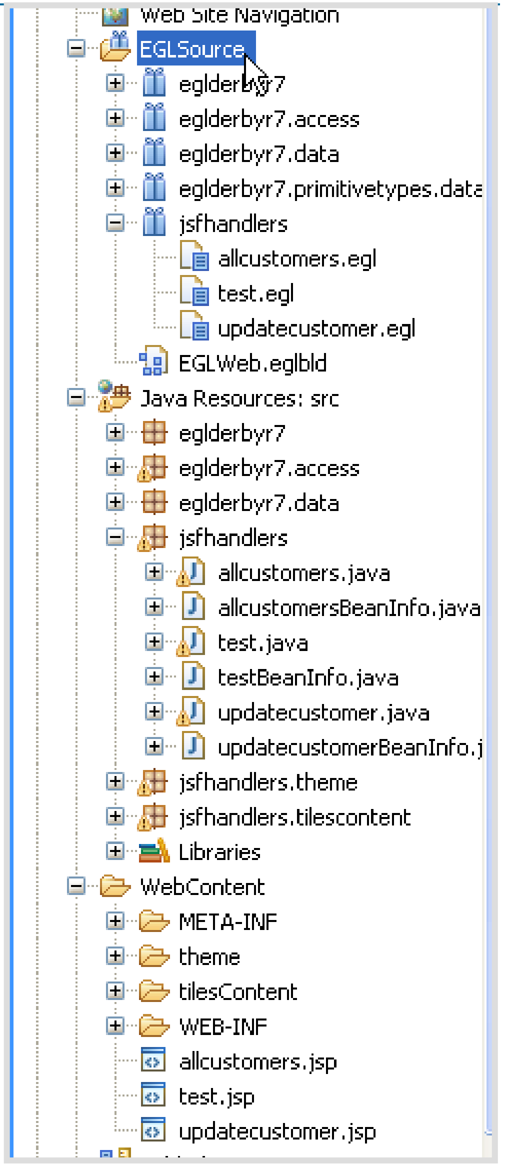
- EGL "Web" projects contain the following high-level folders:

```
\EGLSource\
\Java Resources: src\
\WebContent\
```

EGL Source Files

Java Source Files

Web Pages and Web App Resource Folders



```
Web Site Navigation
EGLSource
  egldrbyr7
  eglderbyr7.access
  eglderbyr7.data
  eglderbyr7.primitivetypes.data
  jsfhandlers
    allcustomers.egl
    test.egl
    updatecustomer.egl
  EGLWeb.eglbld
Java Resources: src
  eglderbyr7
  eglderbyr7.access
  eglderbyr7.data
  jsfhandlers
    allcustomers.java
    allcustomersBeanInfo.java
    test.java
    testBeanInfo.java
    updatecustomer.java
    updatecustomerBeanInfo.j
  jsfhandlers.theme
  jsfhandlers.tilescontent
  Libraries
WebContent
  META-INF
  theme
  tilesContent
  WEB-INF
  allcustomers.jsp
  test.jsp
  updatecustomer.jsp
```

📖 - See slide **NOTES** and the Help topic: Contents of an EGL Application

What is in \EGLSource\

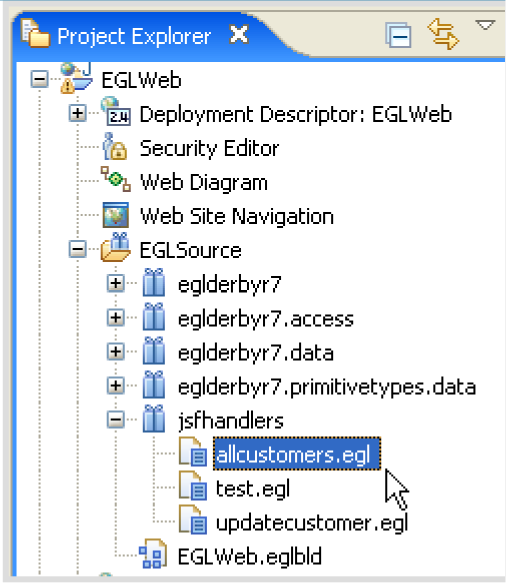
`\EGLSource\` is the default folder that is the highest level folder under which all of your EGL resources are organized.

- These EGL resources include EGL:
  - Packages
  - Files
    - Programs
    - Libraries
    - Services
    - JSFHandlers
    - EGL build-files
    - …

- Typically you will create sub-folders under **`\EGLSource\`** to manage your EGL resources

- We will define what is in all these EGL resources a bit later in this course

- For now, it is enough to know that **all EGL files end with the extension: .egl**

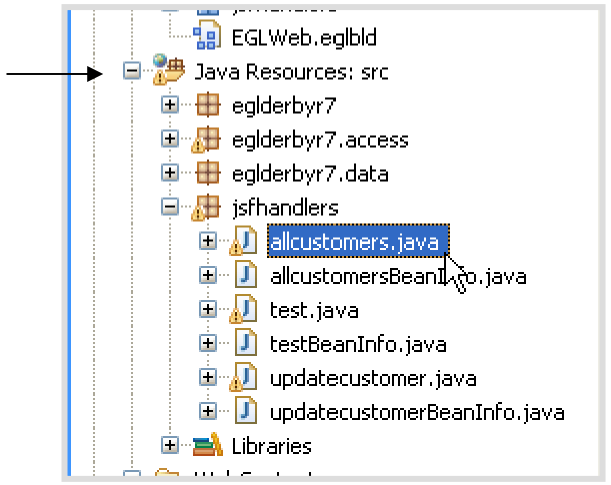

```
Project Explorer
EGLWeb
  Deployment Descriptor: EGLWeb
  Security Editor
  Web Diagram
  Web Site Navigation
  EGLSource
    eglderbyr7
    eglderbyr7.access
    eglderbyr7.data
    eglderbyr7.primitivetypes.data
    jsfhandlers
      allcustomers.egl
      test.egl
      updatecustomer.egl
    EGLWeb.eglbld
```

📖 - See the Help topic: EGL Projects, Packages and Files

What is in Java Resources: src

\src\ is the default highest level folder that contains all of the generated Java, for each and every successfully compiled (or in EGL terms, "generated") EGL resource

- You will rarely if ever need to actually open or even view the contents of **\src\**
- But the folder structure of **\src\** will match **\EGLSource\** one-for-one
- Your generated java files will end with a **.java** extension



EGLWeb.eglbld
Java Resources: src
eglderbyr7
eglderbyr7.access
eglderbyr7.data
jsfhandlers
allcustomers.java
allcustomersBeanInfo.java
test.java
testBeanInfo.java
updatecustomer.java
updatecustomerBeanInfo.java
Libraries

What is in \WebContent\

WebContent\ is the default highest-level folder under which all of your web resources will be organized, including:

- Application Server reserved folders:
  - META-INF
  - WEB-INF
- Web Pages (*.jsp) and web page resources:

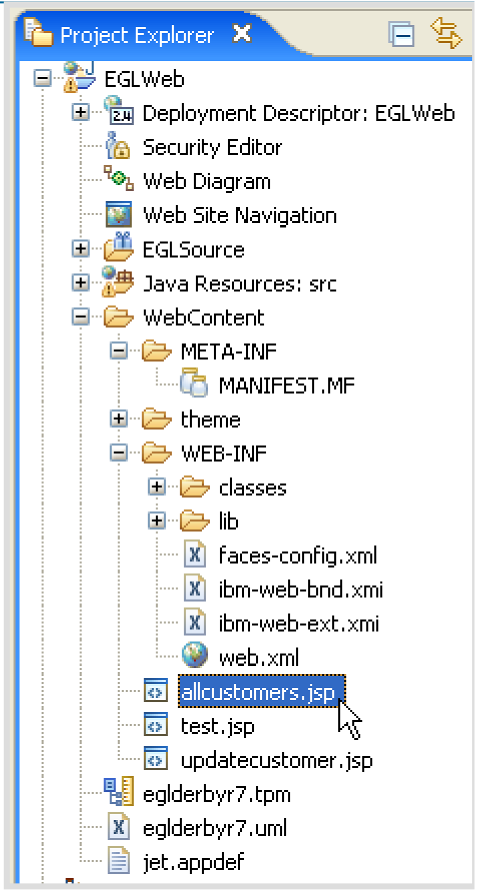
  `\theme\` - Graphics (*.gif, *.jpg)

  `\theme\` - Template pages (*.htpl)

  `\theme\` - Cascading Style Sheets (*.css)

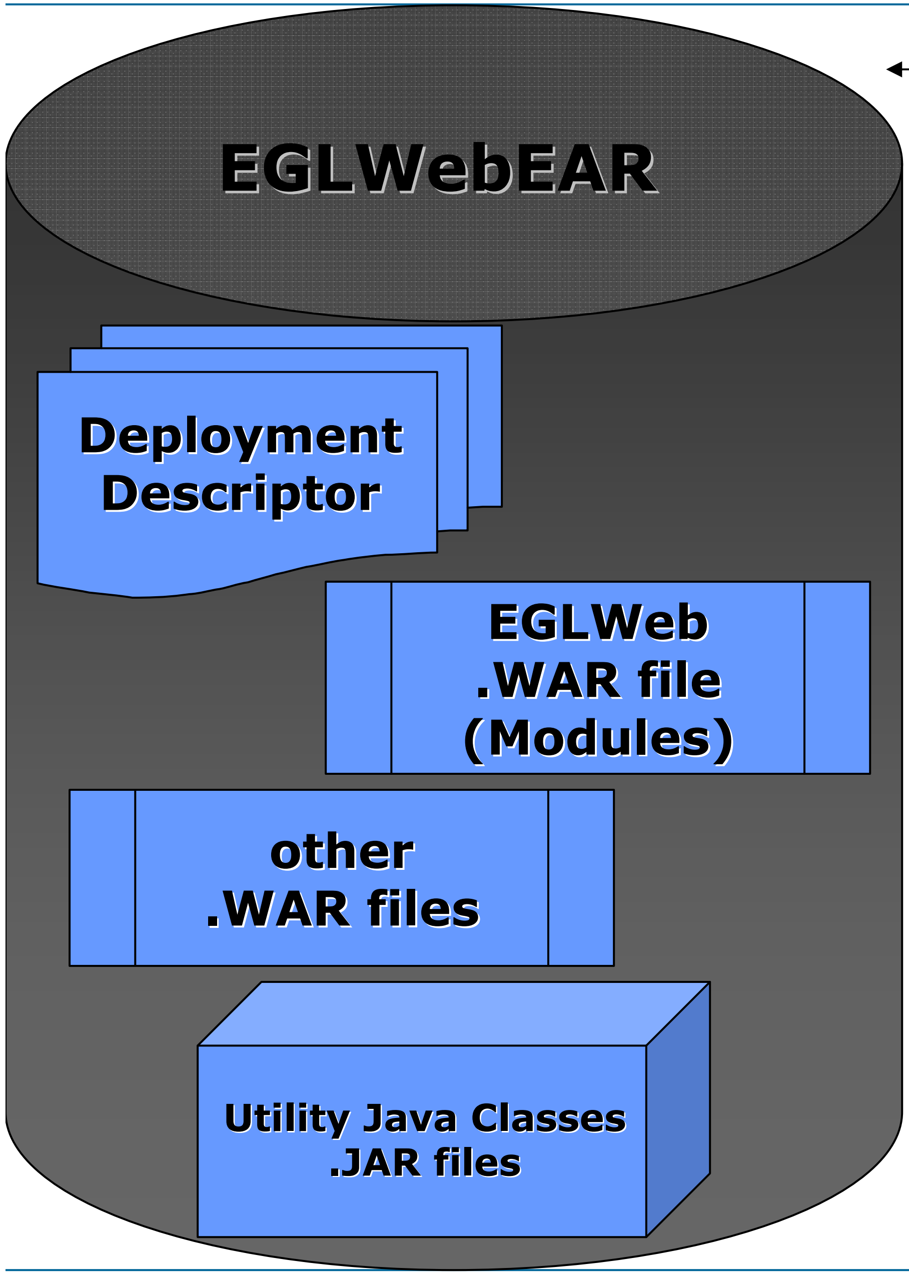
✐ Note that your installed EGLWeb project might have a few additional folders and files – such as an \images\ directory, that we've supplied, containing graphics files you'll use throughout the course

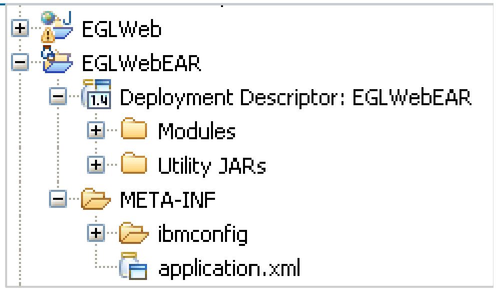
? And if you're not familiar with the above types of web files don't worry. We'll be covering them later on in the course



```
EGLWeb
  Deployment Descriptor: EGLWeb
  Security Editor
  Web Diagram
  Web Site Navigation
  EGLSource
  Java Resources: src
  WebContent
    META-INF
      MANIFEST.MF
    theme
    WEB-INF
      classes
      lib
      faces-config.xml
      ibm-web-bnd.xmi
      ibm-web-ext.xmi
      web.xml
    allcustomers.jsp
    test.jsp
    updatecustomer.jsp
  eglderbyr7.tpm
  eglderbyr7.uml
  jet.appdef
```

Terms and Concepts – .EAR File – Simplified J2EE Deployment – for WAS Users





- An EAR (**E**nterprise **AR**chive) file contains a J2EE application, which is a collection of J2EE modules. The modules contain Java generated from EGL. Because individual J2EE modules are contained within the EAR, they can easily be managed and deployed as a whole to WebSphere.
- An EAR also has a Deployment Descriptor file which describes how it should be deployed to WebSphere. In addition, each module within the EAR has its own Deployment Descriptor.
- An EAR file is similar to an iSeries "**savefile** for i" – or to a System z "**load library**" – which are files deployed on mainframes that also contain individual "modules"

Workshop

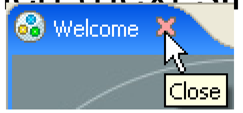
⏭ Launch RBD

- ⏭ From the Windows start menu, launch RBD …or…RAD with RBD (depending on what product you have installed on your PC)

- ⏭ At the Workspace Launcher window, specify the directory into which you want to organize all of your RBD Education resources.
  - ⏭ Usually something like: **`c:\RBDv7\`** …or… **`d:\RBDv7\`**

- ⏭ Close the Welcome window

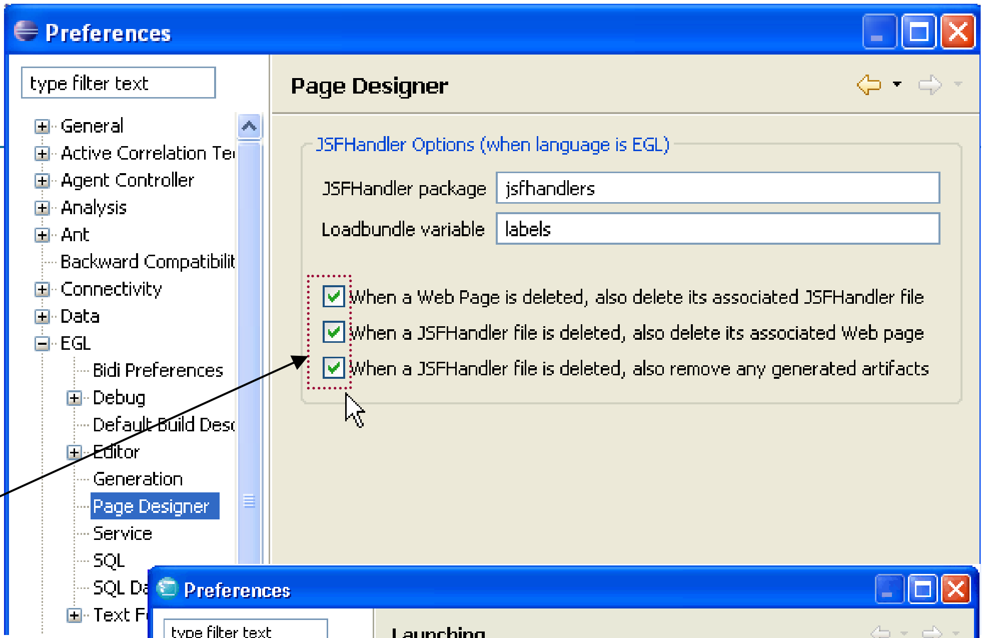
This opens to the RBD Workbench (next slide)



⏭ Configure Preferences – 1 of 3

You need to configure a few preferences, in order to complete the labs in this course.

- From the RBD Workbench menu
  - Pull down the **Window** menu, and select **Preferences…**
  - Expand the **EGL** folder and from within the EGL folder…
    - Click **Page Designer**
    - Check ☑ all three options as shown in the screen capture
    - Click **APPLY**



Preferences

Page Designer

JSFHandler Options (when language is EGL)

JSFHandler package: jsfhandlers

Loadbundle variable: labels

☑ When a Web Page is deleted, also delete its associated JSFHandler file

☑ When a JSFHandler file is deleted, also delete its associated Web page

☑ When a JSFHandler file is deleted, also remove any generated artifacts

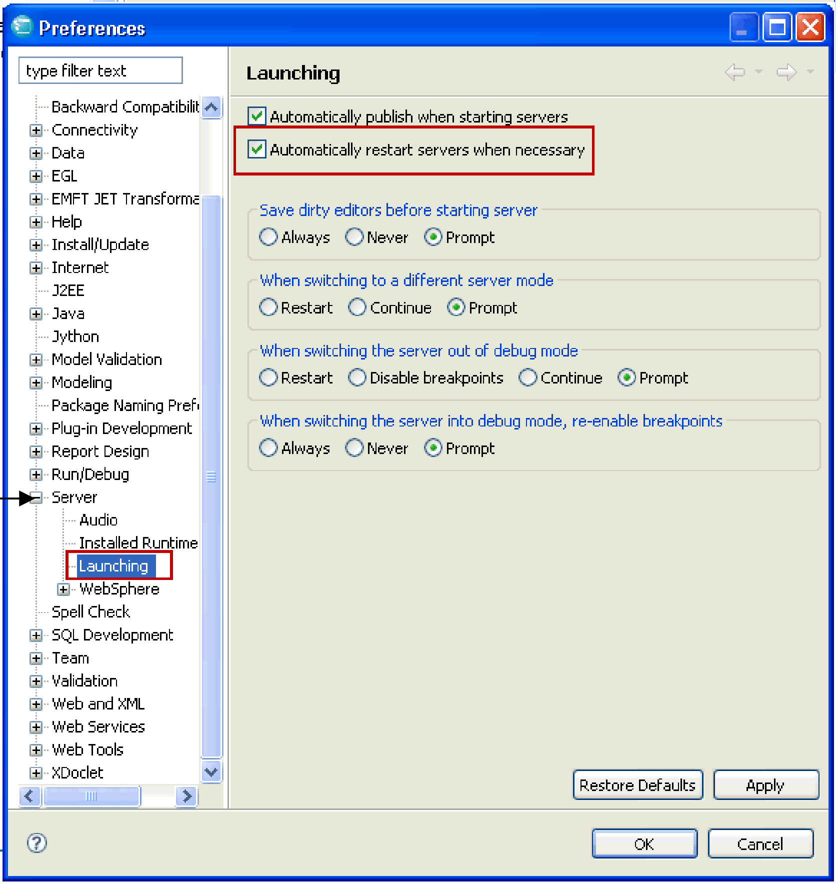
- **Expand the Server Preferences**

  From **Launching** – check to see that:

  ☑ **Automatically restart servers when necessary is on by default**

  Click Apply

Do not leave Preferences yet



Preferences

Launching

☑ Automatically publish when starting servers

☑ Automatically restart servers when necessary

Save dirty editors before starting server: ○ Always ○ Never [✓ ● Prompt]

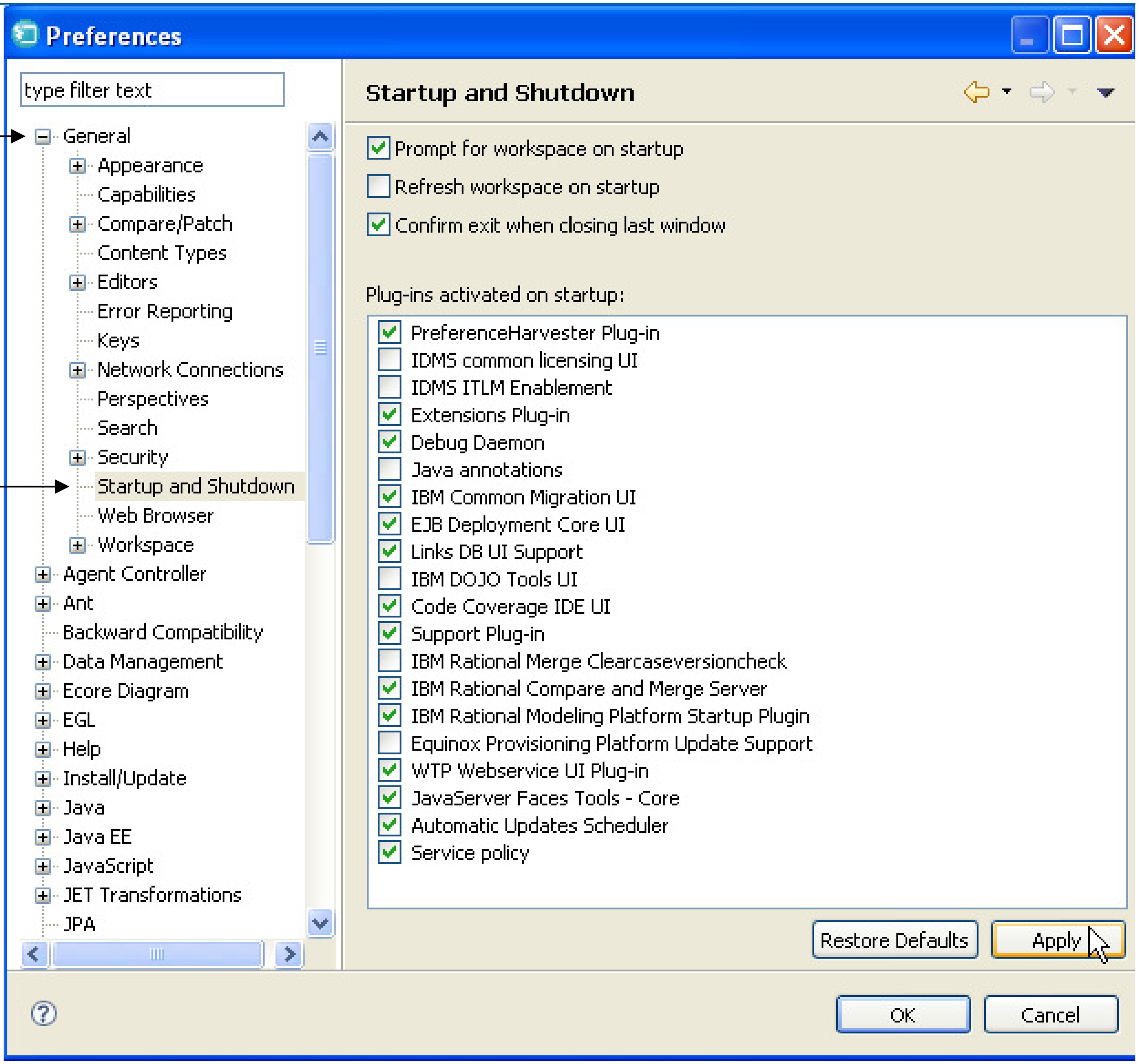
When switching to a different server mode: ○ Restart ○ Continue [✓ ● Prompt]

When switching the server out of debug mode: ○ Restart ○ Disable breakpoints ○ Continue [✓ ● Prompt]

When switching the server into debug mode, re-enable breakpoints: ○ Always ○ Never [✓ ● Prompt]

⏭ Configure Preferences – 2 of 3

- Still from within the Preferences Window:
- From **General**
  - Startup and Shutdown:
    - **Uncheck all of the options shown here in the screen capture ➔**
- **Click Apply**



Startup and Shutdown

- [✓] Prompt for workspace on startup
- [ ] Refresh workspace on startup
- [✓] Confirm exit when closing last window

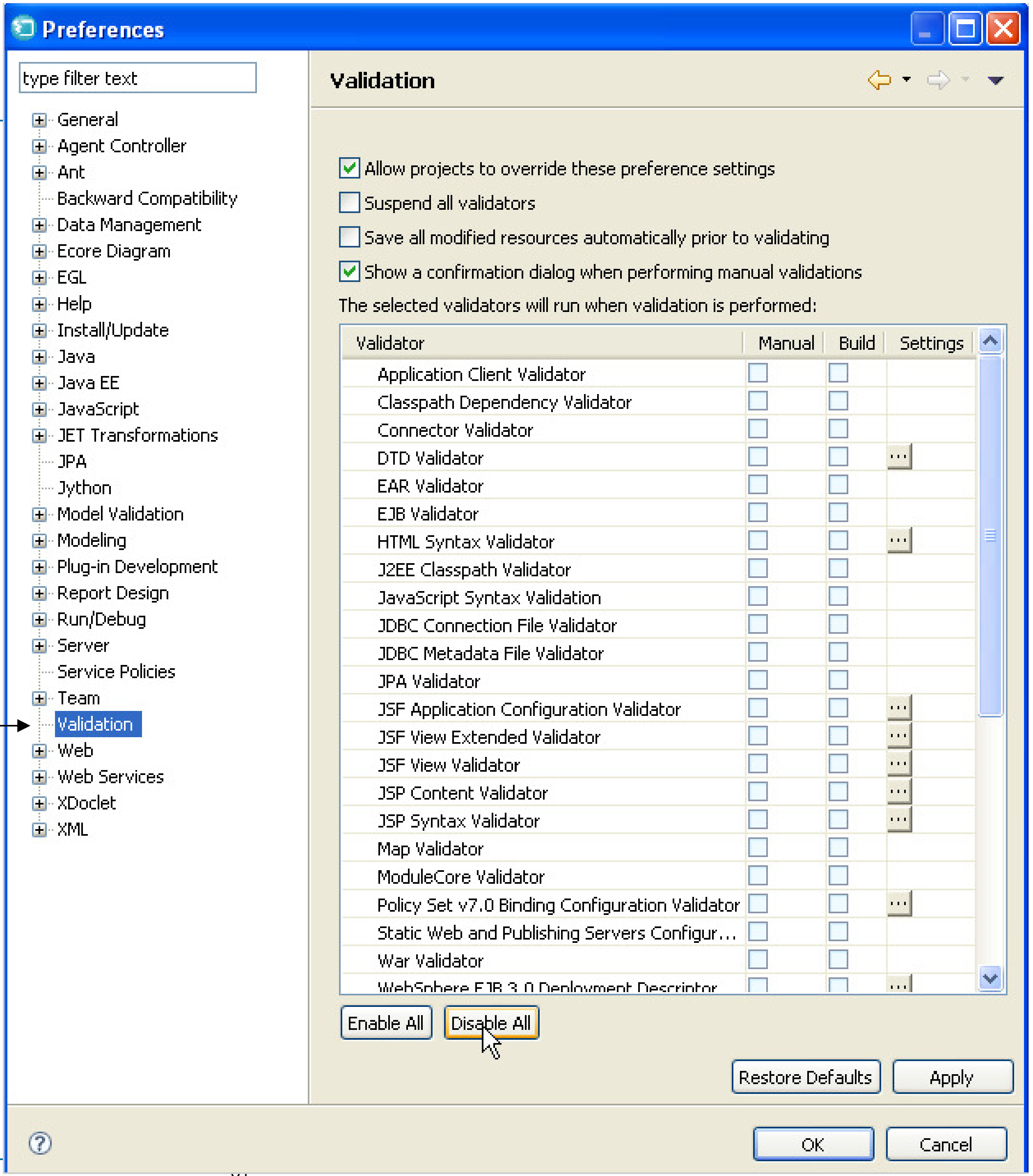
Plug-ins activated on startup:

- [✓] PreferenceHarvester Plug-in
- [ ] IDMS common licensing UI
- [ ] IDMS ITLM Enablement
- [✓] Extensions Plug-in
- [✓] Debug Daemon
- [ ] Java annotations
- [✓] IBM Common Migration UI
- [✓] EJB Deployment Core UI
- [✓] Links DB UI Support
- [ ] IBM DOJO Tools UI
- [✓] Code Coverage IDE UI
- [✓] Support Plug-in
- [ ] IBM Rational Merge Clearcaseversioncheck
- [✓] IBM Rational Compare and Merge Server
- [✓] IBM Rational Modeling Platform Startup Plugin
- [ ] Equinox Provisioning Platform Update Support
- [✓] WTP Webservice UI Plug-in
- [✓] JavaServer Faces Tools - Core
- [✓] Automatic Updates Scheduler
- [✓] Service policy

⏭ Configure &* Preferences – 3 of 3

Still from within the Preferences Window

- **From Validation**
  - **Click Disable All**
  - Click **Apply**
- **If prompted, do a full rebuild**



Validation

[✓] Allow projects to override these preference settings
[ ] Suspend all validators
[ ] Save all modified resources automatically prior to validating
[✓] Show a confirmation dialog when performing manual validations

The selected validators will run when validation is performed:

Validator	Manual	Build	Settings
Application Client Validator	☐	☐	
Classpath Dependency Validator	☐	☐	
Connector Validator	☐	☐	
DTD Validator	☐	☐	...
EAR Validator	☐	☐	
EJB Validator	☐	☐	
HTML Syntax Validator	☐	☐	...
J2EE Classpath Validator	☐	☐	
JavaScript Syntax Validation	☐	☐	
JDBC Connection File Validator	☐	☐	
JDBC Metadata File Validator	☐	☐	
JPA Validator	☐	☐	
JSF Application Configuration Validator	☐	☐	...
JSF View Extended Validator	☐	☐	...
JSF View Validator	☐	☐	...
JSP Content Validator	☐	☐	...
JSP Syntax Validator	☐	☐	...
Map Validator	☐	☐	
ModuleCore Validator	☐	☐	
Policy Set v7.0 Binding Configuration Validator	☐	☐	...
Static Web and Publishing Servers Configur...	☐	☐	
War Validator	☐	☐	
WebSphere EJB 3.0 Deployment Descriptor	☐	☐	...

⏭ Workspace Preferences – Customizing the Workbench

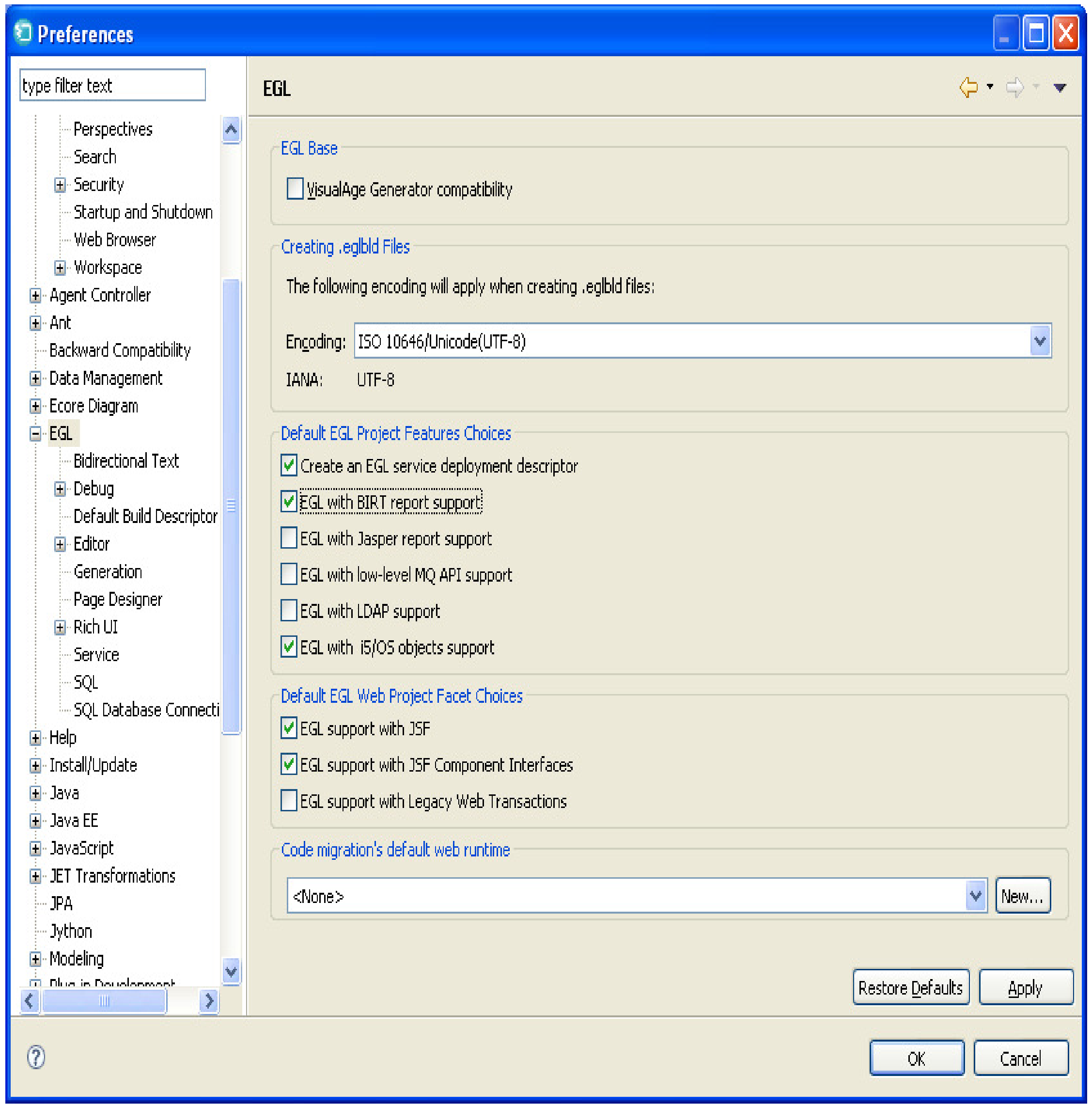
The Preferences dialog (under the Windows menu) allows you to customize your workspace development environment.

⏭ From the main Preferences dialog you should check:

- ☑ **EGL with BIRT report support**
- ☑ **EGL support with JSF Component Interfaces**

Other preferences you can customize:

- **Editor**
  - You can change the editor's appearance
- **Page Designer**
  - Specify whether to delete associated files
- **SQL Database Connections**
  - Specify a connection to an external database (DB2, IDS, etc.) for SQLRetrieve
- **Click Apply**



EGL

EGL Base
- [ ] VisualAge Generator compatibility

Creating .eglbld Files

The following encoding will apply when creating .eglbld files:

Encoding: ISO 10646/Unicode(UTF-8)

IANA: UTF-8

Default EGL Project Features Choices
- [✓] Create an EGL service deployment descriptor
- [✓] EGL with BIRT report support
- [ ] EGL with Jasper report support
- [ ] EGL with low-level MQ API support
- [ ] EGL with LDAP support
- [✓] EGL with i5/OS objects support

Default EGL Web Project Facet Choices
- [✓] EGL support with JSF
- [✓] EGL support with JSF Component Interfaces
- [ ] EGL support with Legacy Web Transactions

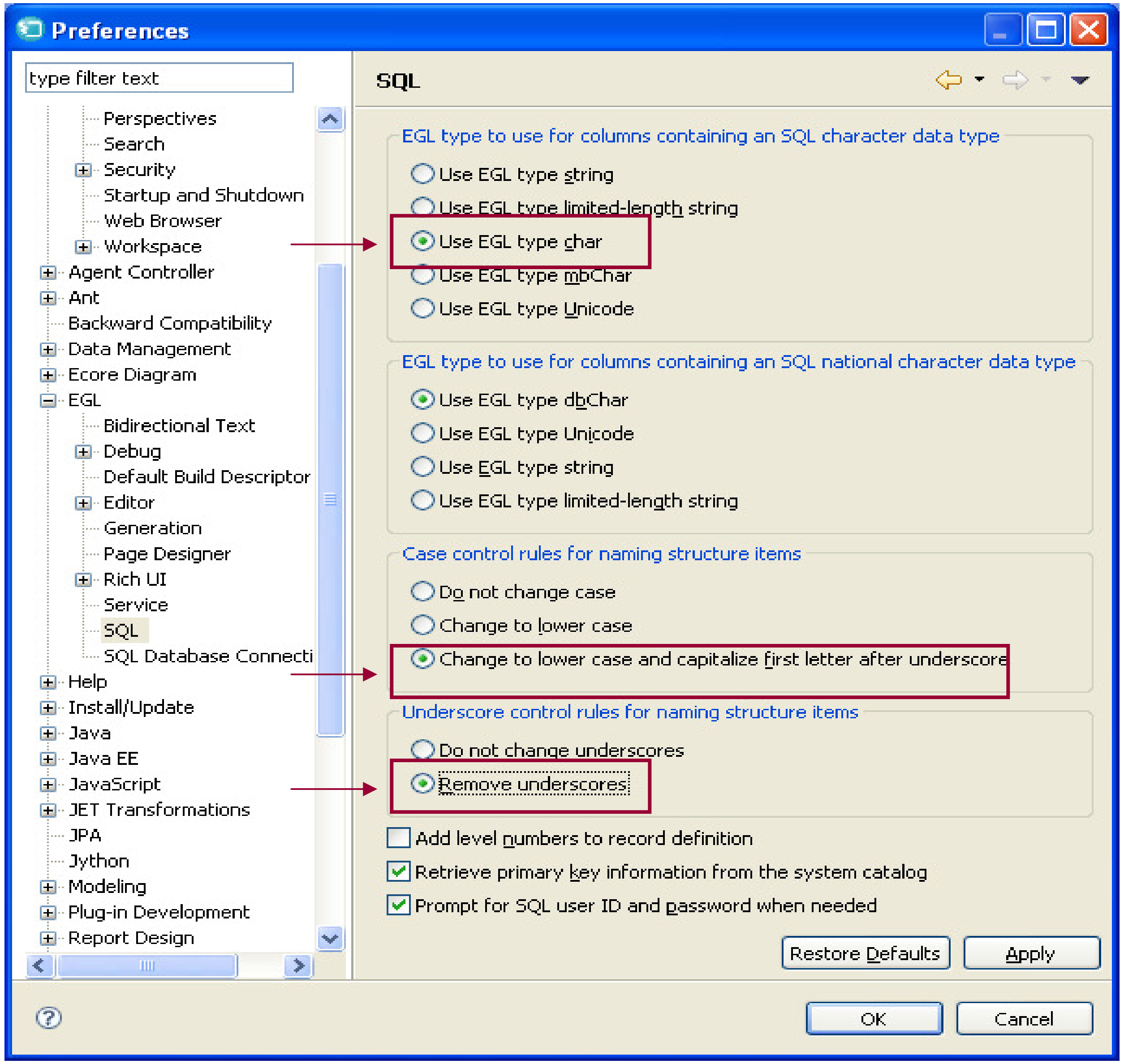
Code migration's default web runtime

<None> New...

Workspace Preferences – Customizing the Workbench – Modify SQL Preferences

The SQL Preferences allow you to customize how the RBD tooling generates SQL resources

- From the **SQL** Preferences dialog check:
  - ⊙ Change to lower case and capitalize first letter after underscore
  - ⊙ Remove underscores



SQL

EGL type to use for columns containing an SQL character data type
- ◯ Use EGL type string
- ◯ Use EGL type limited-length string
- ● Use EGL type char
- ◯ Use EGL type mbChar
- ◯ Use EGL type Unicode

EGL type to use for columns containing an SQL national character data type
- ● Use EGL type dbChar
- ◯ Use EGL type Unicode
- ◯ Use EGL type string
- ◯ Use EGL type limited-length string

Case control rules for naming structure items
- ◯ Do not change case
- ◯ Change to lower case
- ● Change to lower case and capitalize first letter after underscore

Underscore control rules for naming structure items
- ◯ Do not change underscores
- ● Remove underscores

- ☐ Add level numbers to record definition
- ☑ Retrieve primary key information from the system catalog
- ☑ Prompt for SQL user ID and password when needed

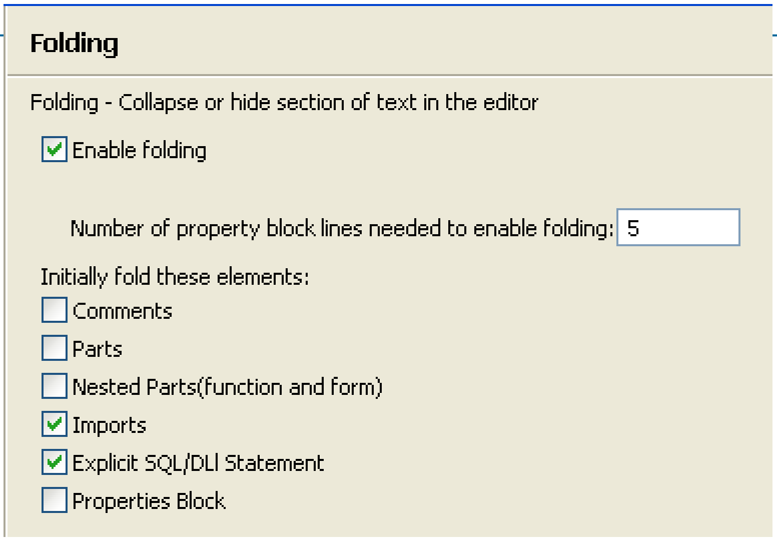
- **Click Apply**

Workspace Preferences Lab – EGL Editor

Editor Preferences

- You can change the editor's appearance, and how it treats and displays your source statements

UN-check: Annotate errors as you type


Folding

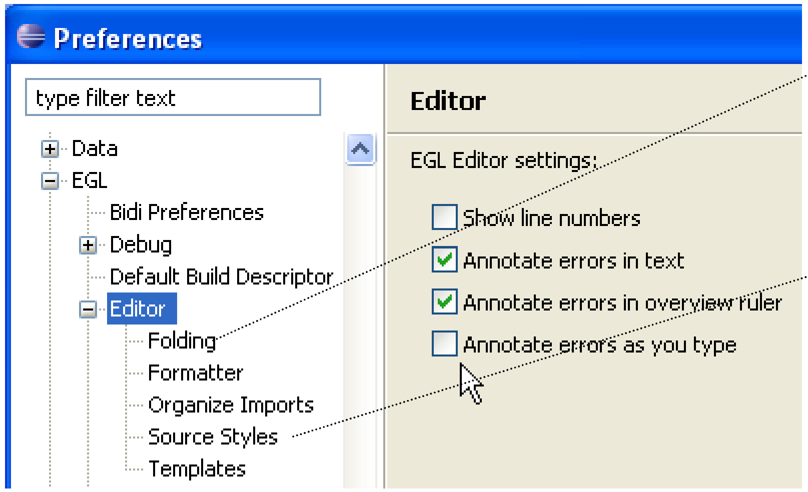
Folding - Collapse or hide section of text in the editor

- [✓] Enable folding

Number of property block lines needed to enable folding: 5

Initially fold these elements:

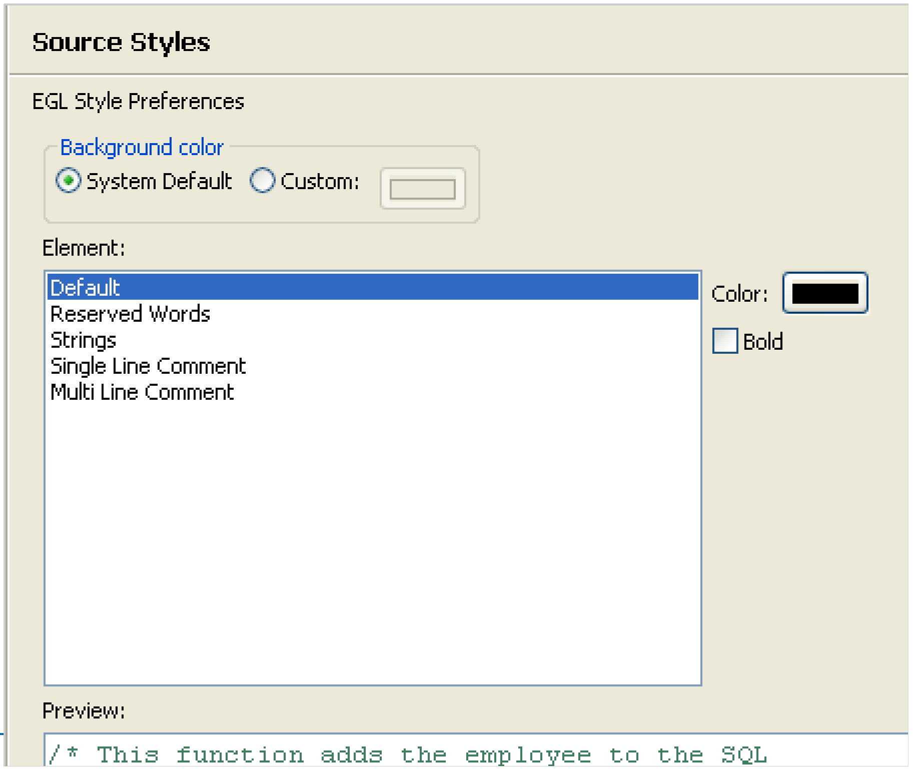
- [ ] Comments
- [ ] Parts
- [ ] Nested Parts(function and form)
- [✓] Imports
- [✓] Explicit SQL/DLI Statement
- [ ] Properties Block



Editor

EGL Editor settings:

- [ ] Show line numbers
- [✓] Annotate errors in text
- [✓] Annotate errors in overview ruler
- [ ] Annotate errors as you type


Source Styles

EGL Style Preferences

Background color: [● System Default] ○ Custom:

Element:

Default
Reserved Words
Strings
Single Line Comment
Multi Line Comment

Color: ■

- [ ] Bold

Preview:

```
/* This function adds the employee to the SQL
```

Optionally – feel free to try out other editor preferences

Click Apply

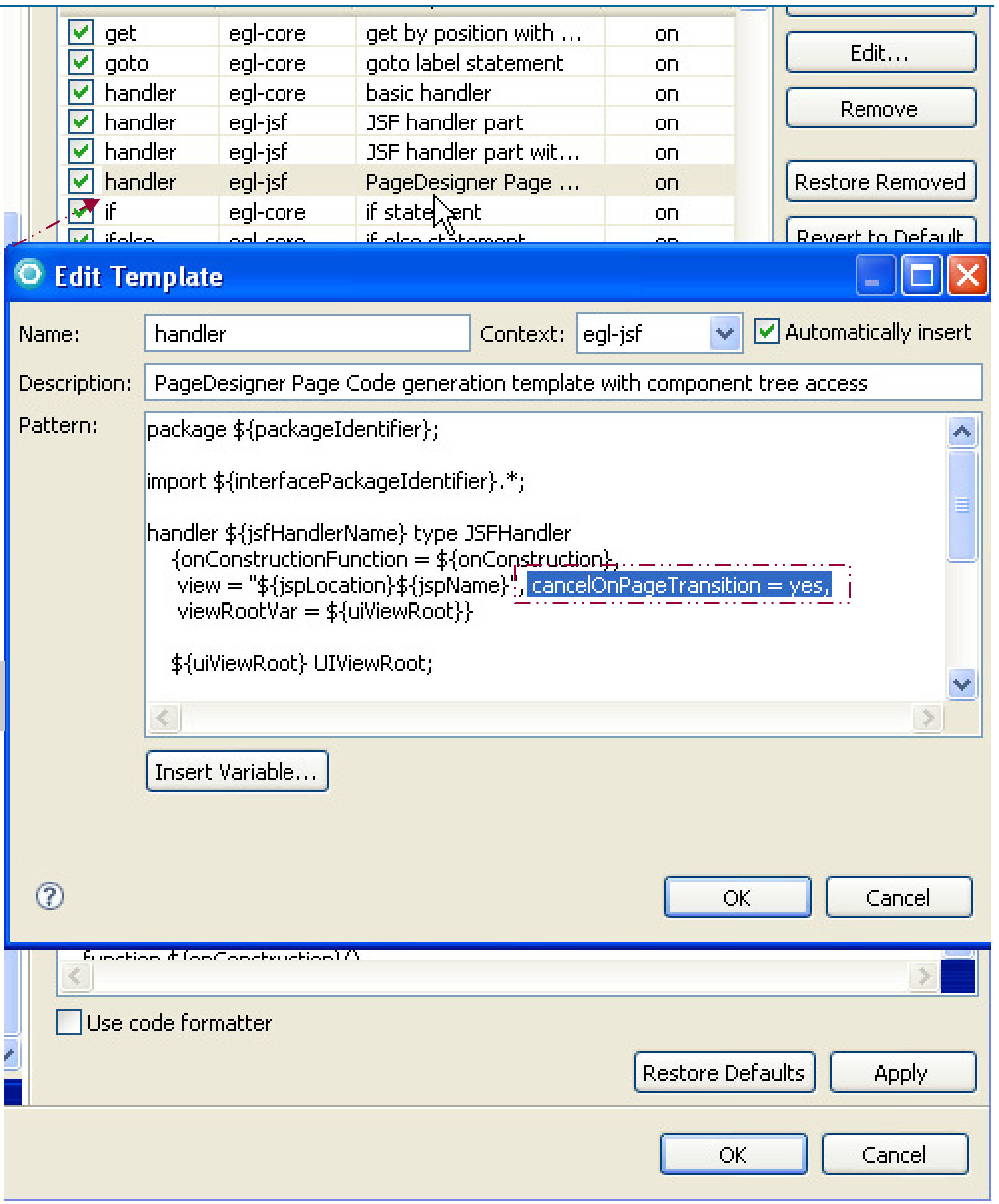
⏭ OPTIONAL Workspace Preferences Lab – EGL Editor Templates/Modify Defaults

Besides adding new statements, you may want to modify the existing Editor Templates:

- Change the defaults
- Add new Properties

⏭ **Optional** Workshop

- Select the **handler egl-jsf "PageDesigner Page Code generation template with component tree access"** template
- **Click Edit…**
- **cancelOnPageTransition=yes,**
- **Click OK**
  - To save your edits
- **Click OK**
  - To save your Template preferences



✓ get	egl-core	get by position with …	on
✓ goto	egl-core	goto label statement	on
✓ handler	egl-core	basic handler	on
✓ handler	egl-jsf	JSF handler part	on
✓ handler	egl-jsf	JSF handler part wit…	on
✓ handler	egl-jsf	PageDesigner Page …	on
✓ if	egl-core	if statement	on

Edit Template

Name: handler Context: egl-jsf ✓ Automatically insert

Description: PageDesigner Page Code generation template with component tree access

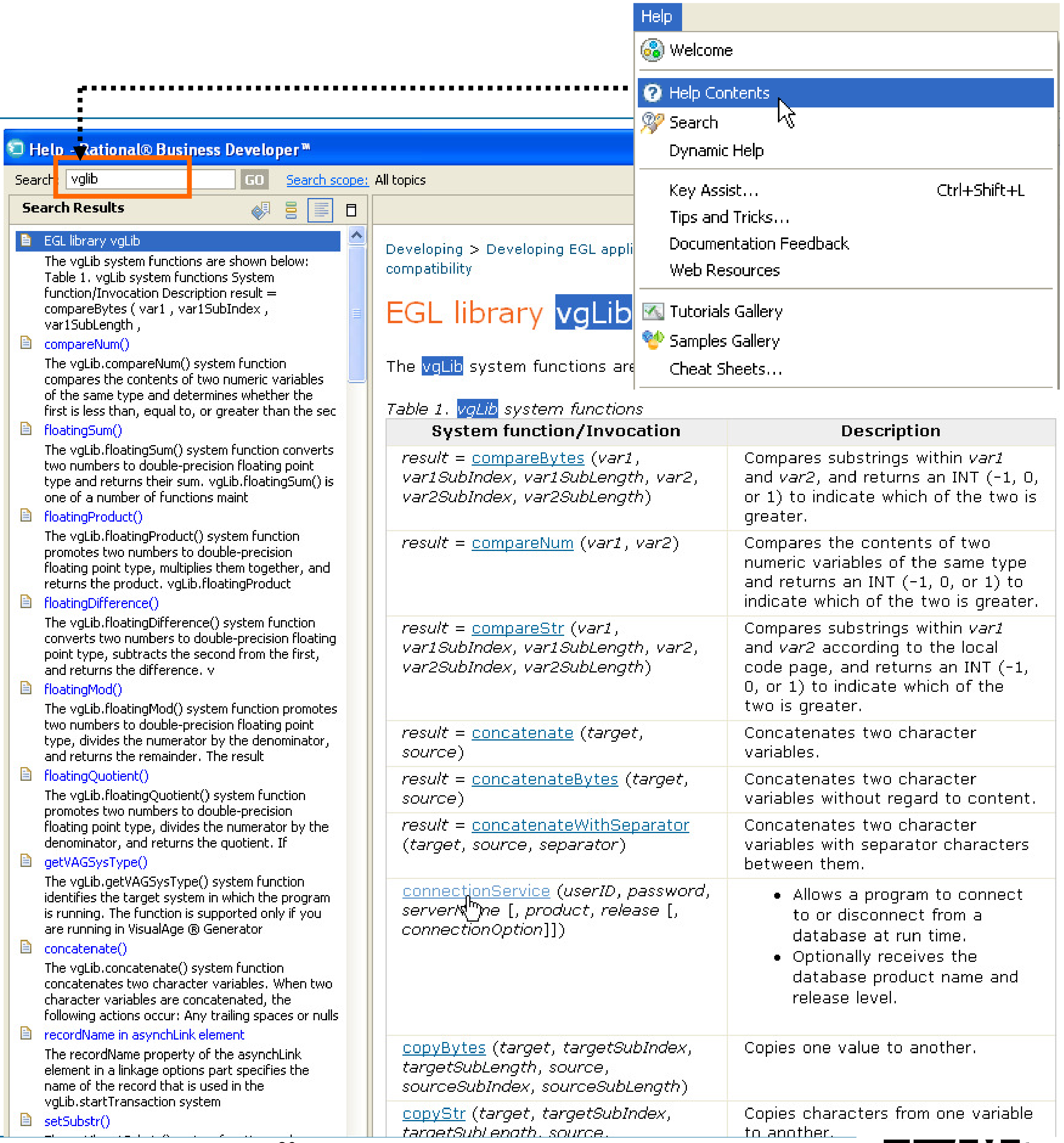
Pattern:

```
package ${packageIdentifier};

import ${interfacePackageIdentifier}.*;

handler ${jsfHandlerName} type JSFHandler
  {onConstructionFunction = ${onConstruction},
   view = "${jspLocation}${jspName}", cancelOnPageTransition = yes,
   viewRootVar = ${uiViewRoot}}

  ${uiViewRoot} UIViewRoot;
```

Help

There is a robust help system with documentation and examples available in many categories:

- **Keyword search**
- **Category search**
- **Web Resources**
  - Points you to the EGL forum and home page on DeveloperWorks
- **Tutorials Gallery**
  - Contains additional in-the-box education samples
- **Samples Gallery**
  - Contains working, sample applications
- **Cheat Sheets**
  - Contains step-by-step "how-to" instructions



EGL library vgLib
The vgLib system functions are shown below: Table 1. vgLib system functions System function/Invocation Description result = compareBytes (var1 , var1SubIndex , var1SubLength ,

compareNum()
The vgLib.compareNum() system function compares the contents of two numeric variables of the same type and determines whether the first is less than, equal to, or greater than the sec

floatingSum()
The vgLib.floatingSum() system function converts two numbers to double-precision floating point type and returns their sum. vgLib.floatingSum() is one of a number of functions maint

floatingProduct()
The vgLib.floatingProduct() system function promotes two numbers to double-precision floating point type, multiplies them together, and returns the product. vgLib.floatingProduct

floatingDifference()
The vgLib.floatingDifference() system function converts two numbers to double-precision floating point type, subtracts the second from the first, and returns the difference. v

floatingMod()
The vgLib.floatingMod() system function promotes two numbers to double-precision floating point type, divides the numerator by the denominator, and returns the remainder. The result

floatingQuotient()
The vgLib.floatingQuotient() system function promotes two numbers to double-precision floating point type, divides the numerator by the denominator, and returns the quotient. If

getVAGSysType()
The vgLib.getVAGSysType() system function identifies the target system in which the program is running. The function is supported only if you are running in VisualAge ® Generator

concatenate()
The vgLib.concatenate() system function concatenates two character variables. When two character variables are concatenated, the following actions occur: Any trailing spaces or nulls

recordName in asynchLink element
The recordName property of the asynchLink element in a linkage options part specifies the name of the record that is used in the vgLib.startTransaction system

setSubstr()

Developing > Developing EGL appli
compatibility

EGL library vgLib

The vgLib system functions are

Table 1. vgLib system functions

System function/Invocation	Description
result = compareBytes (*var1*, *var1SubIndex*, *var1SubLength*, *var2*, *var2SubIndex*, *var2SubLength*)	Compares substrings within *var1* and *var2*, and returns an INT (-1, 0, or 1) to indicate which of the two is greater.
result = compareNum (*var1*, *var2*)	Compares the contents of two numeric variables of the same type and returns an INT (-1, 0, or 1) to indicate which of the two is greater.
result = compareStr (*var1*, *var1SubIndex*, *var1SubLength*, *var2*, *var2SubIndex*, *var2SubLength*)	Compares substrings within *var1* and *var2* according to the local code page, and returns an INT (-1, 0, or 1) to indicate which of the two is greater.
result = concatenate (*target*, *source*)	Concatenates two character variables.
result = concatenateBytes (*target*, *source*)	Concatenates two character variables without regard to content.
result = concatenateWithSeparator (*target*, *source*, *separator*)	Concatenates two character variables with separator characters between them.
connectionService (*userID*, *password*, *serverName* [, *product*, *release* [, *connectionOption*]])	• Allows a program to connect to or disconnect from a database at run time. • Optionally receives the database product name and release level.
copyBytes (*target*, *targetSubIndex*, *targetSubLength*, *source*, *sourceSubIndex*, *sourceSubLength*)	Copies one value to another.
copyStr (*target*, *targetSubIndex*, *targetSubLength*, *source*,	Copies characters from one variable to another.

UNIT

System i Connections

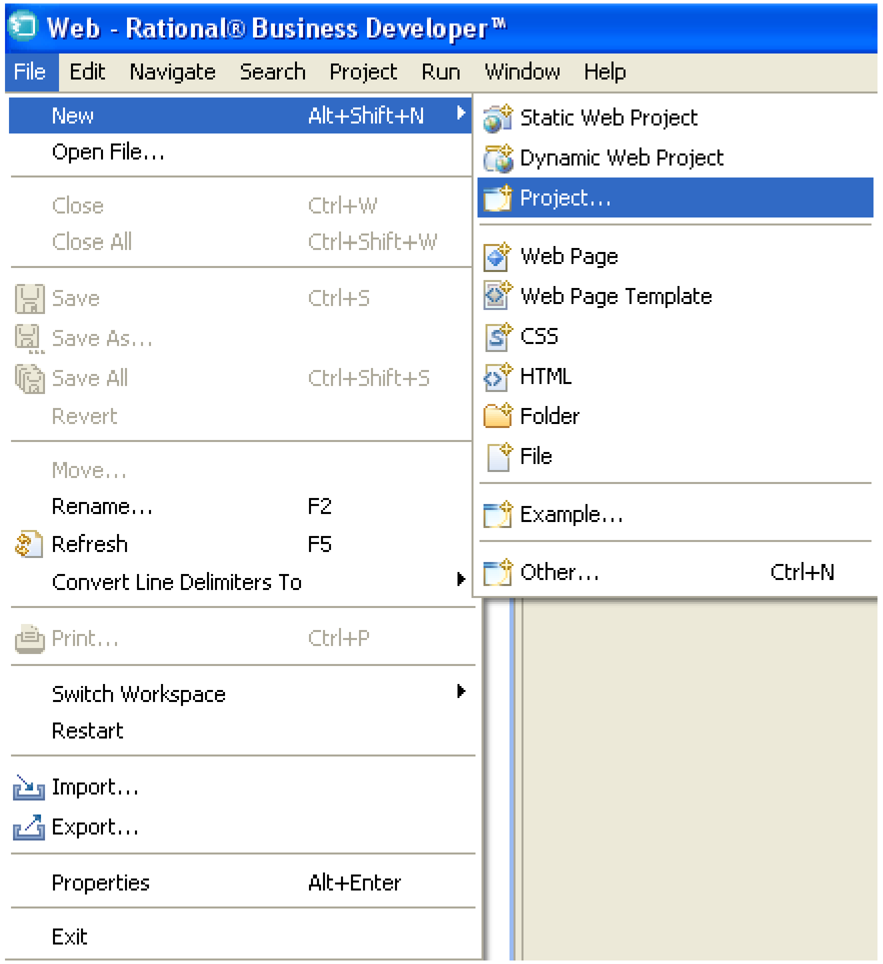
Topics:

- **Creating a New Project**
- Creating a DB2/400 Connection
- Defining a DB2/400 Runtime Data Source
- Adding JT400 Toolkit to Your Project

⏭ Lab – Launch RBD and Create a New Project

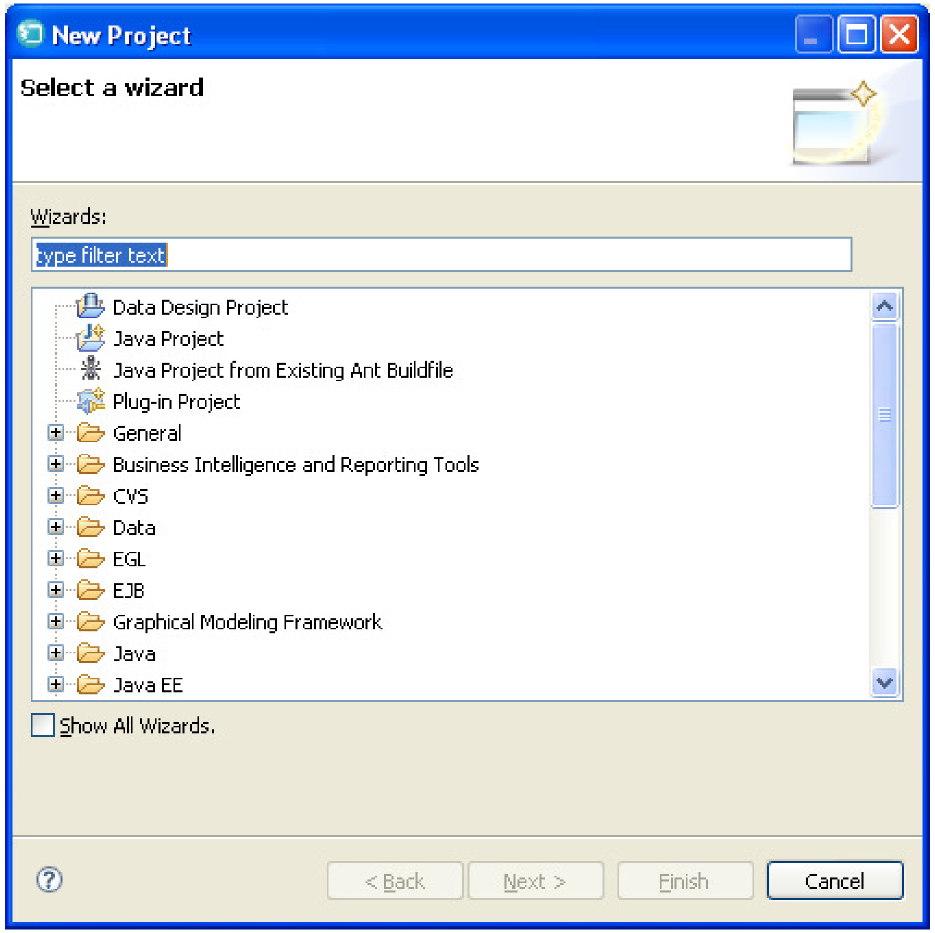
(If RBD is not already up and running)

- From the Windows start menu launch RBD
- From the initial prompt, specify or browse to and select your Workspace
- Select File → New → Project to create a new project



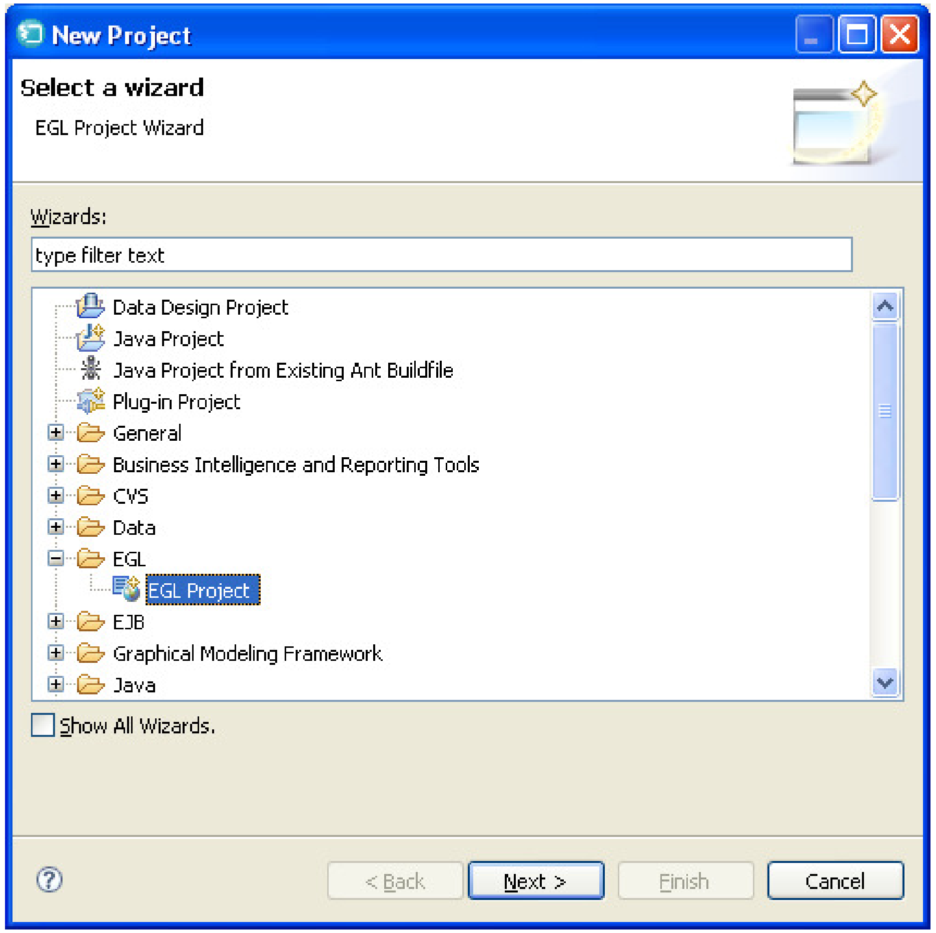
Lab – Create a New Project

- Expand EGL folder



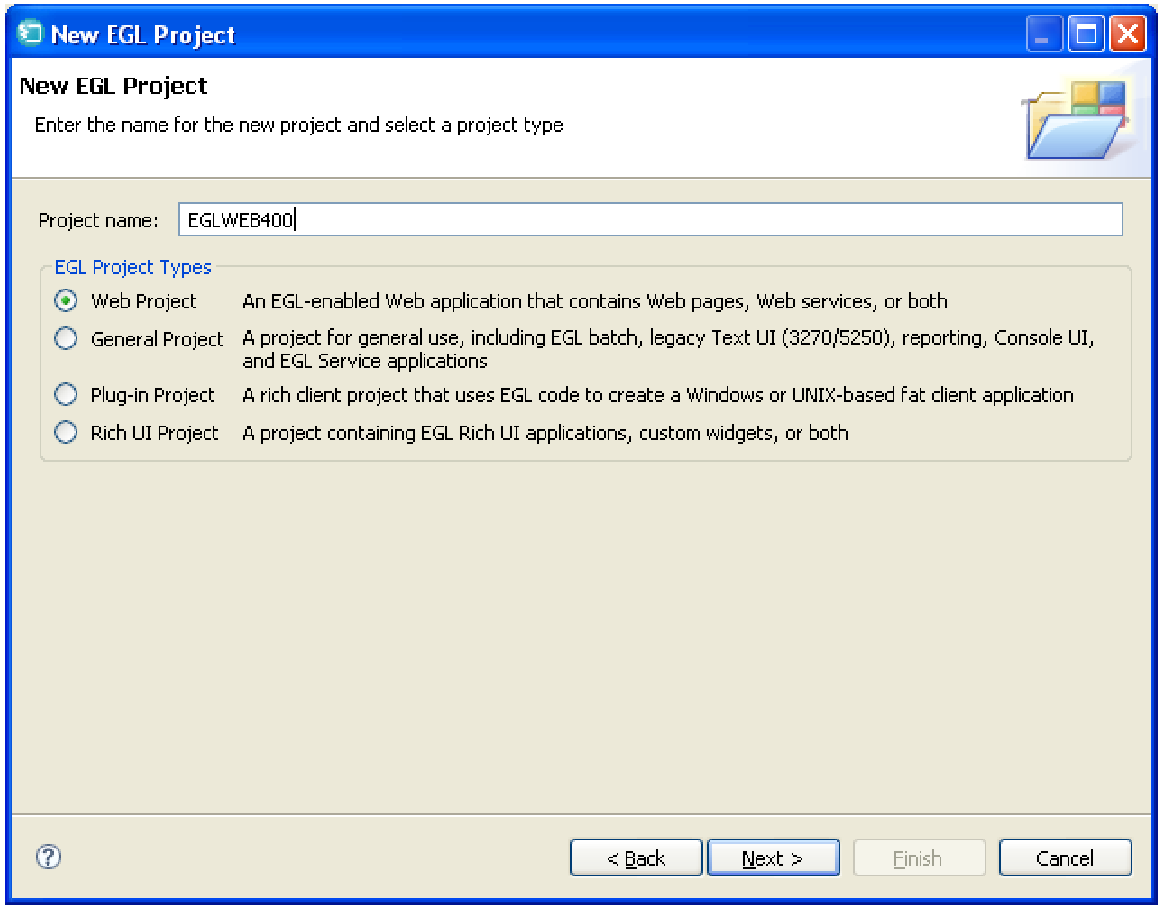
Lab – Create a New Project

- Select "EGL Project"
- Click "Next"



⏭ Lab – Create a New Project

- Enter Project Name as "EGLWEB400"
- Select "Web Project" radio button
- Click "Next"



New EGL Project

Enter the name for the new project and select a project type

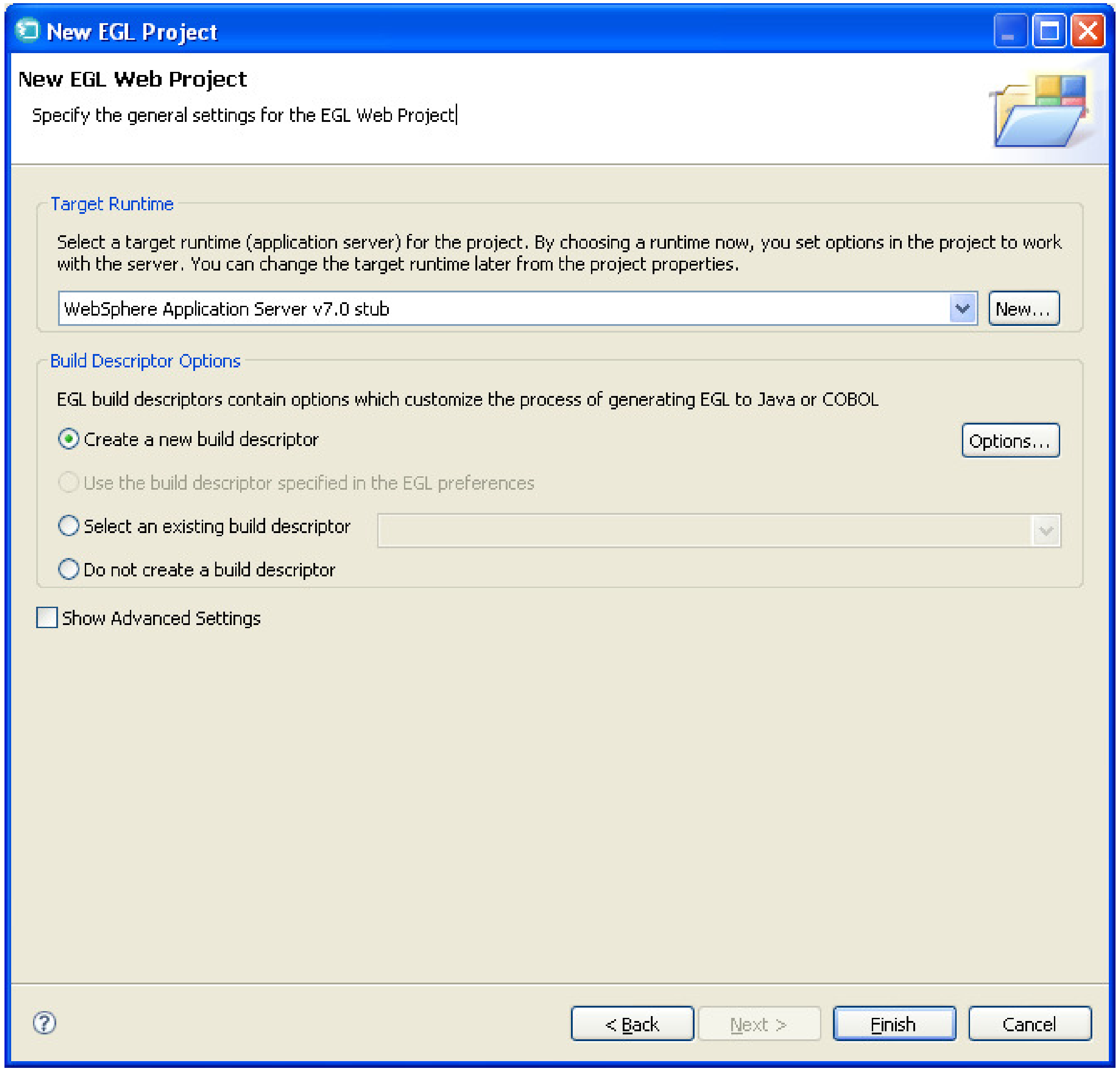
Project name: EGLWEB400

EGL Project Types

[✓ Web Project]	An EGL-enabled Web application that contains Web pages, Web services, or both
General Project	A project for general use, including EGL batch, legacy Text UI (3270/5250), reporting, Console UI, and EGL Service applications
Plug-in Project	A rich client project that uses EGL code to create a Windows or UNIX-based fat client application
Rich UI Project	A project containing EGL Rich UI applications, custom widgets, or both

⏭ Lab – Create a New Project

- Click “New”



New EGL Project

New EGL Web Project

Specify the general settings for the EGL Web Project

Target Runtime

Select a target runtime (application server) for the project. By choosing a runtime now, you set options in the project to work with the server. You can change the target runtime later from the project properties.

WebSphere Application Server v7.0 stub

Build Descriptor Options

EGL build descriptors contain options which customize the process of generating EGL to Java or COBOL

● Create a new build descriptor

○ Use the build descriptor specified in the EGL preferences

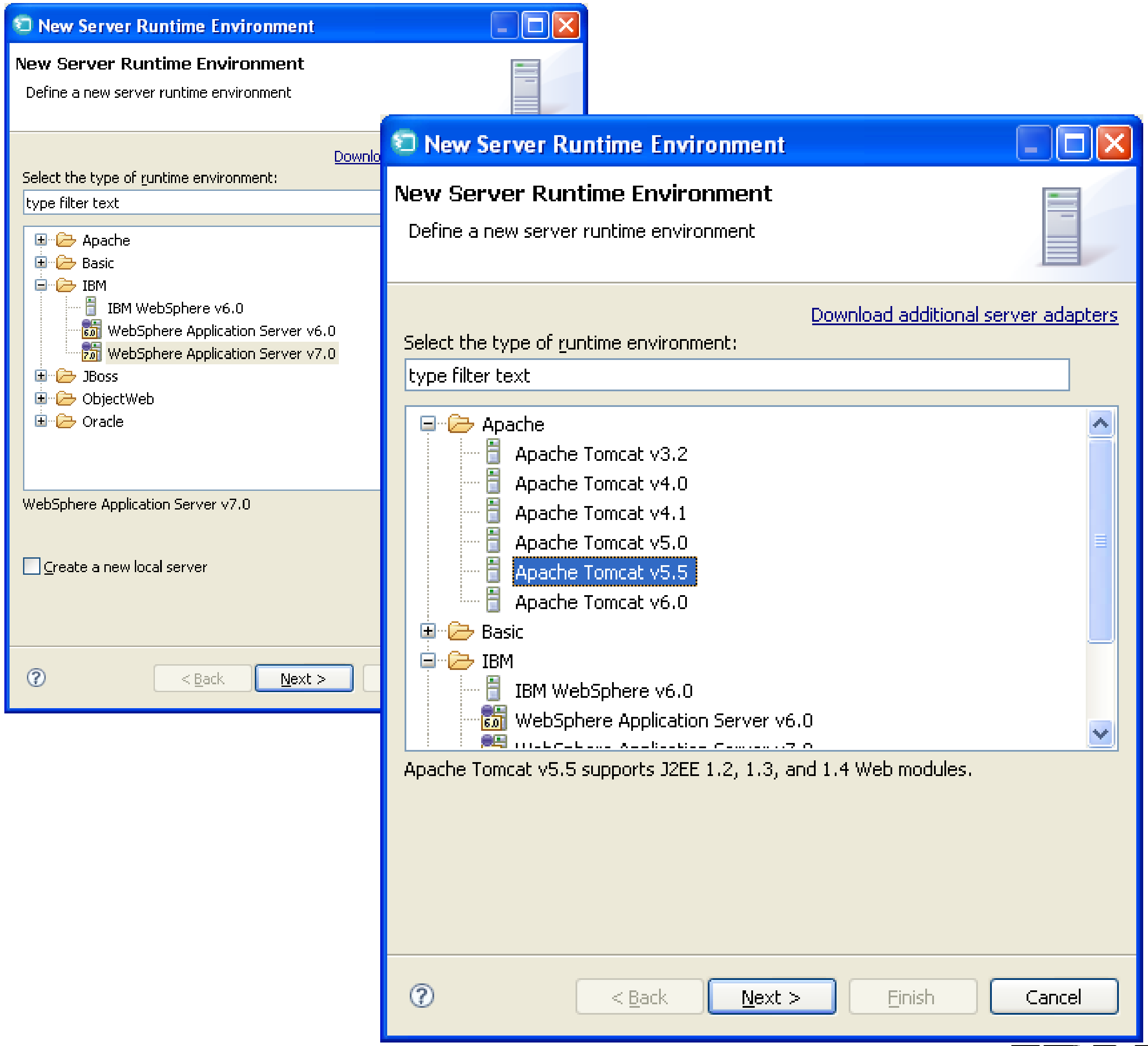
○ Select an existing build descriptor

○ Do not create a build descriptor

☐ Show Advanced Settings

⏭ Lab – Create a New Project

- Expand "Apache" folder
- Select "Apache Tomcat v5.5"
- Click "Next"


New Server Runtime Environment

New Server Runtime Environment

Define a new server runtime environment

Download additional server adapters

Select the type of runtime environment:

type filter text

```
⊟ Apache
    Apache Tomcat v3.2
    Apache Tomcat v4.0
    Apache Tomcat v4.1
    Apache Tomcat v5.0
    Apache Tomcat v5.5
    Apache Tomcat v6.0
⊞ Basic
⊟ IBM
    IBM WebSphere v6.0
    WebSphere Application Server v6.0
    WebSphere Application Server v7.0
⊞ JBoss
⊞ ObjectWeb
⊞ Oracle
```

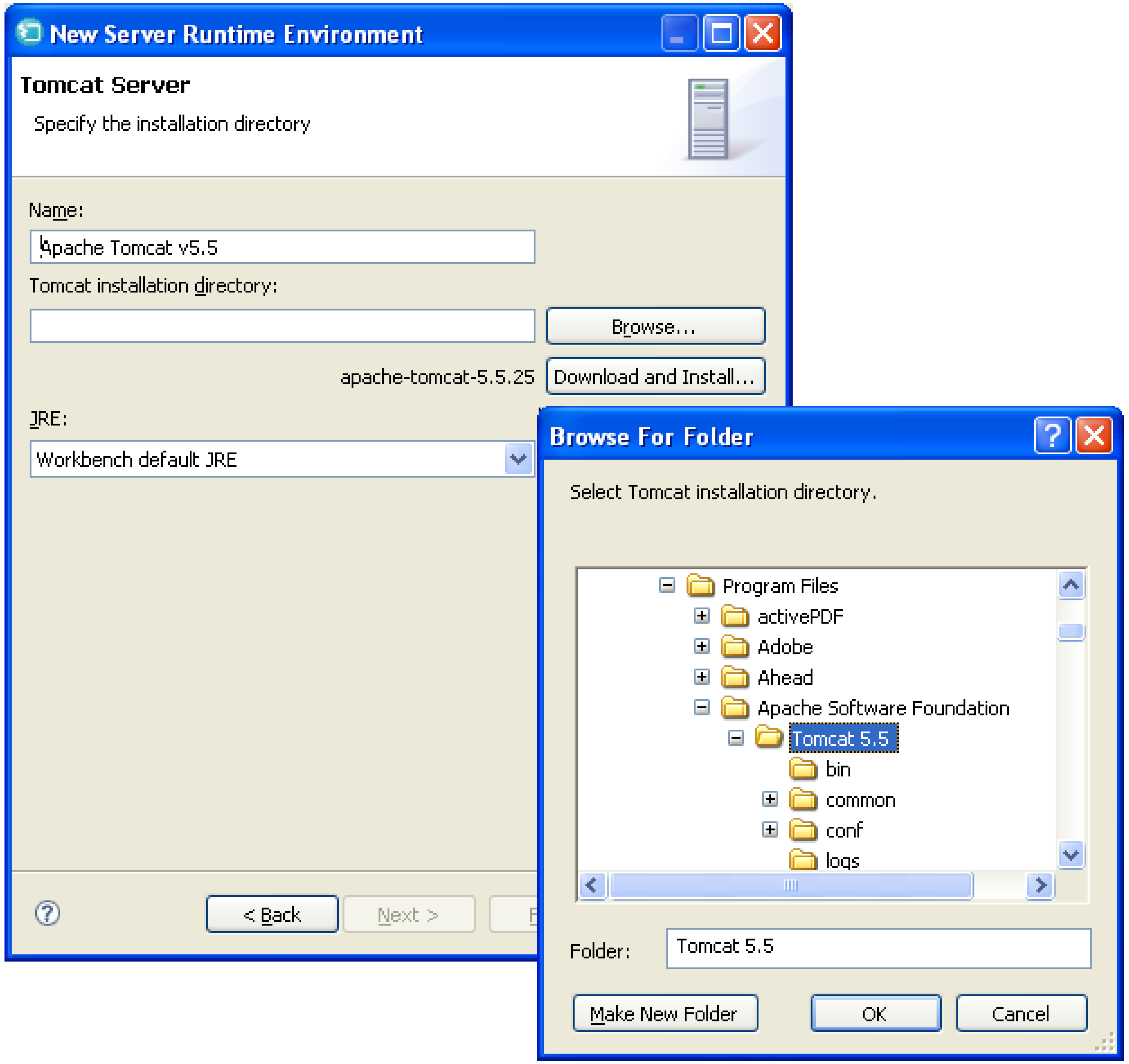
WebSphere Application Server v7.0

☐ Create a new local server

Apache Tomcat v5.5 supports J2EE 1.2, 1.3, and 1.4 Web modules.

Lab – Create a New Project

- Click "Browse"
- Open "Program Files" folder
- Open "Apache Software Foundation" folder
- Select "Tomcat 5.5"
- Click "OK"



New Server Runtime Environment

Tomcat Server

Specify the installation directory

Name: Apache Tomcat v5.5

Tomcat installation directory:

apache-tomcat-5.5.25

JRE: Workbench default JRE

Browse For Folder

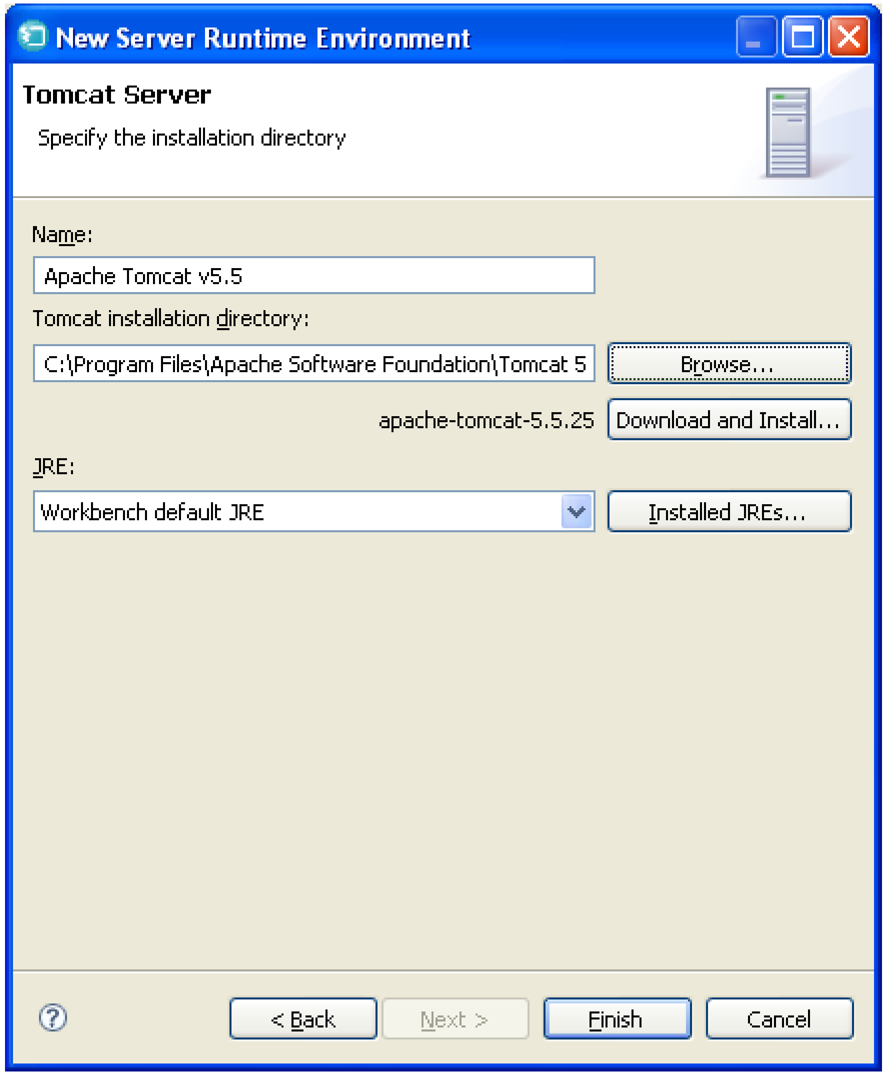
Select Tomcat installation directory.

Program Files
activePDF
Adobe
Ahead
Apache Software Foundation
Tomcat 5.5
bin
common
conf
logs

Folder: Tomcat 5.5

Lab – Create a New Project

- Click “Finish”



New Server Runtime Environment

Tomcat Server

Specify the installation directory

Name: Apache Tomcat v5.5

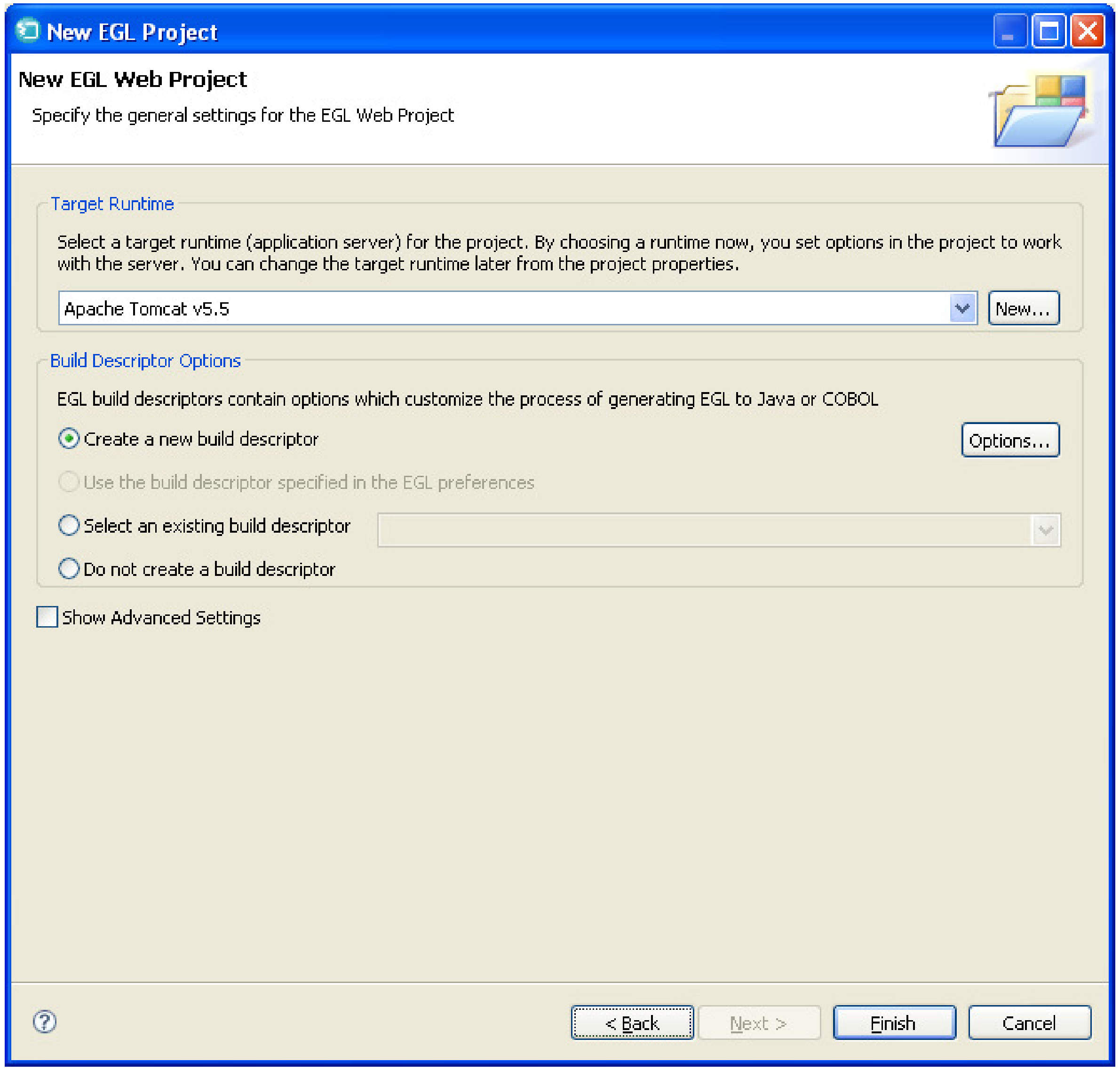
Tomcat installation directory: C:\Program Files\Apache Software Foundation\Tomcat 5

apache-tomcat-5.5.25

JRE: Workbench default JRE

Lab – Create a New Project

- Select "Apache Tomcat v5.5" as the Target Runtime
- Click "Finish"



New EGL Web Project

Specify the general settings for the EGL Web Project

Target Runtime

Select a target runtime (application server) for the project. By choosing a runtime now, you set options in the project to work with the server. You can change the target runtime later from the project properties.

Apache Tomcat v5.5 | New...

Build Descriptor Options

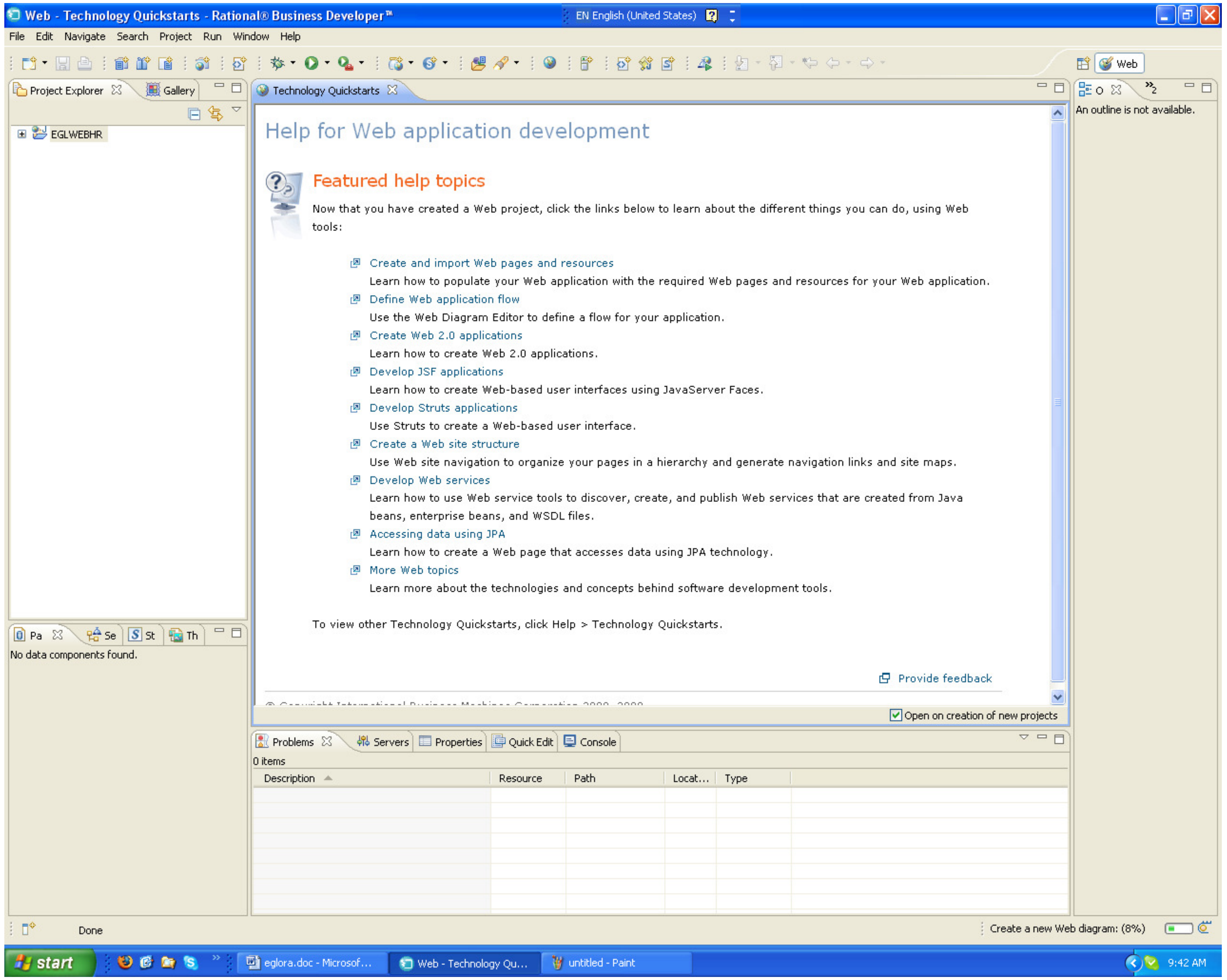
EGL build descriptors contain options which customize the process of generating EGL to Java or COBOL

- [✓ Create a new build descriptor] Options...
- Use the build descriptor specified in the EGL preferences
- Select an existing build descriptor
- Do not create a build descriptor

☐ Show Advanced Settings

Lab – Create a New Project

- Your EGLWEB400 project is created
- Close “Technology Quickstart” tab



Help for Web application development

Featured help topics

Now that you have created a Web project, click the links below to learn about the different things you can do, using Web tools:

- Create and import Web pages and resources
  Learn how to populate your Web application with the required Web pages and resources for your Web application.
- Define Web application flow
  Use the Web Diagram Editor to define a flow for your application.
- Create Web 2.0 applications
  Learn how to create Web 2.0 applications.
- Develop JSF applications
  Learn how to create Web-based user interfaces using JavaServer Faces.
- Develop Struts applications
  Use Struts to create a Web-based user interface.
- Create a Web site structure
  Use Web site navigation to organize your pages in a hierarchy and generate navigation links and site maps.
- Develop Web services
  Learn how to use Web service tools to discover, create, and publish Web services that are created from Java beans, enterprise beans, and WSDL files.
- Accessing data using JPA
  Learn how to create a Web page that accesses data using JPA technology.
- More Web topics
  Learn more about the technologies and concepts behind software development tools.

To view other Technology Quickstarts, click Help > Technology Quickstarts.

Provide feedback

[✓] Open on creation of new projects

UNIT

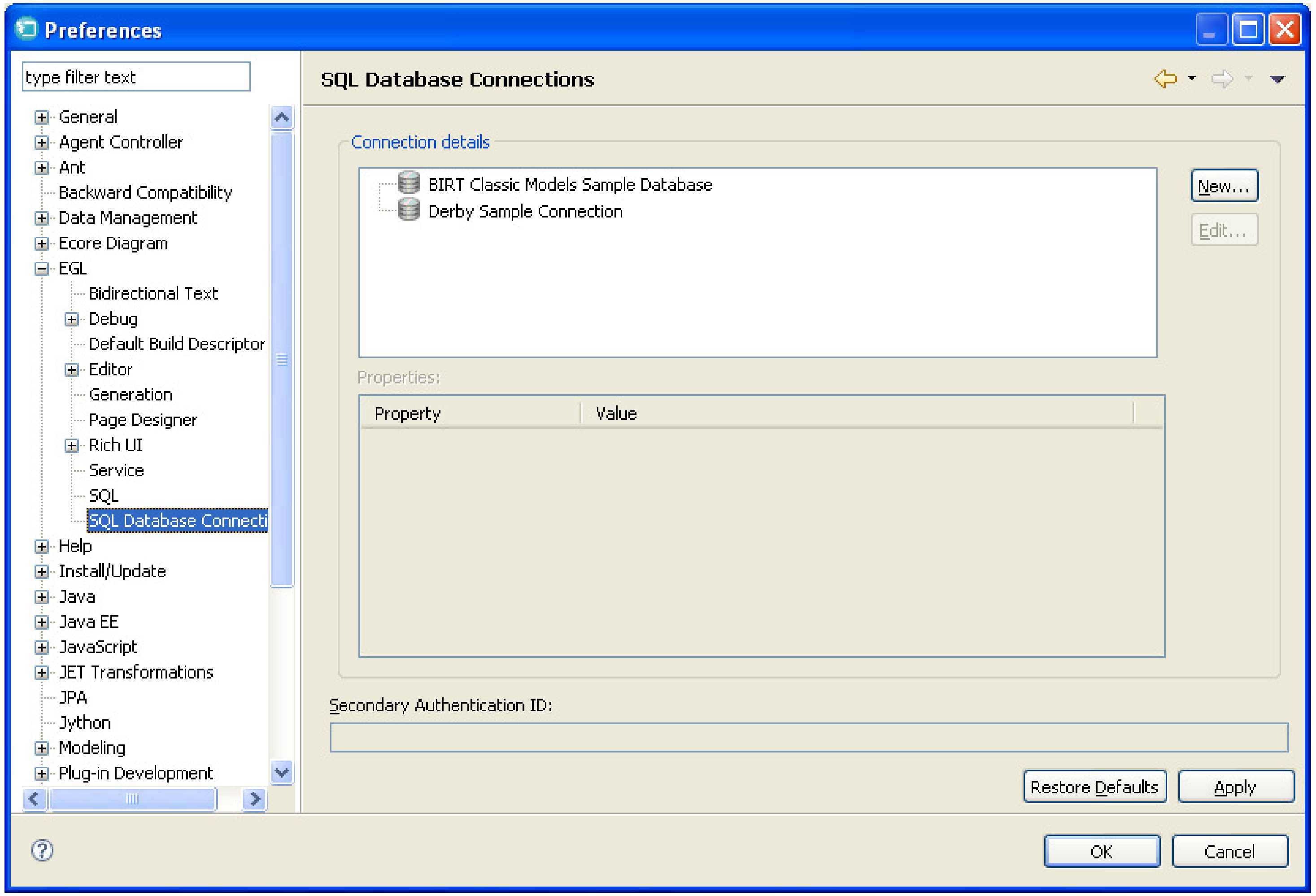
System i Connections

Topics:

- Creating a New Project
- **Creating a DB2/400 Connection**
- Defining a DB2/400 Runtime Data Source
- Adding JT400 Toolkit to Your Project

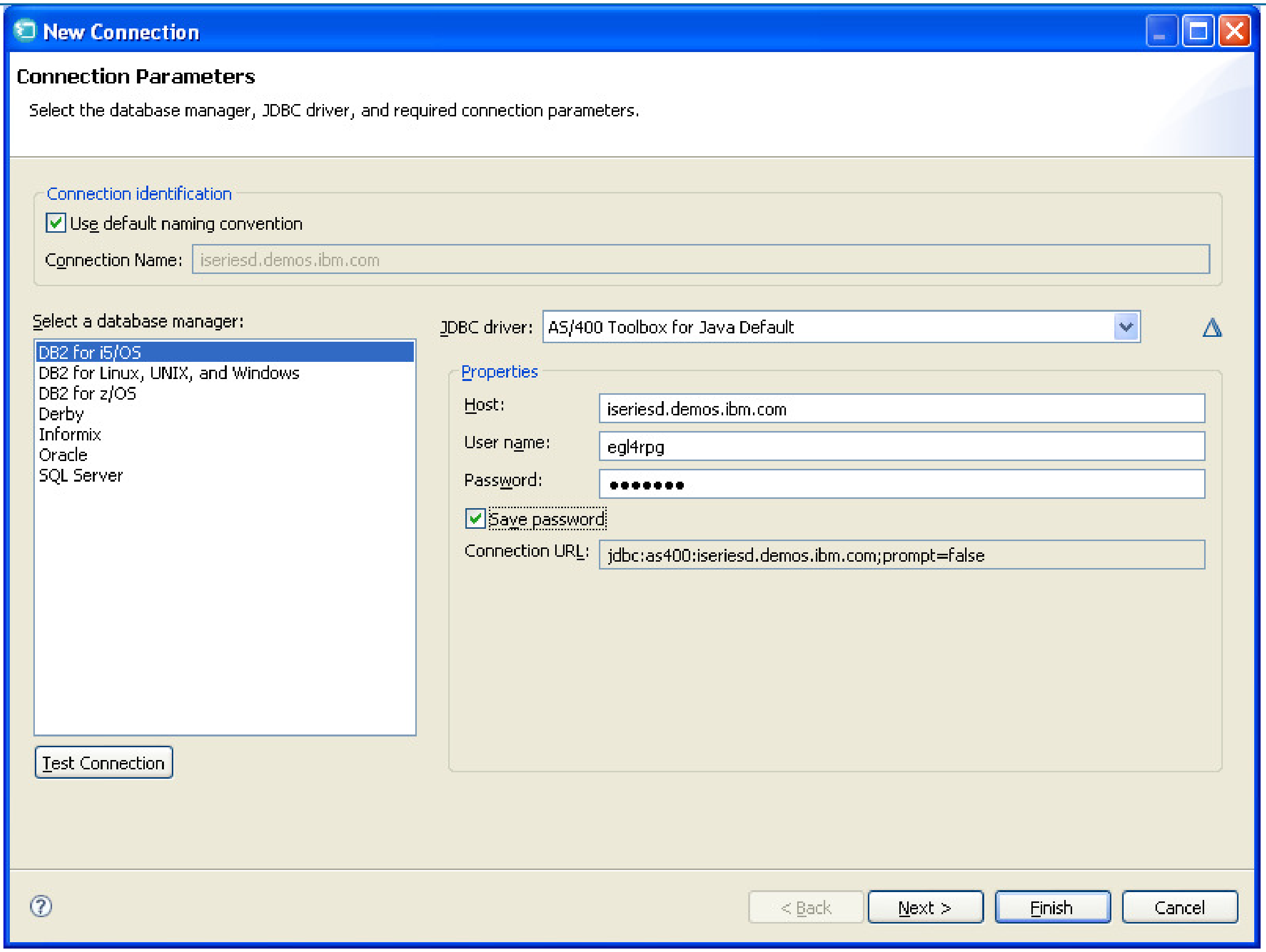
⏭ Lab – Create a DB2/400 Connection

- From Window → Preference
- Expand “EGL” folder
- Click “SQL Database Connection
- Click “New”



⏭ Lab – Create a DB2/400 Connection

- Select a database manager: DB2 for i5/OS
- JDBC driver: AS/400 toolbox for Java Default
- Host: iseries.demos.ibm.com
- User name: egl4rpg
- Password: egl4you
- Check “Save password”
- Click “Test Connection”
- Click OK



Connection Parameters

Select the database manager, JDBC driver, and required connection parameters.

Connection identification

☑ Use default naming convention

Connection Name: iseriesd.demos.ibm.com

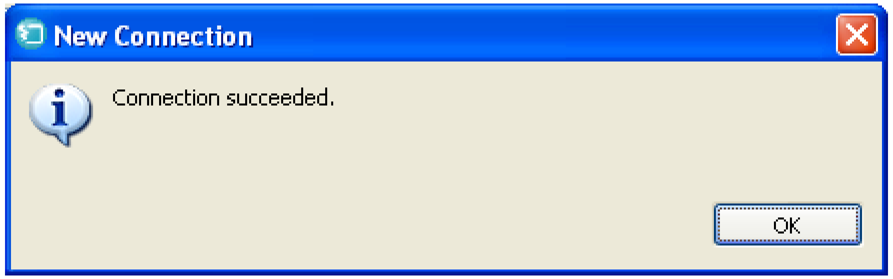
Select a database manager:

- DB2 for i5/OS
- DB2 for Linux, UNIX, and Windows
- DB2 for z/OS
- Derby
- Informix
- Oracle
- SQL Server

JDBC driver: AS/400 Toolbox for Java Default

Properties

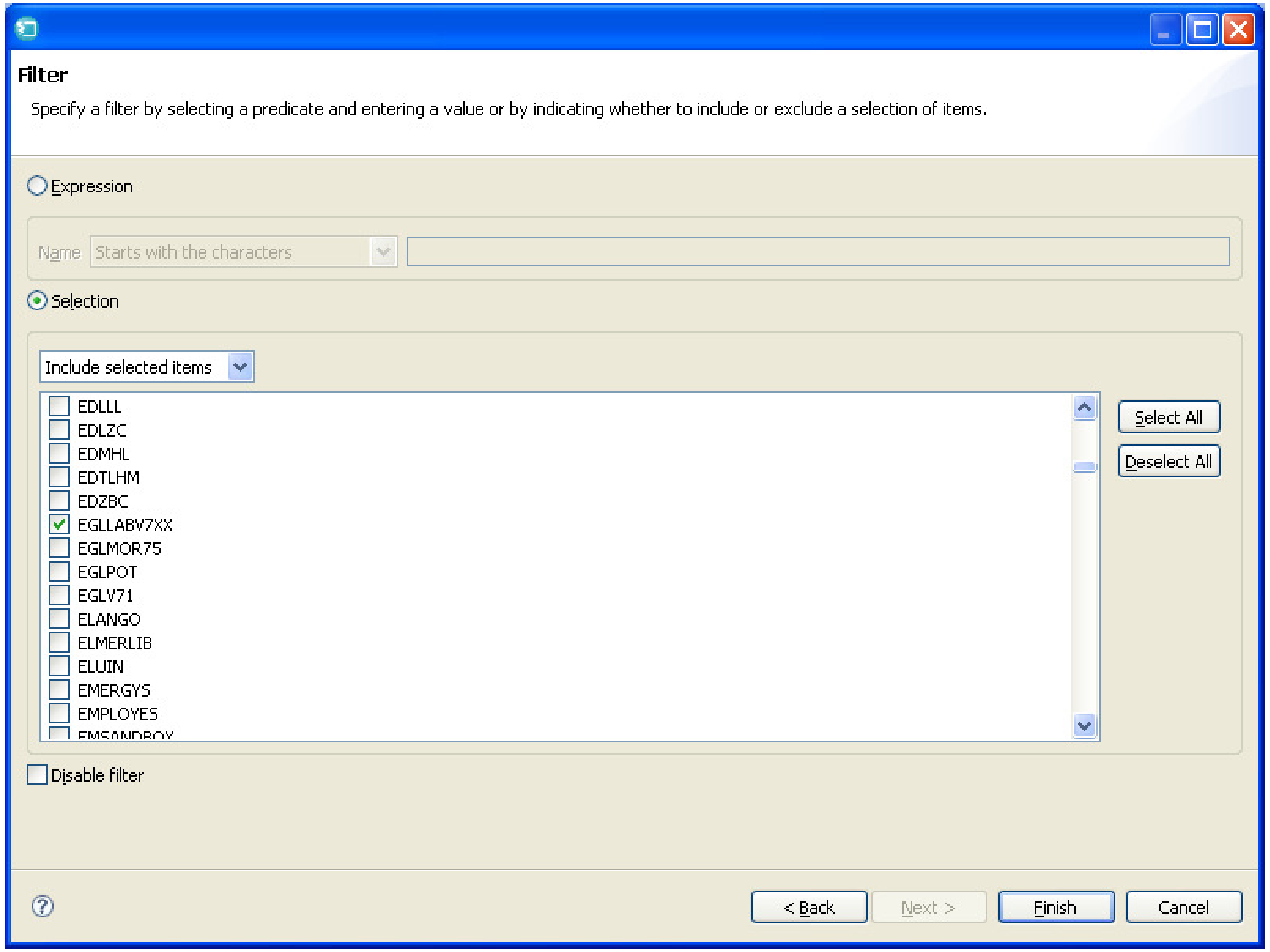
Host:	iseriesd.demos.ibm.com
User name:	egl4rpg
Password:	●●●●●●●
☑ Save password	
Connection URL:	jdbc:as400:iseriesd.demos.ibm.com;prompt=false



Connection succeeded.

Lab – Create a DB2/400 Connection

- **Un**-check “Disable Filter”
- Click “Selection”
- Scroll down and check “EGLLABV7XX” library
- Click “Finish”



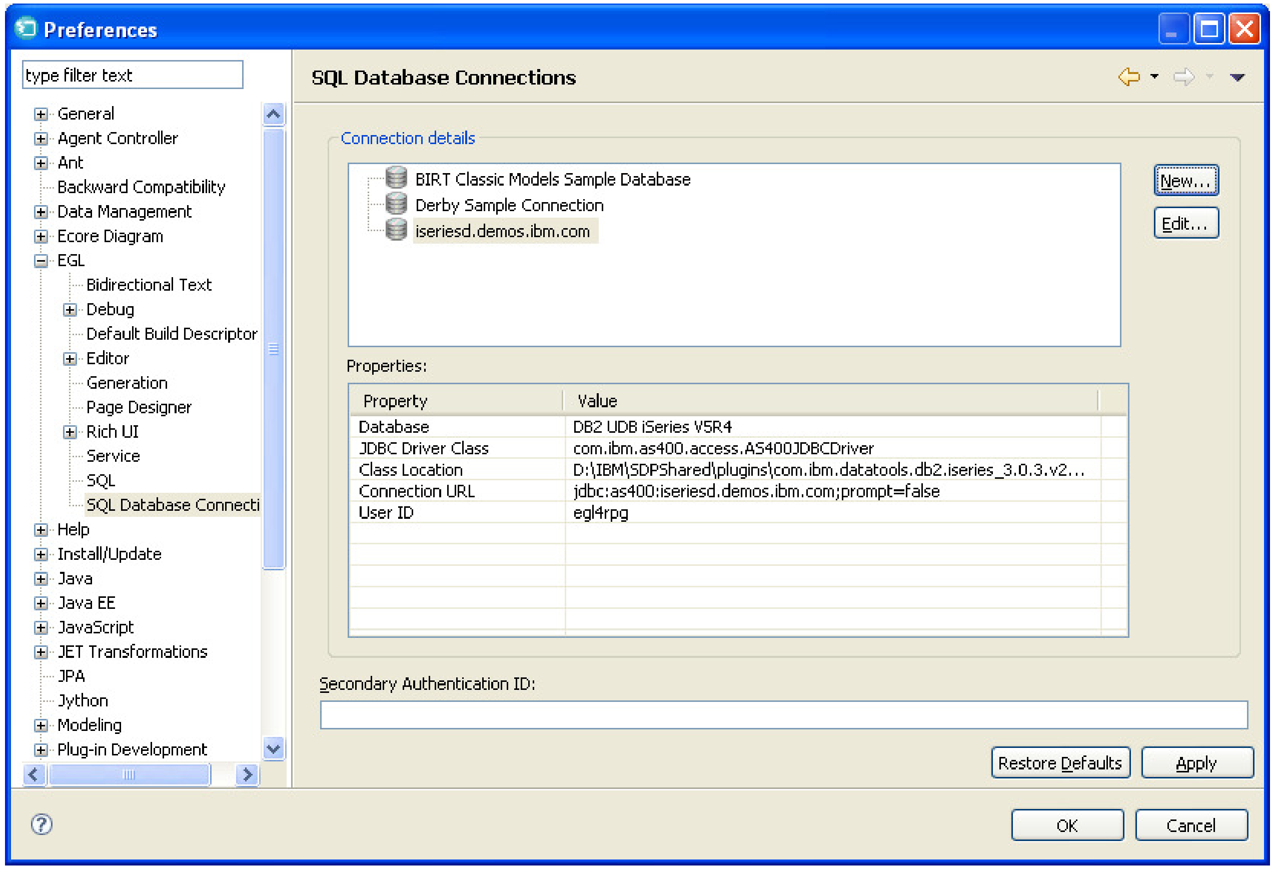
Filter

Specify a filter by selecting a predicate and entering a value or by indicating whether to include or exclude a selection of items.

- ○ Expression
- Name: Starts with the characters
- ● Selection
- Include selected items
- ☐ EDLLL
- ☐ EDLZC
- ☐ EDMHL
- ☐ EDTLHM
- ☐ EDZBC
- ☑ EGLLABV7XX
- ☐ EGLMOR75
- ☐ EGLPOT
- ☐ EGLV71
- ☐ ELANGO
- ☐ ELMERLIB
- ☐ ELUIN
- ☐ EMERGYS
- ☐ EMPLOYES
- ☐ EMSANDBOX
- ☐ Disable filter

⏭ Lab – Create a DB2/400 Connection

- Click “Apply”
- Click OK



Property	Value
Database	DB2 UDB iSeries V5R4
JDBC Driver Class	com.ibm.as400.access.AS400JDBCDriver
Class Location	D:\IBM\SDPShared\plugins\com.ibm.datatools.db2.iseries_3.0.3.v2...
Connection URL	jdbc:as400:iseriesd.demos.ibm.com;prompt=false
User ID	egl4rpg

UNIT

System i Connections

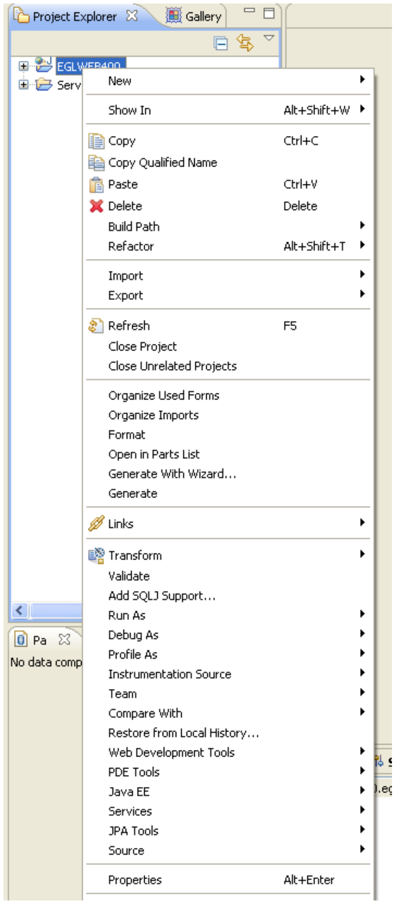
Topics:

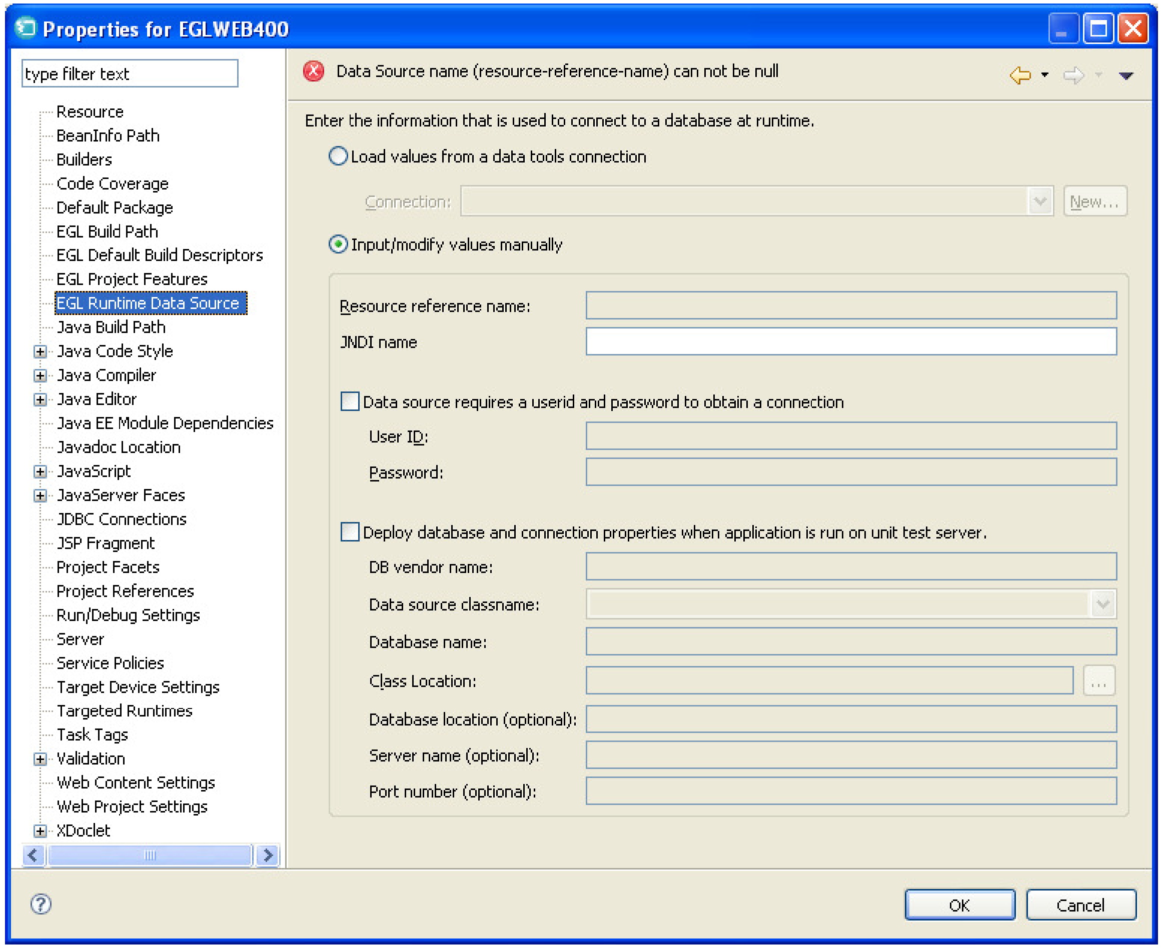
- Creating a New Project
- Creating a DB2/400 Connection
- **Defining a DB2/400 Runtime Data Source**
- Adding JT400 Toolkit to Your Project

⏭ Lab – Define a DB2/400 Runtime Data Source

- Right-click over EGLWEB400 project and select Properties

- Click EGL Runtime Data Source





Data Source name (resource-reference-name) can not be null

Enter the information that is used to connect to a database at runtime.

○ Load values from a data tools connection

Connection:

● Input/modify values manually

Resource reference name:

JNDI name

☐ Data source requires a userid and password to obtain a connection

User ID:

Password:

☐ Deploy database and connection properties when application is run on unit test server.

DB vendor name:

Data source classname:

Database name:

Class Location:

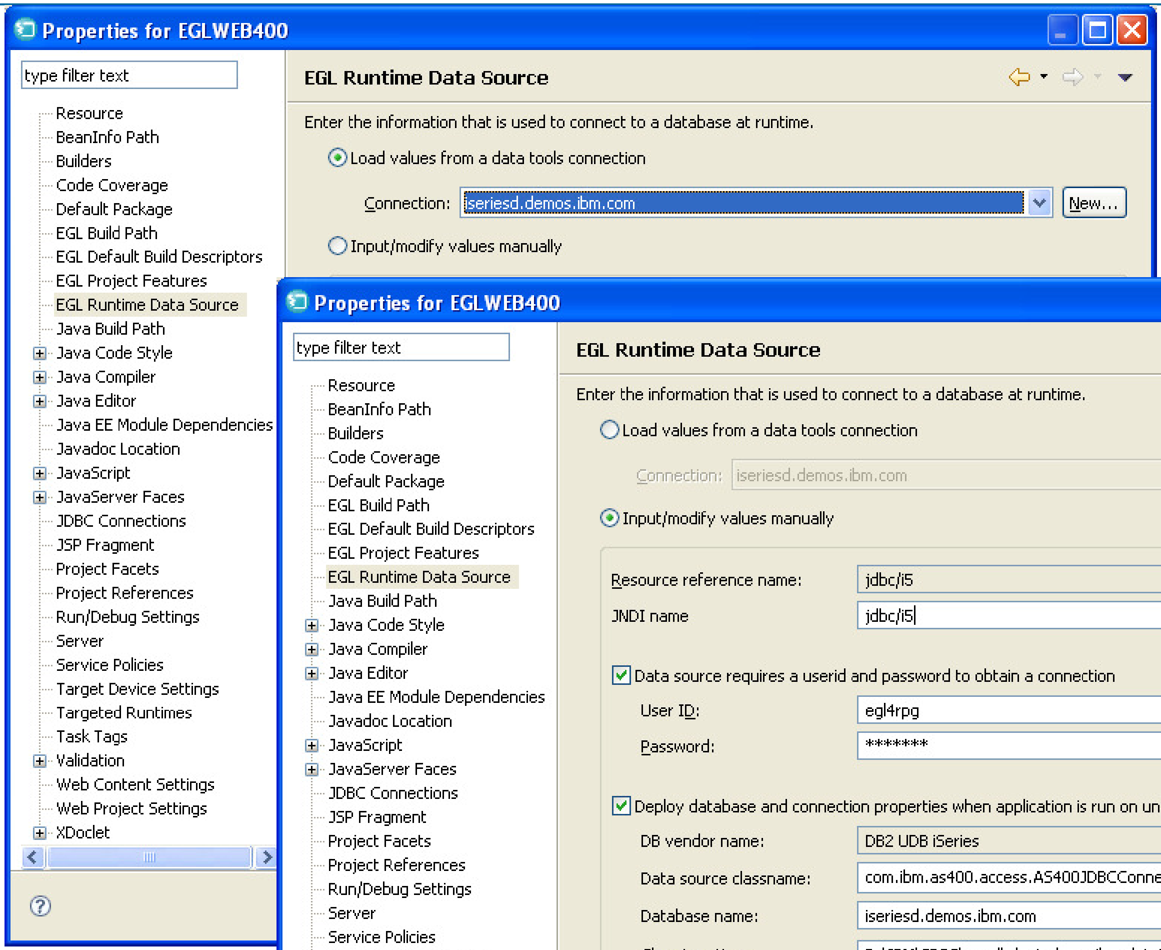
Database location (optional):

Server name (optional):

Port number (optional):

Lab – Define a DB2/400 Runtime Data Source

- Click "Load values from a data tools connection"
- Select iseriesd.demos.ibm.com from the pull down-list
- Click input/modify values manually
- Change the JNDI name to "jdbc/i5"
- Click OK



Properties for EGLWEB400

type filter text

Resource
BeanInfo Path
Builders
Code Coverage
Default Package
EGL Build Path
EGL Default Build Descriptors
EGL Project Features
EGL Runtime Data Source
Java Build Path
Java Code Style
Java Compiler
Java Editor
Java EE Module Dependencies
Javadoc Location
JavaScript
JavaServer Faces
JDBC Connections
JSP Fragment
Project Facets
Project References
Run/Debug Settings
Server
Service Policies
Target Device Settings
Targeted Runtimes
Task Tags
Validation
Web Content Settings
Web Project Settings
XDoclet

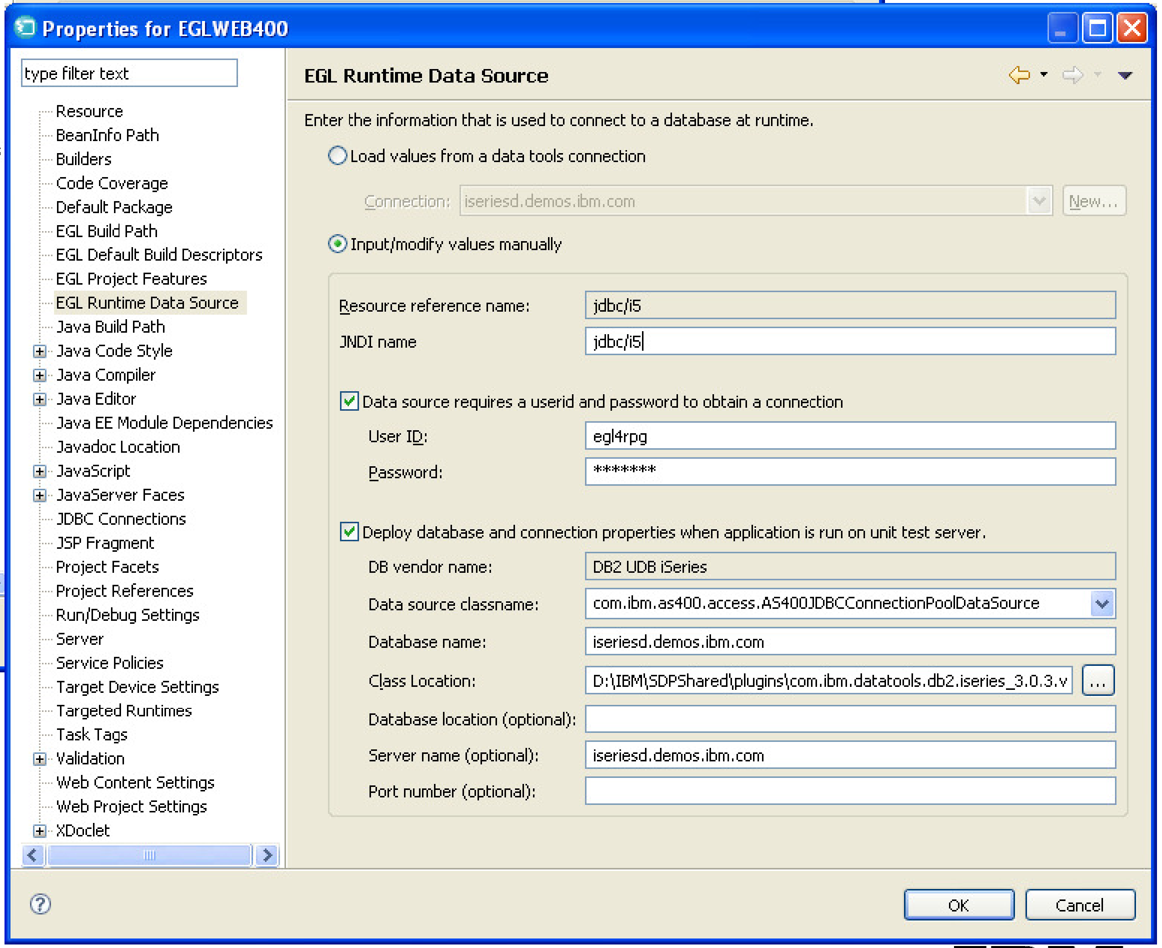
EGL Runtime Data Source

Enter the information that is used to connect to a database at runtime.

(●) Load values from a data tools connection

Connection: iseriesd.demos.ibm.com | New...

( ) Input/modify values manually



Properties for EGLWEB400

type filter text

Resource
BeanInfo Path
Builders
Code Coverage
Default Package
EGL Build Path
EGL Default Build Descriptors
EGL Project Features
EGL Runtime Data Source
Java Build Path
Java Code Style
Java Compiler
Java Editor
Java EE Module Dependencies
Javadoc Location
JavaScript
JavaServer Faces
JDBC Connections
JSP Fragment
Project Facets
Project References
Run/Debug Settings
Server
Service Policies
Target Device Settings
Targeted Runtimes
Task Tags
Validation
Web Content Settings
Web Project Settings
XDoclet

EGL Runtime Data Source

Enter the information that is used to connect to a database at runtime.

( ) Load values from a data tools connection

Connection: iseriesd.demos.ibm.com | New...

(●) Input/modify values manually

Resource reference name:	jdbc/i5
JNDI name	jdbc/i5

[✓] Data source requires a userid and password to obtain a connection

User ID:	egl4rpg
Password:	*******

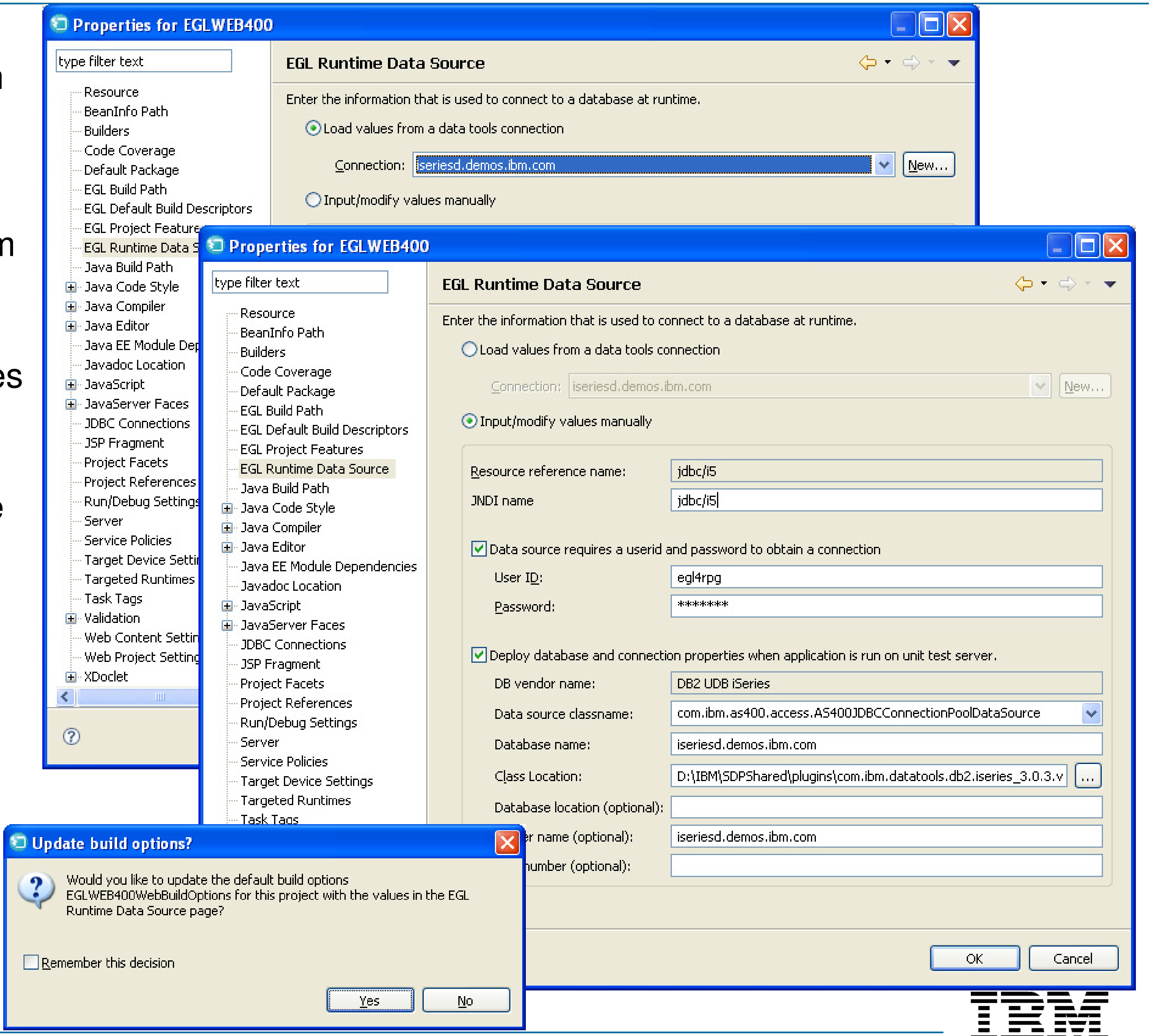
[✓] Deploy database and connection properties when application is run on unit test server.

DB vendor name:	DB2 UDB iSeries
Data source classname:	com.ibm.as400.access.AS400JDBCConnectionPoolDataSource
Database name:	iseriesd.demos.ibm.com
Class Location:	D:\IBM\SDPShared\plugins\com.ibm.datatools.db2.iseries_3.0.3.v
Database location (optional):	
Server name (optional):	iseriesd.demos.ibm.com
Port number (optional):	

OK | Cancel

Lab – Define a DB2/400 Runtime Data Source

- Click "Load values from a data tools connection"
- Select iseriesd.demos.ibm.com from the pull down-list
- Click input/modify values manually
- Change the JNDI name to "jdbc/i5"
- Click OK
- Click "Yes" on Update build option?



EGL Runtime Data Source

Enter the information that is used to connect to a database at runtime.

[✓ Load values from a data tools connection]

Connection: iseriesd.demos.ibm.com

Input/modify values manually

EGL Runtime Data Source

Enter the information that is used to connect to a database at runtime.

Load values from a data tools connection

Connection: iseriesd.demos.ibm.com

[✓ Input/modify values manually]

Resource reference name:	jdbc/i5
JNDI name	jdbc/i5

[✓] Data source requires a userid and password to obtain a connection

User ID:	egl4rpg
Password:	*******

[✓] Deploy database and connection properties when application is run on unit test server.

DB vendor name:	DB2 UDB iSeries
Data source classname:	com.ibm.as400.access.AS400JDBCConnectionPoolDataSource
Database name:	iseriesd.demos.ibm.com
Class Location:	D:\IBM\SDPShared\plugins\com.ibm.datatools.db2.iseries_3.0.3.v
Database location (optional):	
r name (optional):	iseriesd.demos.ibm.com
umber (optional):	

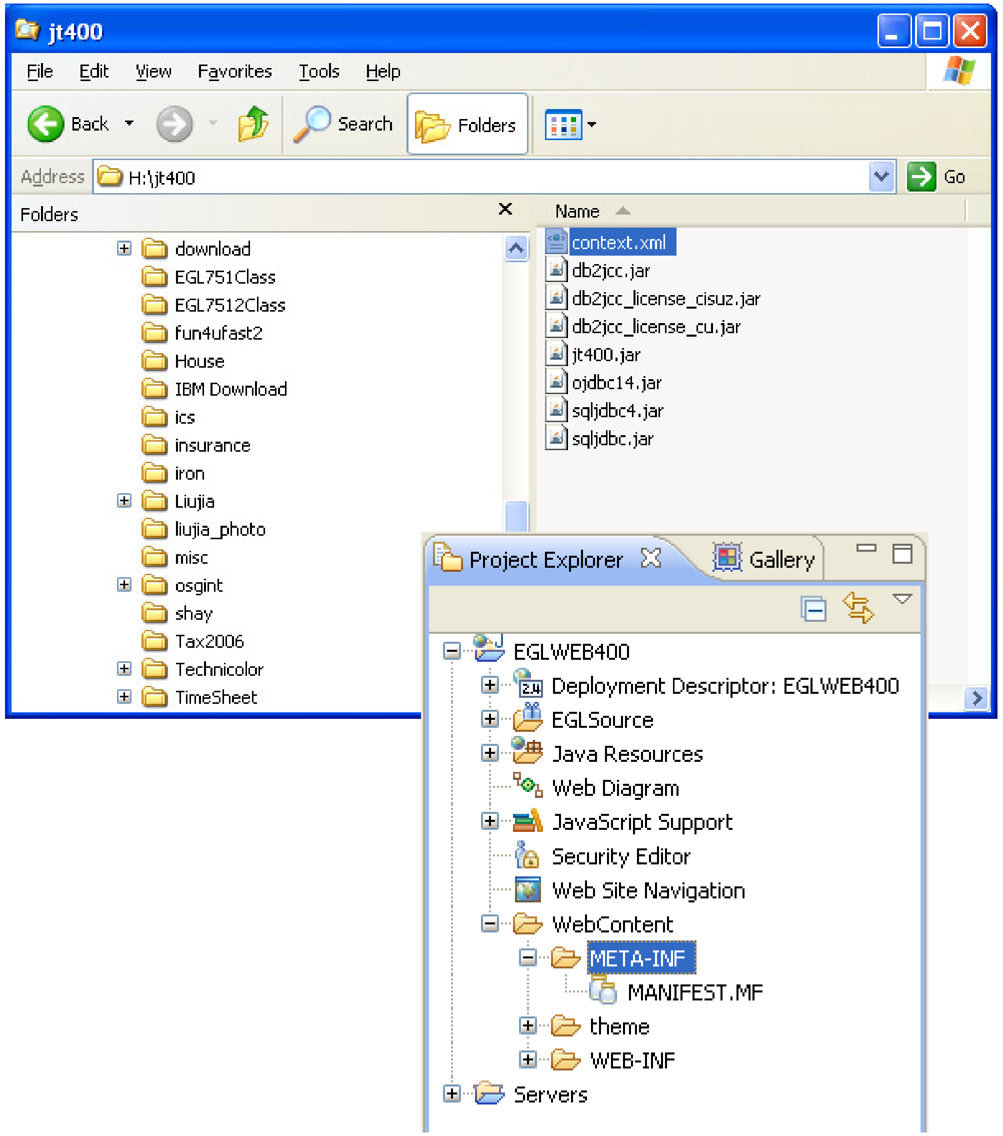
Update build options?

Would you like to update the default build options EGLWEB400WebBuildOptions for this project with the values in the EGL Runtime Data Source page?

[ ] Remember this decision

⏭ Lab – Define a DB2/400 Runtime Data Source

- Copy context.xml from jt400 folder
- Paste it to EGLWEB400 → WebContent → META-INF folder
- The context.xml file is an optional file which contains a <Context> tag (Context Fragment) for a single Tomcat web application. This can be used to define certain behaviours for your application, JNDI resources and other settings.
- The context.xml file was introduced in Tomcat 5, to remove Context settings from the server.xml file.
- The Context Fragment can be embedded in server.xml, placed in the <CATALINA>/conf folder, or placed inside each application in the META-INF folder in a file called context.xml. The META-INF method is preferred as this means changes to the context don't require Tomcat to be restarted. You can restart your application by modifying web.xml when automatic reloading is active, or reloading manually with the Tomcat Manager.
- Your context.xml file should be installed in the META-INF folder of your application, as META-INF/context.xml.
- Switching to the use of a context.xml file removes all dependencies from server.xml, making the application much more portable and easier to deploy.



UNIT

System i Connections

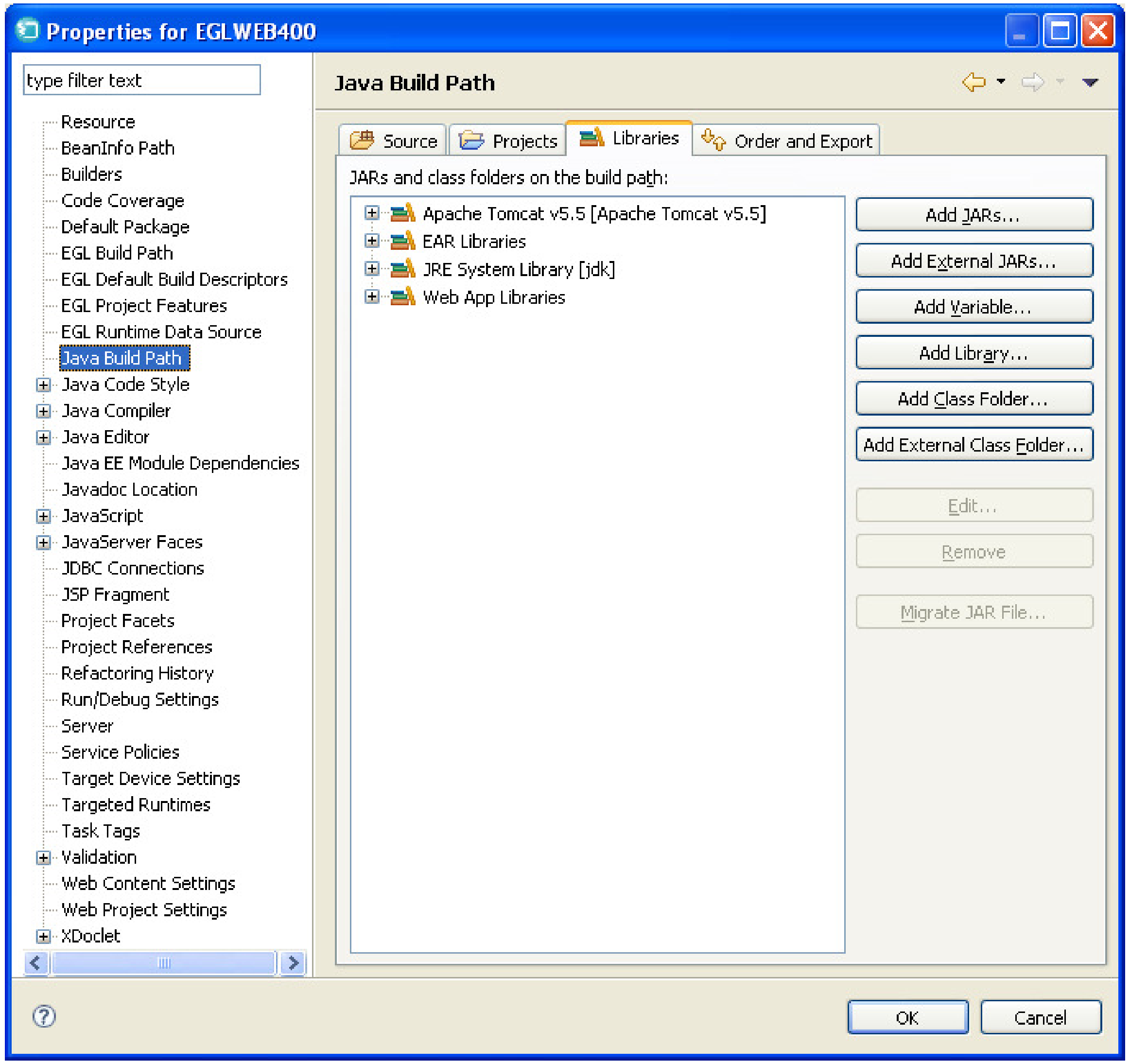
Topics:

- Creating a New Project
- Creating a DB2/400 Connection
- Defining a DB2/400 Runtime Data Source
- **Adding JT400 Toolkit to Your Project**

⏭ Lab – Add JT400 Toolkit to your Project

Access Objects (Programs, Files, Data Areas, Data Queues) on an iSeries box requires the JT400 toolkit in your build path.

- Click "Java Build Path" on "Properties for EGLWEB400"
- Click "Libraries" tab
- Click "Add External JARs"

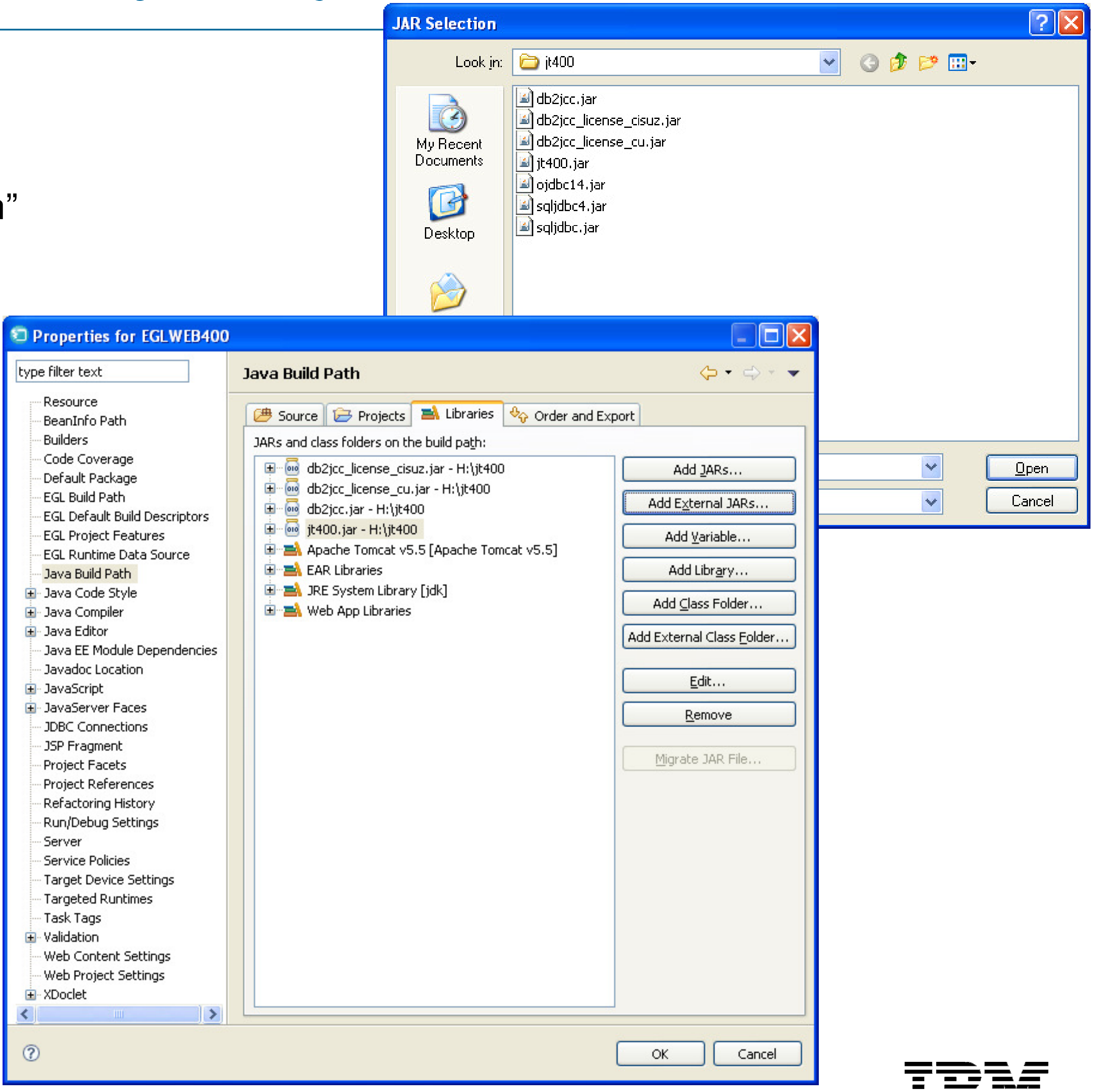


JARs and class folders on the build path:

- Apache Tomcat v5.5 [Apache Tomcat v5.5]
- EAR Libraries
- JRE System Library [jdk]
- Web App Libraries

⏭ Lab – Add JT400 Toolkit to your Project

- Click “Add External JARs”
- Open jt400 folder
- Select “db2jcc.jar” Click “Open”
- Repeat to add:
  - Db2jcc_license_cisuz.jar
  - Db2jcc_license_cu.jar
  - Jt400.jar
- Click “OK”



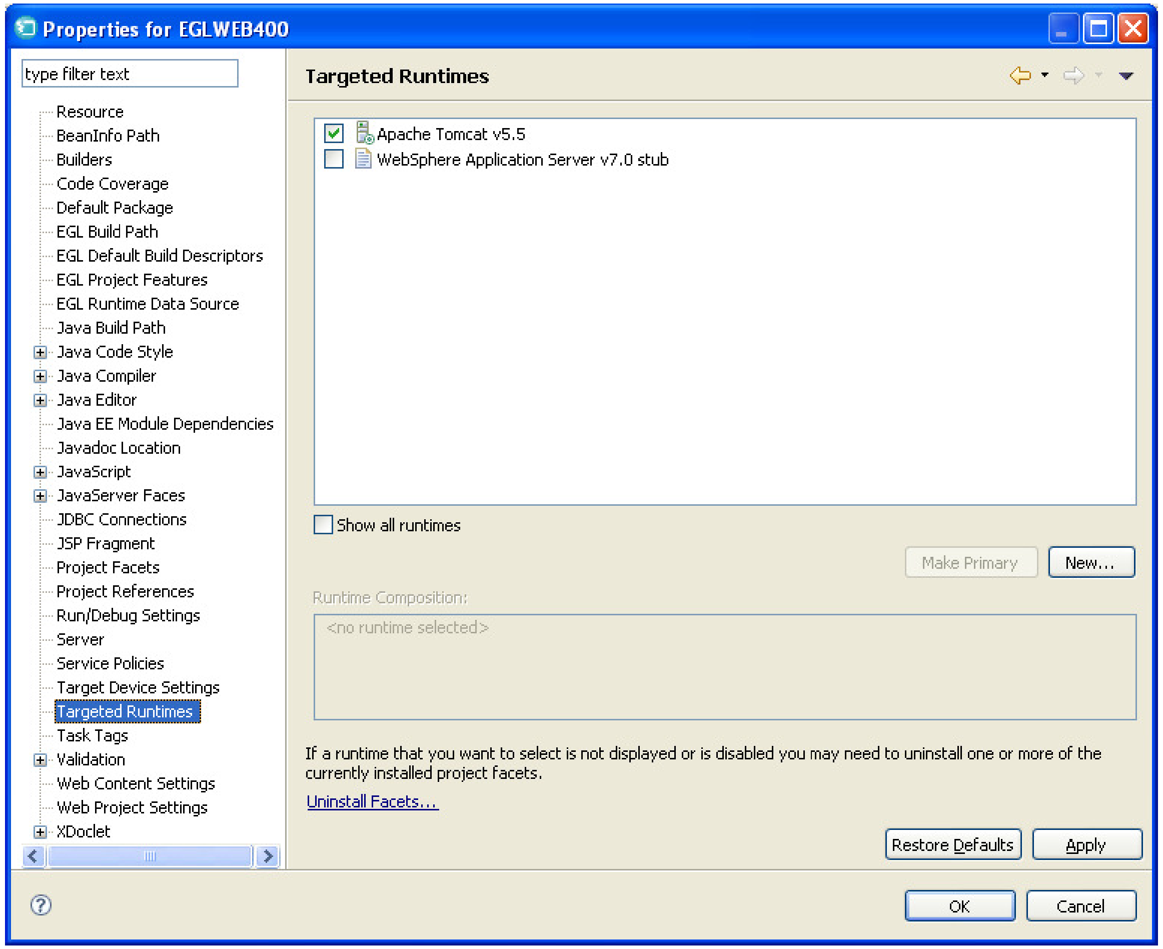
db2jcc.jar
db2jcc_license_cisuz.jar
db2jcc_license_cu.jar
jt400.jar
ojdbc14.jar
sqljdbc4.jar
sqljdbc.jar

JARs and class folders on the build path:

db2jcc_license_cisuz.jar - H:\jt400
db2jcc_license_cu.jar - H:\jt400
db2jcc.jar - H:\jt400
jt400.jar - H:\jt400
Apache Tomcat v5.5 [Apache Tomcat v5.5]
EAR Libraries
JRE System Library [jdk]
Web App Libraries

Lab – Setup Targeted Runtime

- Click “Targeted Runtime” on “Properties for EGLWEB400”
- Check “Apache Tomcat 5.5”
- Click “Apply”
- Click “Yes” if asked to “Update build options”
- Click “OK”



Properties for EGLWEB400

type filter text

Resource
BeanInfo Path
Builders
Code Coverage
Default Package
EGL Build Path
EGL Default Build Descriptors
EGL Project Features
EGL Runtime Data Source
Java Build Path
Java Code Style
Java Compiler
Java Editor
Java EE Module Dependencies
Javadoc Location
JavaScript
JavaServer Faces
JDBC Connections
JSP Fragment
Project Facets
Project References
Run/Debug Settings
Server
Service Policies
Target Device Settings
Targeted Runtimes
Task Tags
Validation
Web Content Settings
Web Project Settings
XDoclet

Targeted Runtimes

- [✓] Apache Tomcat v5.5
- [ ] WebSphere Application Server v7.0 stub

[ ] Show all runtimes

Make Primary | New...

Runtime Composition:

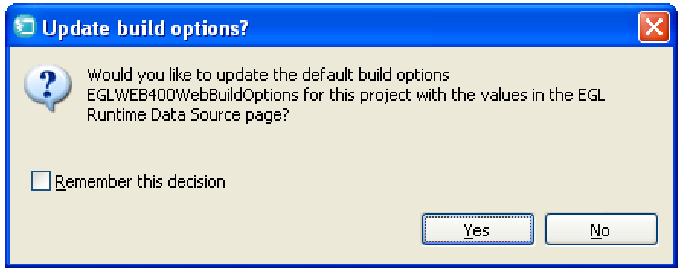
<no runtime selected>

If a runtime that you want to select is not displayed or is disabled you may need to uninstall one or more of the currently installed project facets.

Uninstall Facets...

Restore Defaults | Apply

OK | Cancel



Update build options?

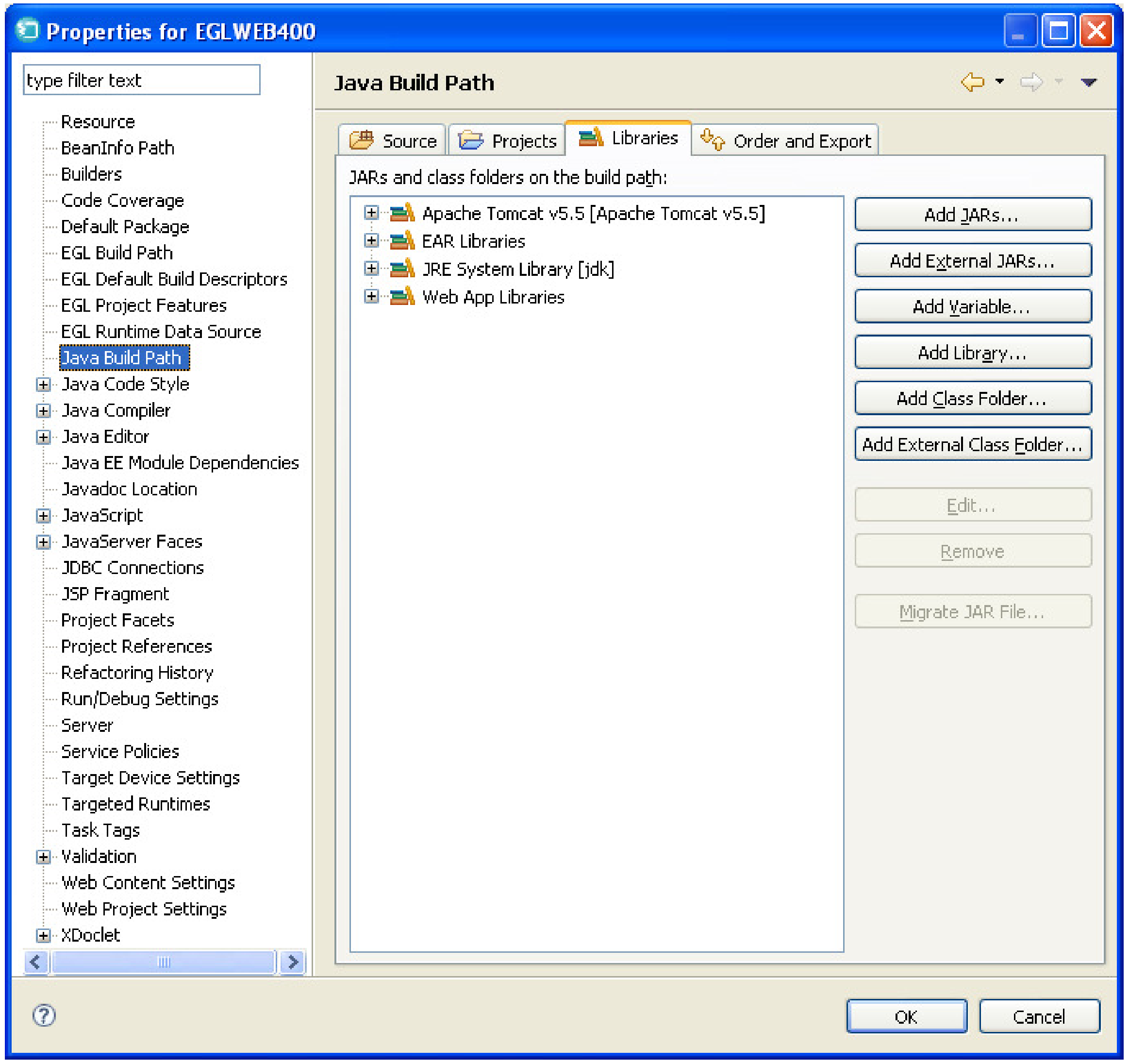
Would you like to update the default build options EGLWEB400WebBuildOptions for this project with the values in the EGL Runtime Data Source page?

[ ] Remember this decision

Yes | No

⏭ Lab – Setup jt400.jar

- Click “Java Build Path” on “Properties for EGLWEB400”
- Click “Libraries” tab
- Click Add External JARs”


Properties for EGLWEB400

type filter text

Resource
BeanInfo Path
Builders
Code Coverage
Default Package
EGL Build Path
EGL Default Build Descriptors
EGL Project Features
EGL Runtime Data Source
Java Build Path
Java Code Style
Java Compiler
Java Editor
Java EE Module Dependencies
Javadoc Location
JavaScript
JavaServer Faces
JDBC Connections
JSP Fragment
Project Facets
Project References
Refactoring History
Run/Debug Settings
Server
Service Policies
Target Device Settings
Targeted Runtimes
Task Tags
Validation
Web Content Settings
Web Project Settings
XDoclet

Java Build Path

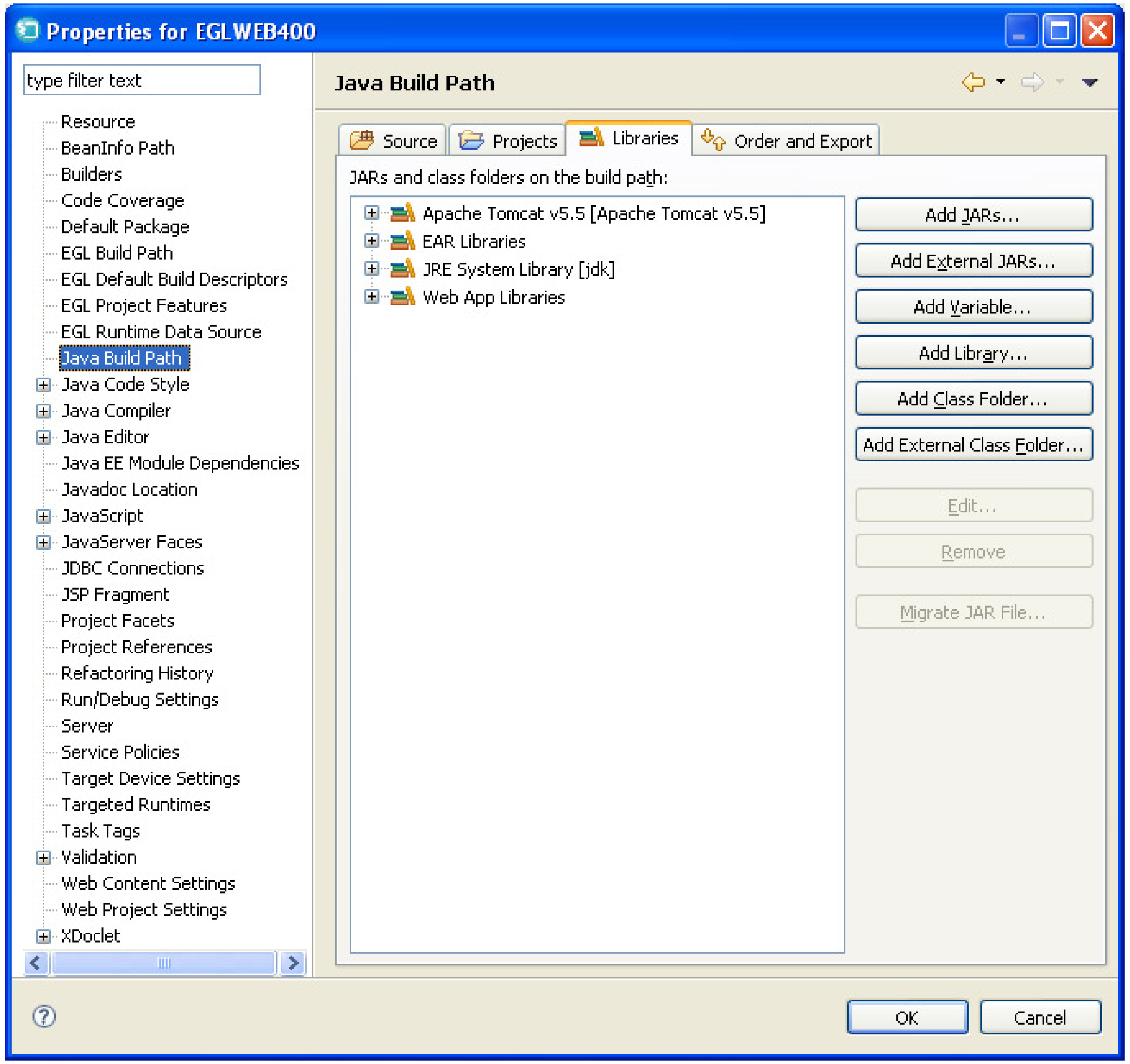
Source | Projects | Libraries | Order and Export

JARs and class folders on the build path:

- Apache Tomcat v5.5 [Apache Tomcat v5.5]
- EAR Libraries
- JRE System Library [jdk]
- Web App Libraries

Lab – Setup jt400.jar

- Click “Java Build Path” on “Properties for EGLWEB400”
- Click “Libraries” tab
- Click Add External JARs”



Properties for EGLWEB400

type filter text

- Resource
- BeanInfo Path
- Builders
- Code Coverage
- Default Package
- EGL Build Path
- EGL Default Build Descriptors
- EGL Project Features
- EGL Runtime Data Source
- Java Build Path
- Java Code Style
- Java Compiler
- Java Editor
- Java EE Module Dependencies
- Javadoc Location
- JavaScript
- JavaServer Faces
- JDBC Connections
- JSP Fragment
- Project Facets
- Project References
- Refactoring History
- Run/Debug Settings
- Server
- Service Policies
- Target Device Settings
- Targeted Runtimes
- Task Tags
- Validation
- Web Content Settings
- Web Project Settings
- XDoclet

Java Build Path

Source | Projects | Libraries | Order and Export

JARs and class folders on the build path:

- Apache Tomcat v5.5 [Apache Tomcat v5.5]
- EAR Libraries
- JRE System Library [jdk]
- Web App Libraries

UNIT

EGL/Web QuickStart

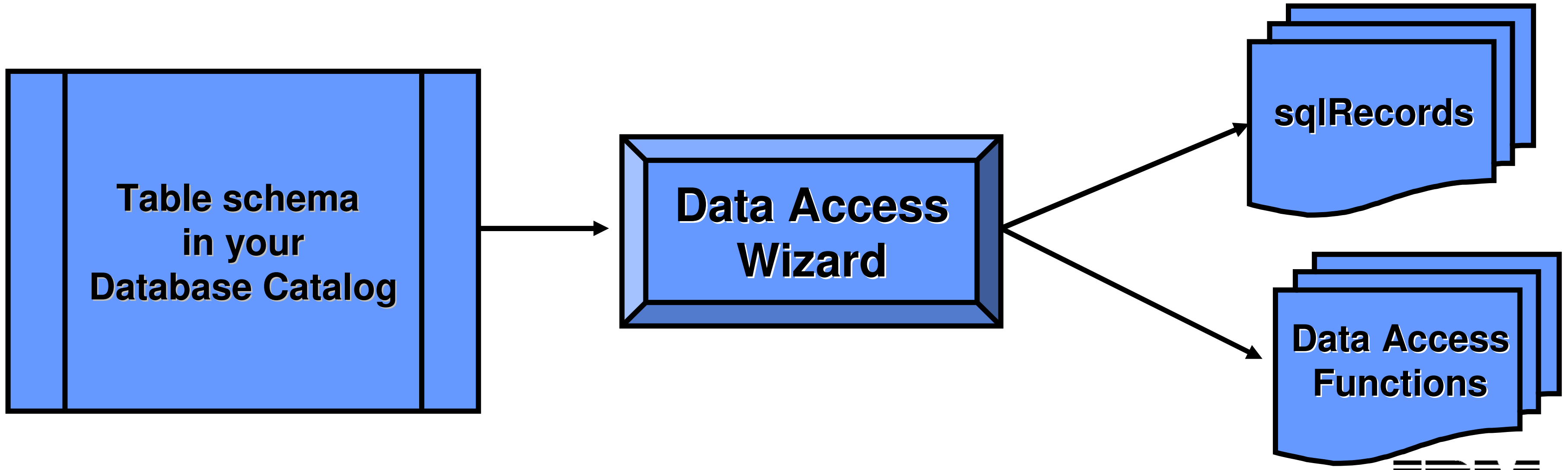
Topics:

- **Importing Database Schema**
- Creating Custom Business Logic
- Creating the U.I. Model
- Creating Dynamic Content Web Pages

Data Access Wizard

To simplify data access EGL provides custom record types, with “properties” that enable generation of native database and file I/O call APIs. You can write these custom records – or – allow RBD’s tooling to build or derive them from the relational tables and views you want to access.

- To get you jump-started on your EGL/SQL statements, the RBD tooling creates design pattern functions, for reading and writing to your database.
- You can create your own custom EGL data access records, design patterns, functions and SQL statements. But for this simple web application, let’s see how far the wizard-generated code gets us.



Data Access Application Wizard – Steps for the Upcoming Workshop

These steps are what you WILL be doing, to import your database:

- Create a new Project
- Launch the EGL Data Access Application Wizard
- Define a connection to your database
- Specify any table filtering
- Select the tables you want to import
- Verify/Specify search keys
- Generate
- View and customize Results

Before starting this process, you must have completed the Workspace **Preferences** setting for your **SQL** generation as described on the slide: **Workspace Preferences – Customizing the Workbench – Modify SQL Preferences**

⏭ Launch the EGL Data Access Application Wizard

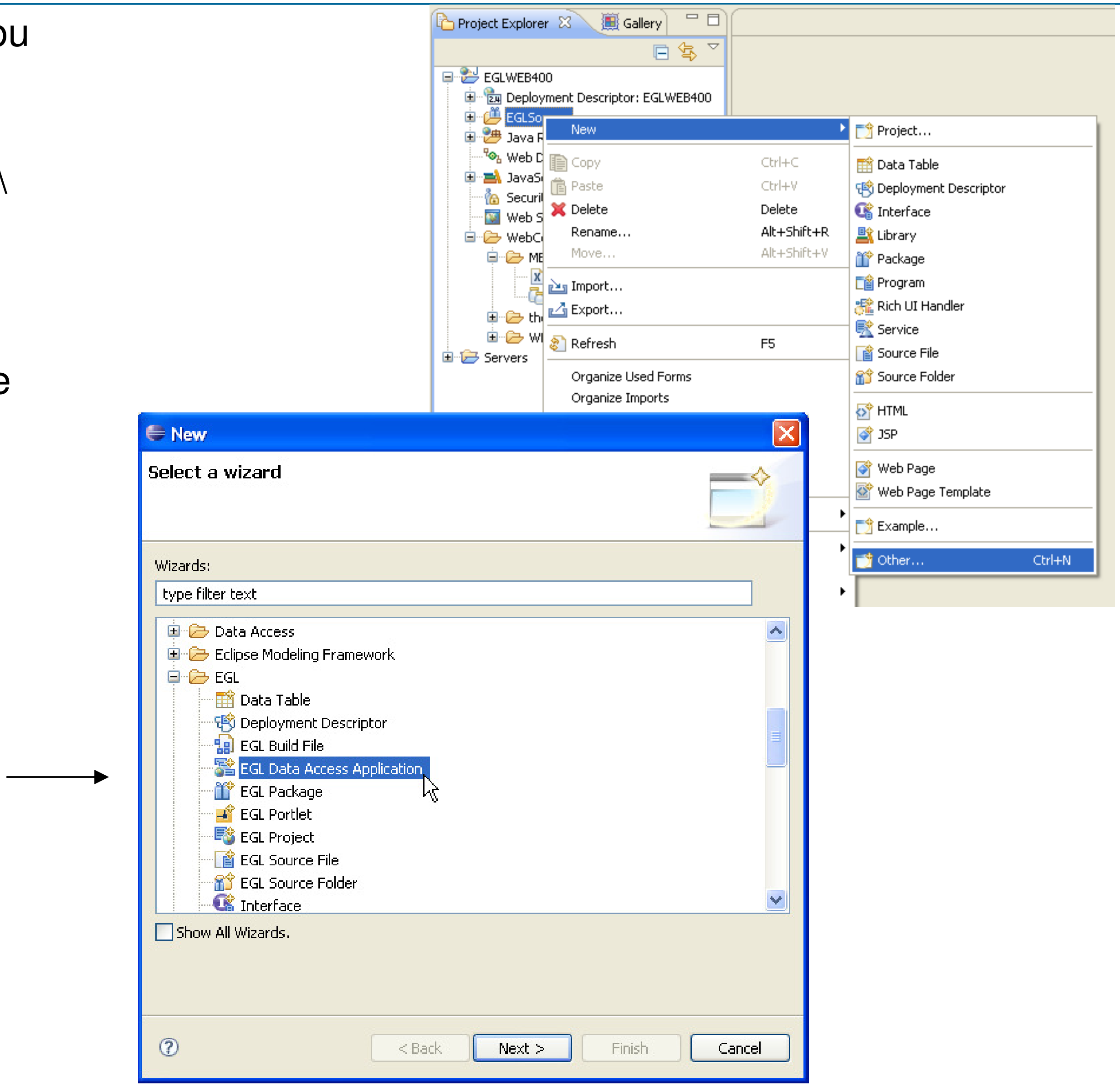
Like creating all new resources, you start from Project Explorer:

- **Right-click** over \EGLSource\\ and select:

  New ▶ Other

- Scroll down to, and expand the **EGL** folder and select:
  - **EGL Data Access Application**

- Click **Next >**



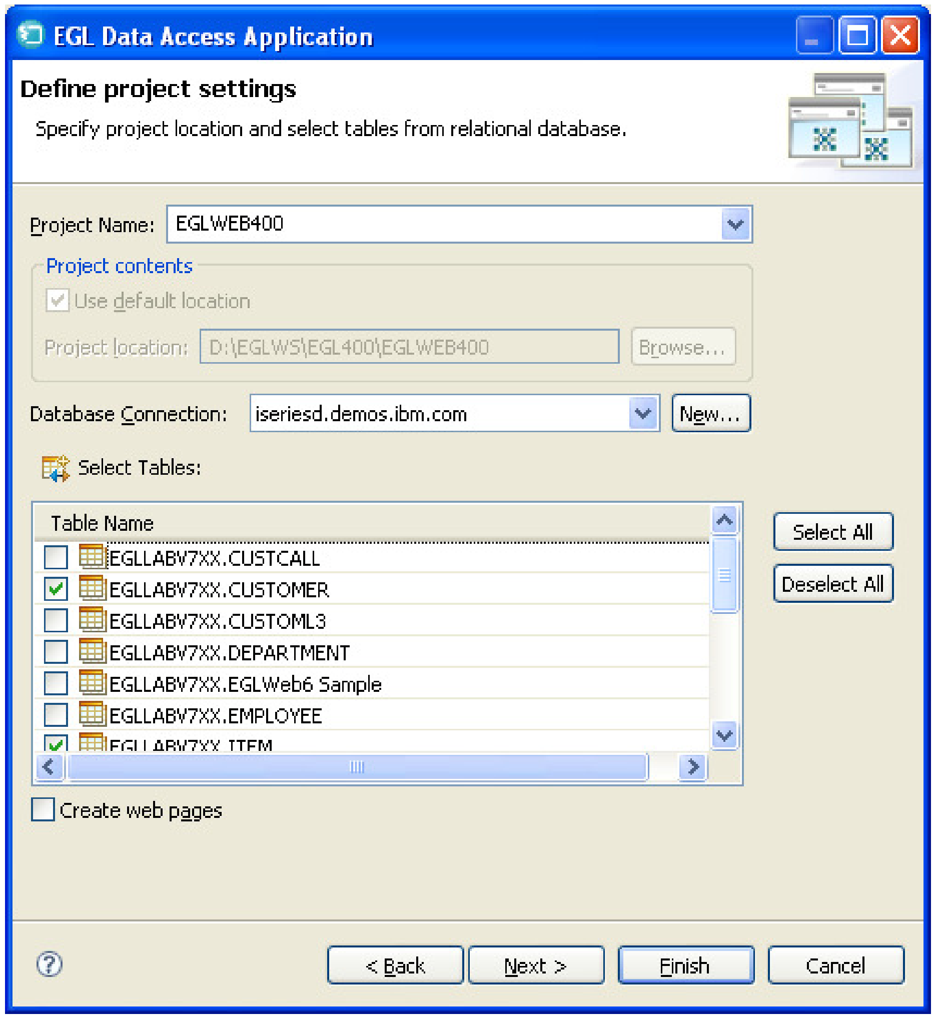
⏭ Select Tables for Import

1. From the Project Name pull down, select EGLWEB400

2. From Database Connection pull down, select iseries.demos.ibm.com

3. Check the following tables:
   - EGLLABV7XX.CUSTOMER
   - EGLLABV7XX.ITEM
   - EGLLABV7XX.ORDERS
   - EGLLABV7XX.ORDER_ITEM
   - EGLLABV7XX.STATETABLE

4. Click **Next >**



EGL Data Access Application

Define project settings

Specify project location and select tables from relational database.

Project Name: EGLWEB400

Project contents

☑ Use default location

Project location: D:\EGLWS\EGL400\EGLWEB400 Browse...

Database Connection: iseriesd.demos.ibm.com New...

Select Tables:

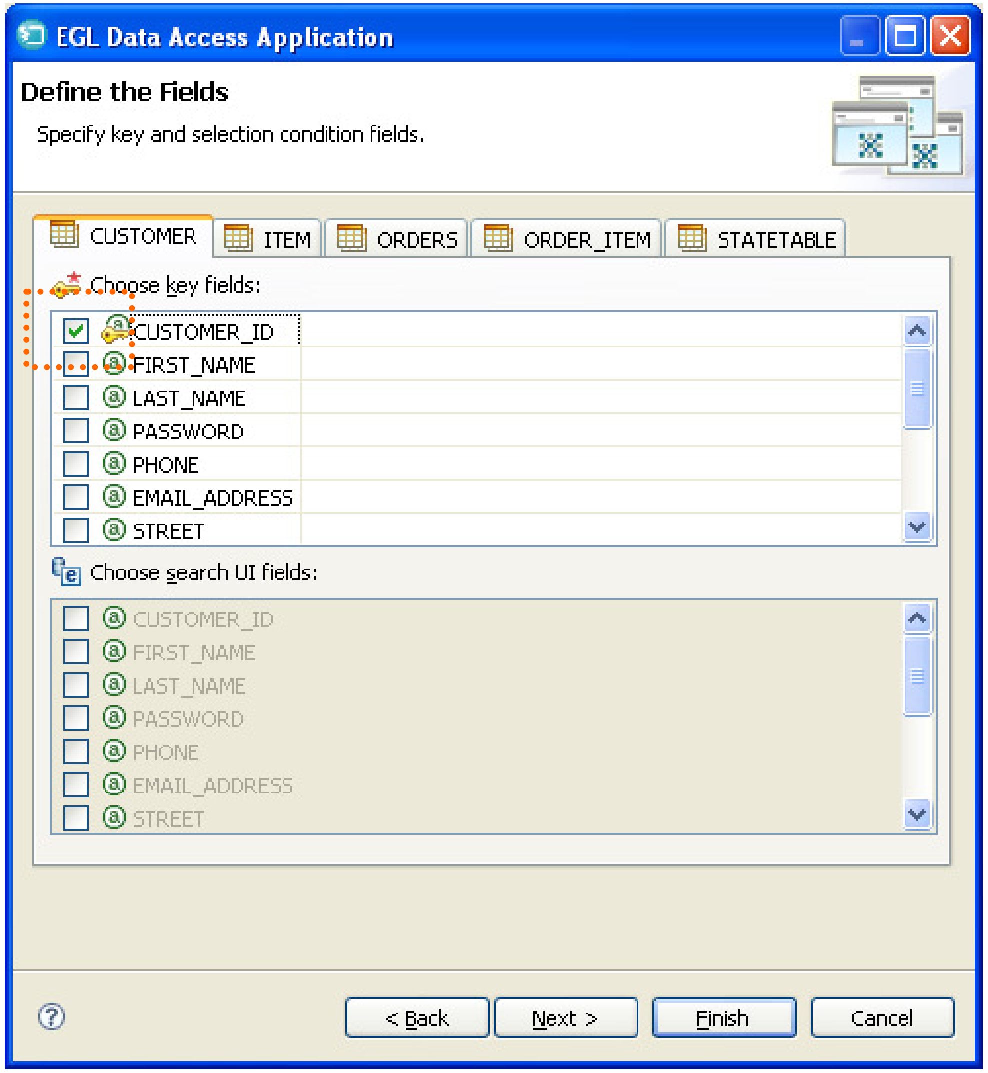
	Table Name
☐	EGLLABV7XX.CUSTCALL
☑	EGLLABV7XX.CUSTOMER
☐	EGLLABV7XX.CUSTOML3
☐	EGLLABV7XX.DEPARTMENT
☐	EGLLABV7XX.EGLWeb6 Sample
☐	EGLLABV7XX.EMPLOYEE
☑	EGLLABV7XX.ITEM

☐ Create web pages

Verify Table Search Keys

If a selected table does not have a Primary Key specified for it in the database catalog, you will be prompted to select a key field for it. This key field is used in the EGL SQL statement generation.

- It might be the case that CUSTOMER needs you to choose a key field.
  - **Choose: CUSTOMER_ID**
- Click ITEM tab:
  - **Choose: ITEM_ID**
- Repeat for the followings:
  - ORDERS: ORDER_ID
  - ORDER_ITEM: ORDER_ITEM_ID
  - STATETABLE: STATEABBREV


EGL Data Access Application

Define the Fields

Specify key and selection condition fields.

Tabs: CUSTOMER | ITEM | ORDERS | ORDER_ITEM | STATETABLE

Choose key fields:

- [✓] CUSTOMER_ID
- [ ] FIRST_NAME
- [ ] LAST_NAME
- [ ] PASSWORD
- [ ] PHONE
- [ ] EMAIL_ADDRESS
- [ ] STREET

Choose search UI fields:

- [ ] CUSTOMER_ID
- [ ] FIRST_NAME
- [ ] LAST_NAME
- [ ] PASSWORD
- [ ] PHONE
- [ ] EMAIL_ADDRESS
- [ ] STREET

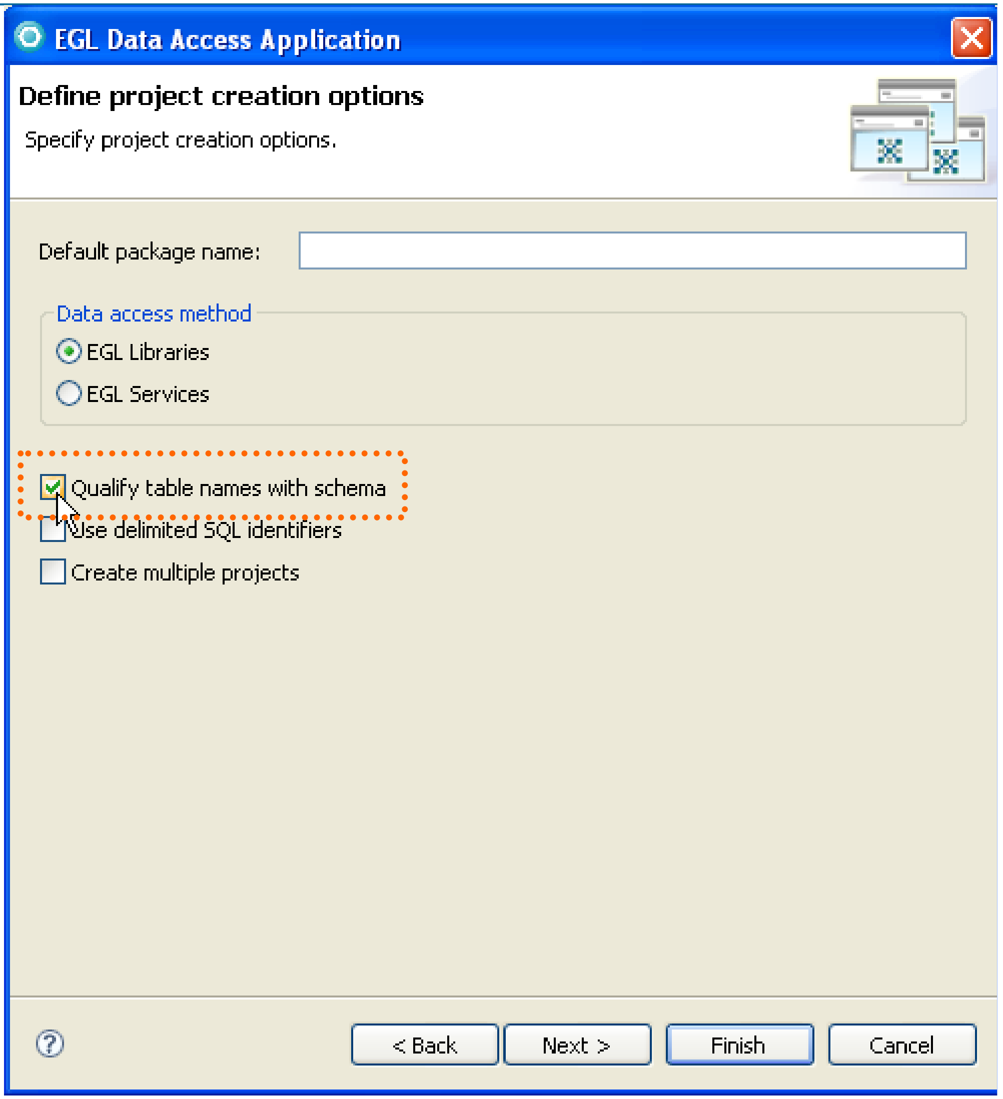
- Click: **Next >**

⏭ Qualify Table Names

- For this workshop you will qualify the table name with the table's schema-name (the schema-name is usually the authorization-id of the table "owner" – like a DBA, for example).

1. Check:
   ☑ **Qualify table names with schema**

2. Click: **Next >**


EGL Data Access Application

Define project creation options

Specify project creation options.

Default package name:

Data access method
- [✓ EGL Libraries]
- EGL Services

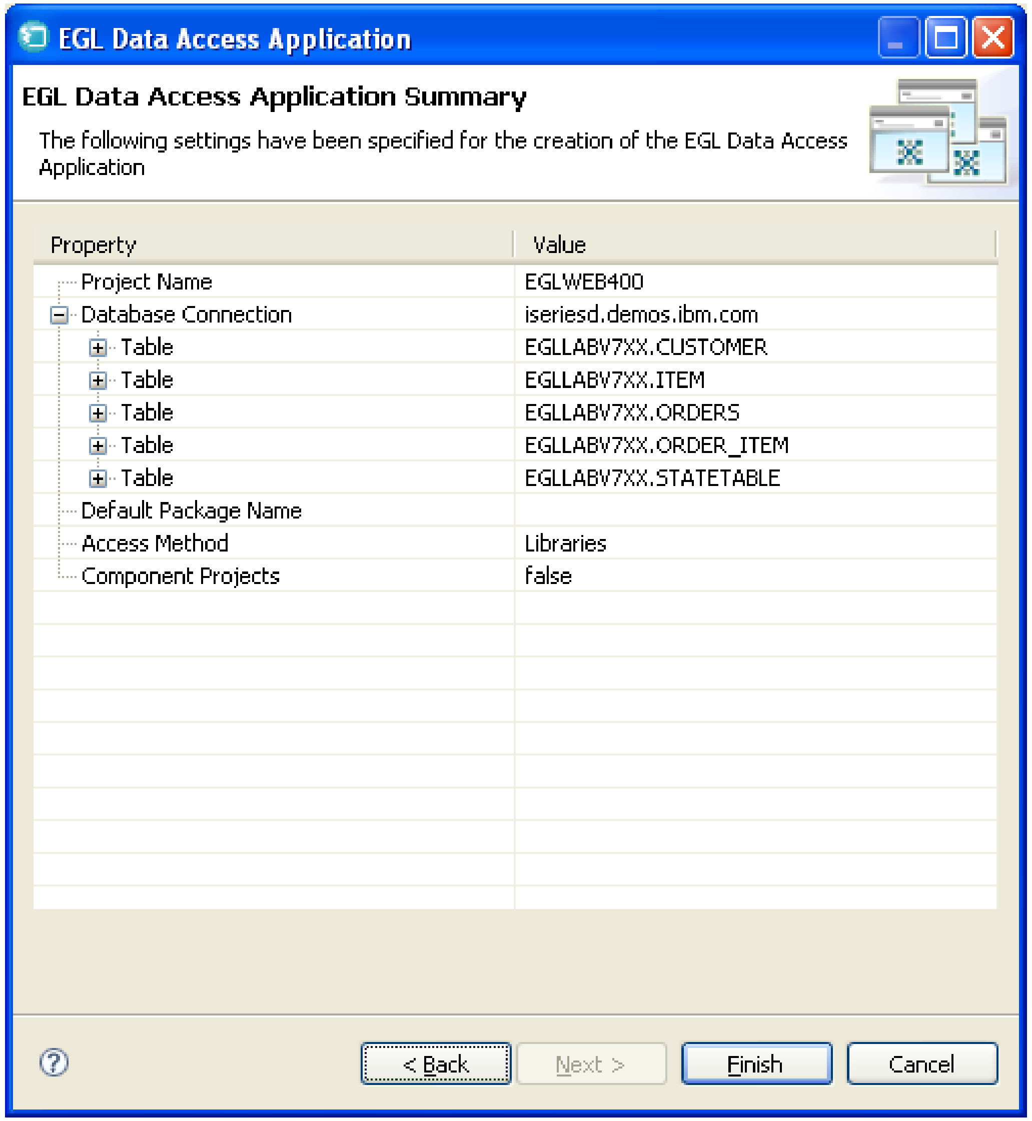
- ☑ Qualify table names with schema
- ☐ Use delimited SQL identifiers
- ☐ Create multiple projects

Generate

Finally, a summary screen is shown, which shows your Data Access import options.

You can expand and look at them, go back, or just:

- Click: **Finish**



EGL Data Access Application Summary

The following settings have been specified for the creation of the EGL Data Access Application

Property	Value
Project Name	EGLWEB400
Database Connection	iseriesd.demos.ibm.com
Table	EGLLABV7XX.CUSTOMER
Table	EGLLABV7XX.ITEM
Table	EGLLABV7XX.ORDERS
Table	EGLLABV7XX.ORDER_ITEM
Table	EGLLABV7XX.STATETABLE
Default Package Name	
Access Method	Libraries
Component Projects	false

What Just Happened?

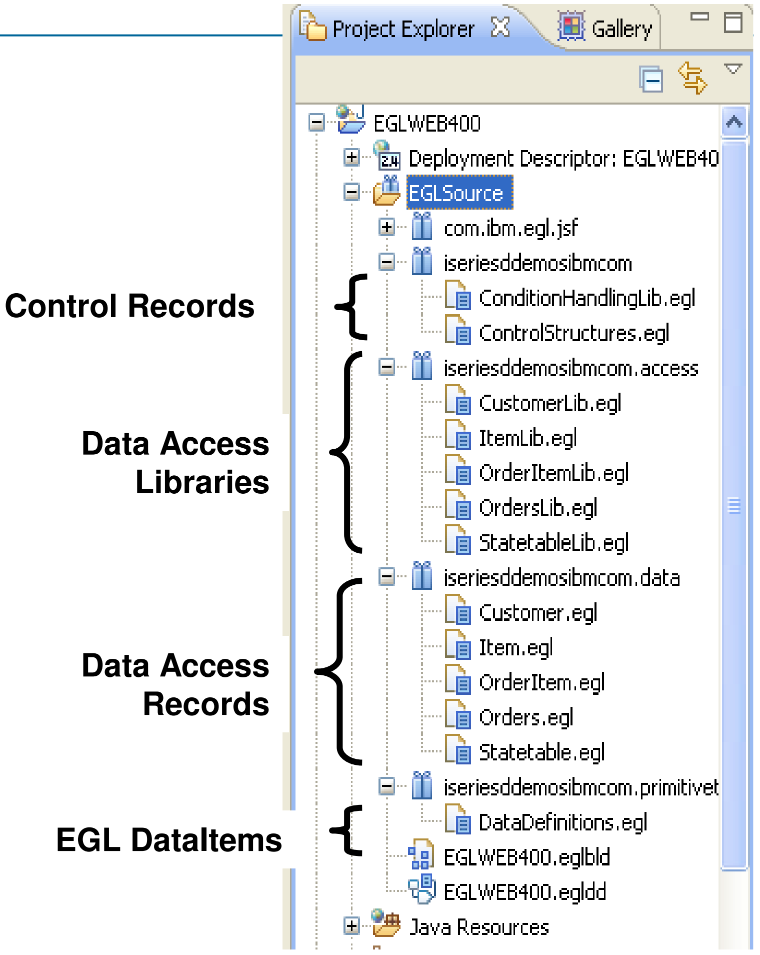
The Data Access Wizard created the following four sets of resources:

1. **Control Records** – used in the default Data Access Library SQL statements

2. **Data Access Libraries** – which contain EGL data access statements (basic data access design patterns), that generate to SQL statements and routines

3. **Data Access Records** – EGL records of type sqlRecord, which participate in the Java and SQL generation, allowing you to code at a high-level of abstraction

4. **EGL DataItems** – in DataDefinitions.egl. This file contains default type definitions for each column in each imported table. 🗒 See notes for more on DataItems



```
EGLWEB400
  Deployment Descriptor: EGLWEB40
  EGLSource
    com.ibm.egl.jsf
    iseriesddemosibmcom
      ConditionHandlingLib.egl
      ControlStructures.egl
    iseriesddemosibmcom.access
      CustomerLib.egl
      ItemLib.egl
      OrderItemLib.egl
      OrdersLib.egl
      StatetableLib.egl
    iseriesddemosibmcom.data
      Customer.egl
      Item.egl
      OrderItem.egl
      Orders.egl
      Statetable.egl
    iseriesddemosibmcom.primitivet
      DataDefinitions.egl
    EGLWEB400.eglbld
    EGLWEB400.egldd
  Java Resources
```

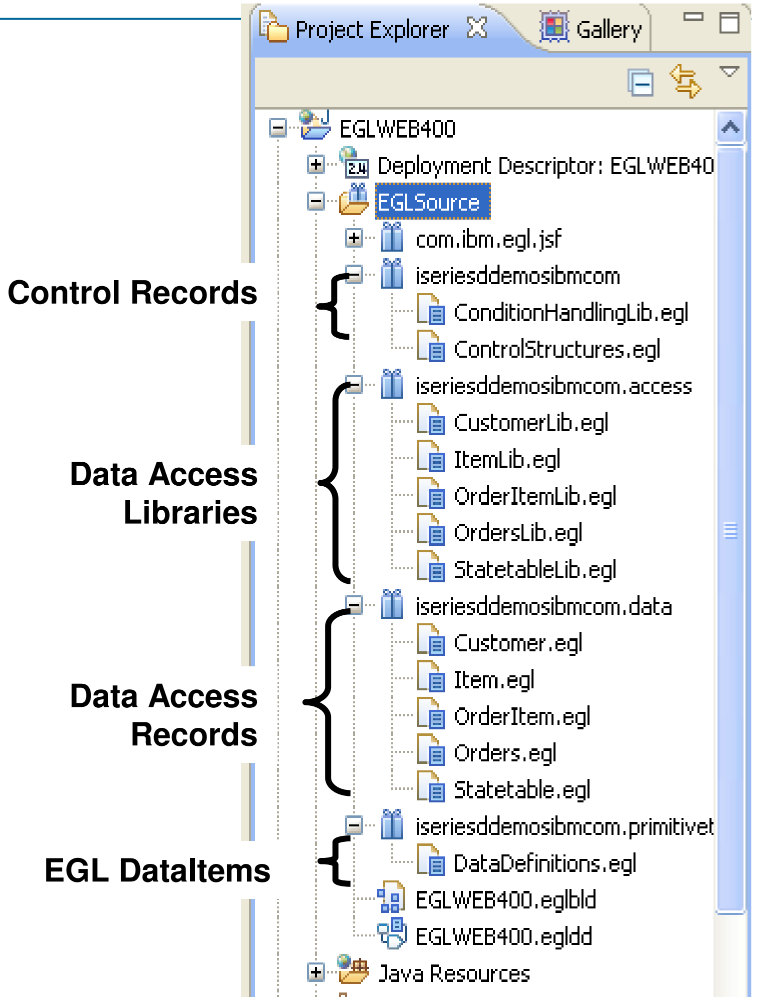
View Results – See What Was Produced by the Wizard

- Expand the four new folders under EGLSource named: **iseriesddemosibmcom**.<something>

- Open a few of the files produced by the wizard in the Content Area (Double-click: `CustomerLib.egl, Customer.egl, DataDefinitions.egl`, etc.) and view the code that was generated.

- Since we will be covering EGL in an upcoming unit of this course it is not expected that you understand all – of what was produced from Data Access import.

- However the default data access code is:
  - A nice starting point for doing and learning about database programming in EGL
  - Nice to have been generated (as opposed to hand-coding yourself)



```
EGLWEB400
  Deployment Descriptor: EGLWEB40
  EGLSource
    com.ibm.egl.jsf
    iseriesddemosibmcom
      ConditionHandlingLib.egl
      ControlStructures.egl
    iseriesddemosibmcom.access
      CustomerLib.egl
      ItemLib.egl
      OrderItemLib.egl
      OrdersLib.egl
      StatetableLib.egl
    iseriesddemosibmcom.data
      Customer.egl
      Item.egl
      OrderItem.egl
      Orders.egl
      Statetable.egl
    iseriesddemosibmcom.primitivet
      DataDefinitions.egl
    EGLWEB400.eglbld
    EGLWEB400.egldd
  Java Resources
```

- Let's do a few specific things to some of these files (steps begin on the next slide)…

⏭ Customize the Default-Generated Code – 1 of 2

In \EGLSource\eglderbyr7.access\

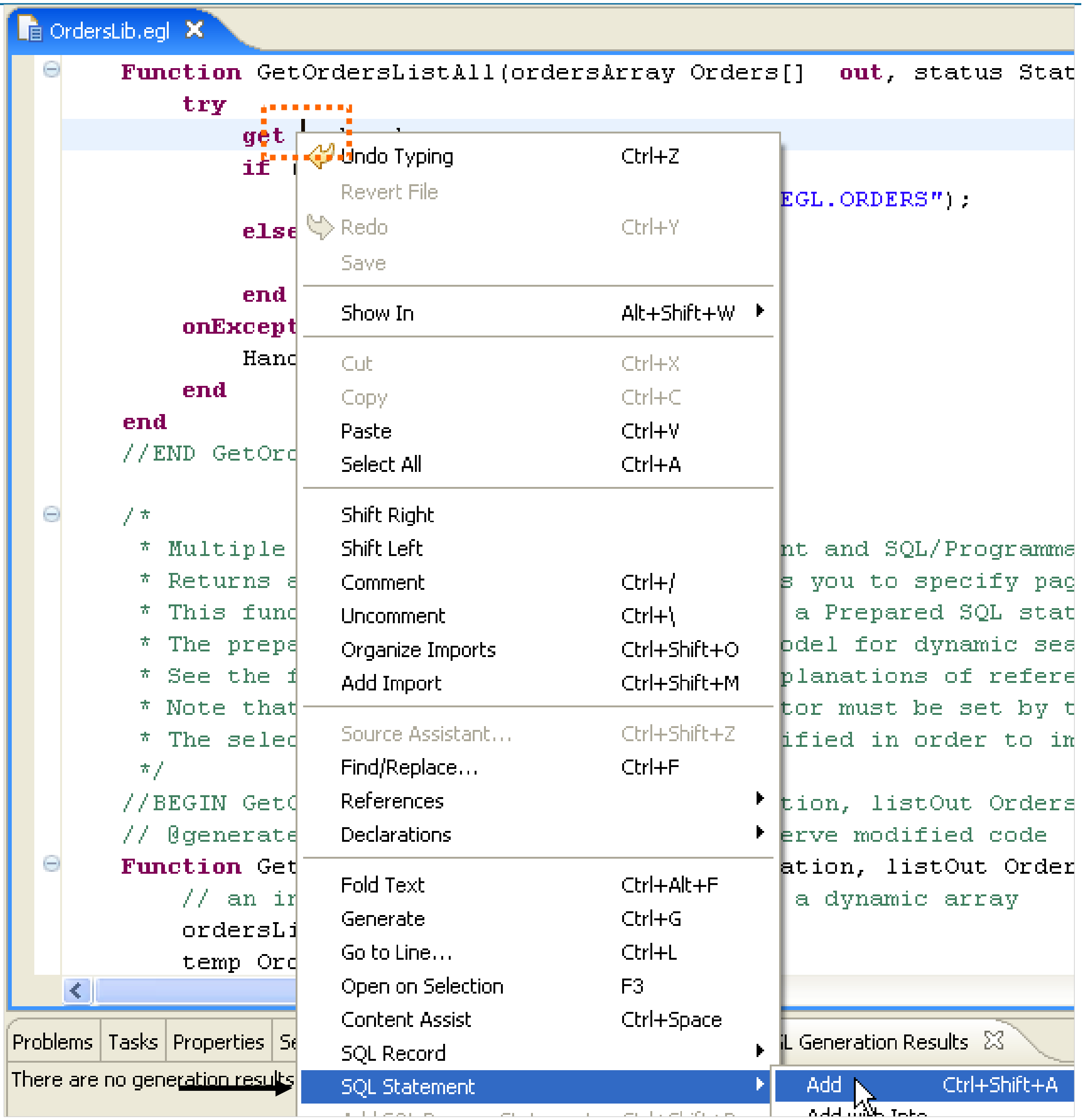
- Open **OrdersLib.egl** and find the function: **`GetOrdersListAll`**

- Click your mouse between the word **`get`** and **`ordersArray;`**

- **Right-click** and select:

  - **SQL Statement ▶ Add**

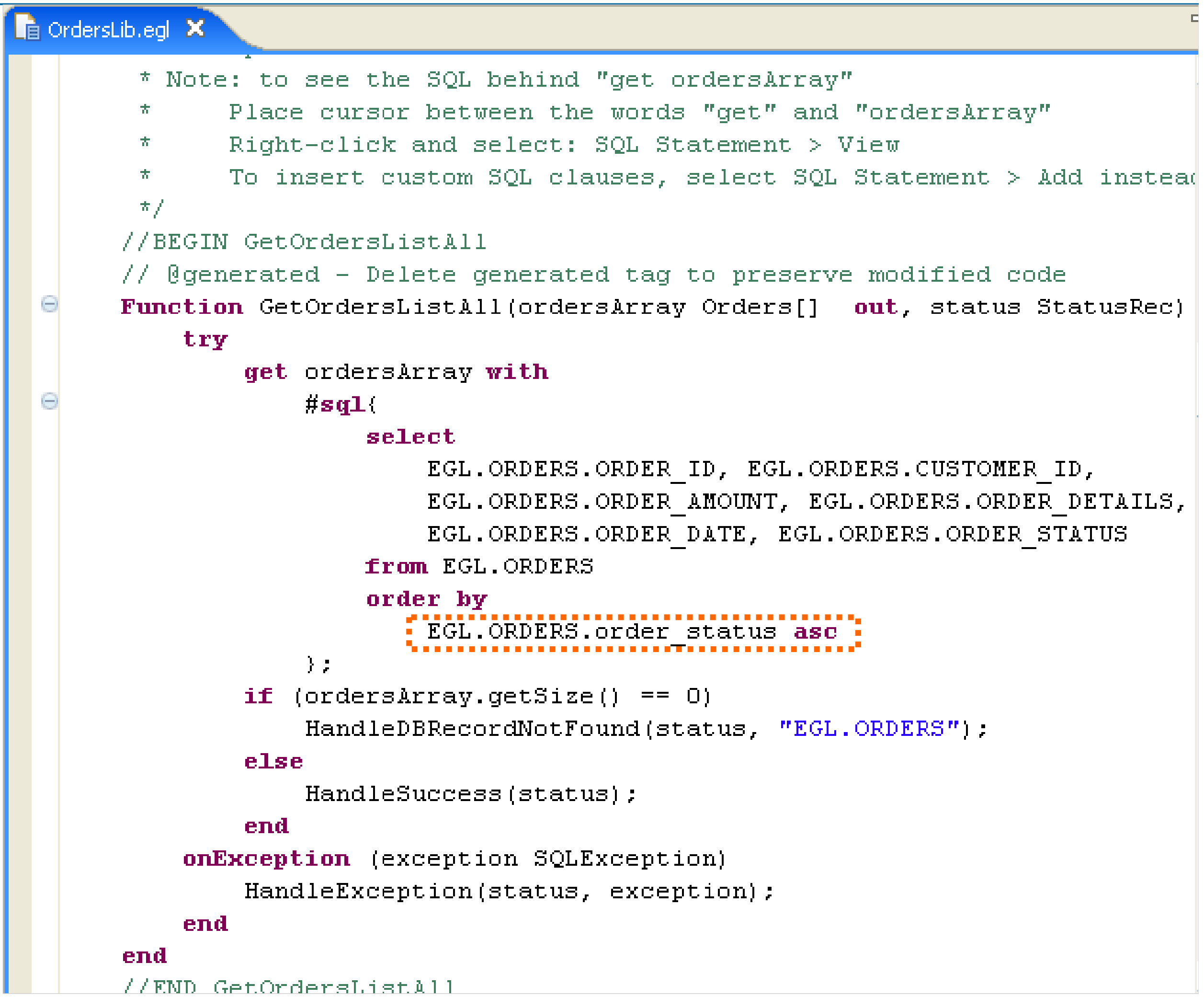
This causes RBD to insert the SQL statement it would generate, inline in your function, allowing you to customize the data access code (see next slide)



```
Function GetOrdersListAll(ordersArray Orders[]  out, status Stat
    try
        get | 
        if                                                EGL.ORDERS");

        else

        end
    onExcept
        Hanc
    end
end
//END GetOrc

/*
 * Multiple                                              nt and SQL/Programme
 * Returns a                                             s you to specify pag
 * This func                                             a Prepared SQL stat
 * The prepa                                             odel for dynamic sea
 * See the f                                             planations of refere
 * Note that                                             tor must be set by t
 * The selec                                             ified in order to im
 */
//BEGIN GetC                                             tion, listOut Orders
// @generate                                             erve modified code
Function Get                                             ation, listOut Order
    // an ir                                             a dynamic array
    ordersLi
    temp Orc
```

⏭ Customize the Default-Generated Code – 2 of 2

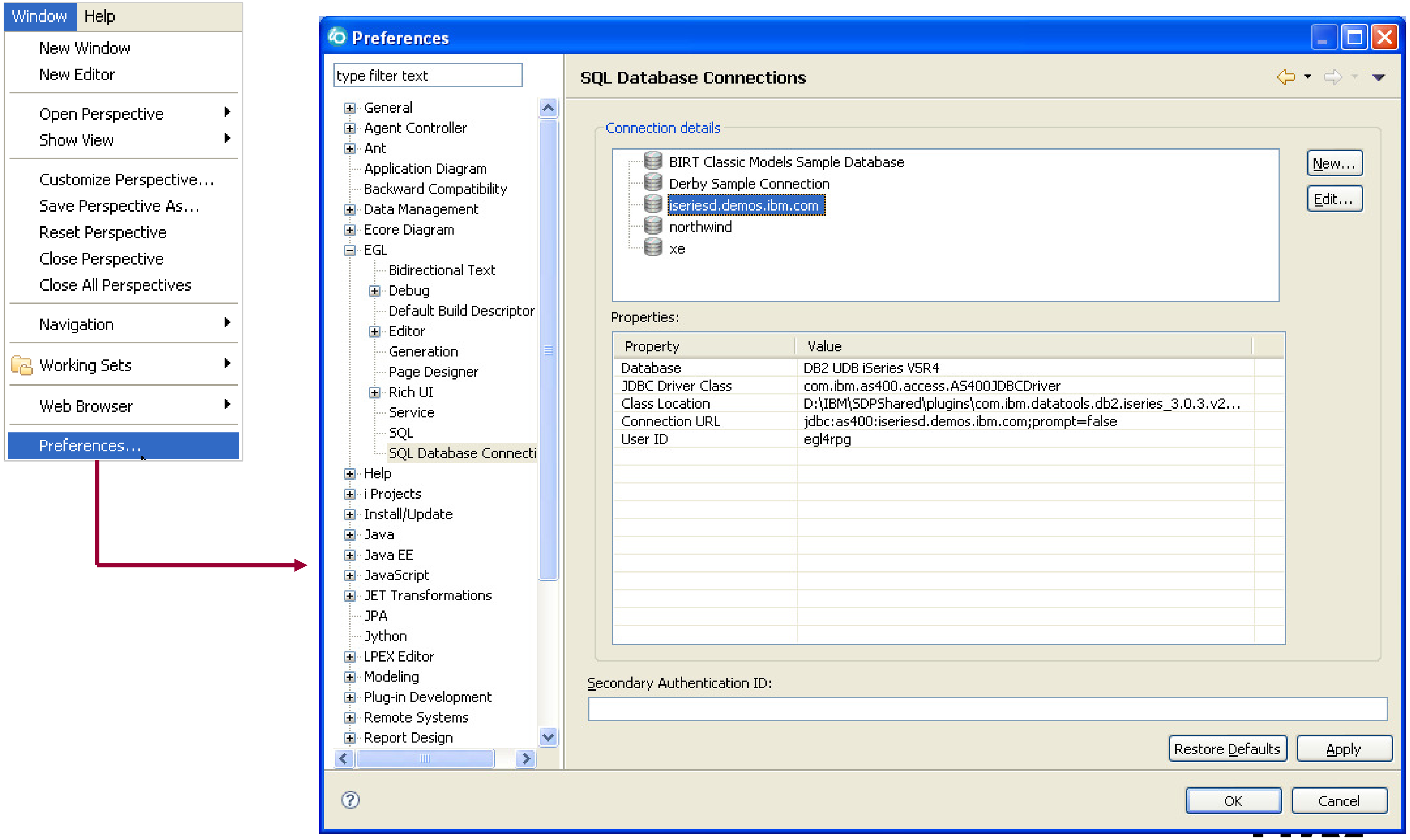
- ⏭ Change the **order by** clause to sort the results of this SQL statement by the **order_status** field.
  - Note – type ⌨ carefully.
- Press **Ctrl/S** to save
- In doing EGL/Java development, most SQL statement typos are not caught at compile time, but at run-time, making them a royal pain to debug. **See Notes**
- But the tooling can catch these (as we'll see in an upcoming section) – want a hint? (see Optional Exercise on the next slide)


OrdersLib.egl

```
   * Note: to see the SQL behind "get ordersArray"
   *     Place cursor between the words "get" and "ordersArray"
   *     Right-click and select: SQL Statement > View
   *     To insert custom SQL clauses, select SQL Statement > Add instead
   */
  //BEGIN GetOrdersListAll
  // @generated - Delete generated tag to preserve modified code
  Function GetOrdersListAll(ordersArray Orders[]  out, status StatusRec)
      try
          get ordersArray with
              #sql{
                  select
                      EGL.ORDERS.ORDER_ID, EGL.ORDERS.CUSTOMER_ID,
                      EGL.ORDERS.ORDER_AMOUNT, EGL.ORDERS.ORDER_DETAILS,
                      EGL.ORDERS.ORDER_DATE, EGL.ORDERS.ORDER_STATUS
                  from EGL.ORDERS
                  order by
                      EGL.ORDERS.order_status asc
              };
          if (ordersArray.getSize() == 0)
              HandleDBRecordNotFound(status, "EGL.ORDERS");
          else
              HandleSuccess(status);
          end
      onException (exception SQLException)
          HandleException(status, exception);
      end
  end
  //END GetOrdersListAll
```

Test the Customized SQL Statement – 1 of 2 (Set Preferences)

- **Optional Exercise:** From **Window, Preferences**, open **EGL, SQL Database Connections**, and select your iseriesd.demos.ibm.com connection – then click: **OK**



Property	Value
Database	DB2 UDB iSeries V5R4
JDBC Driver Class	com.ibm.as400.access.AS400JDBCDriver
Class Location	D:\IBM\SDPShared\plugins\com.ibm.datatools.db2.iseries_3.0.3.v2...
Connection URL	jdbc:as400:iseriesd.demos.ibm.com;prompt=false
User ID	egl4rpg

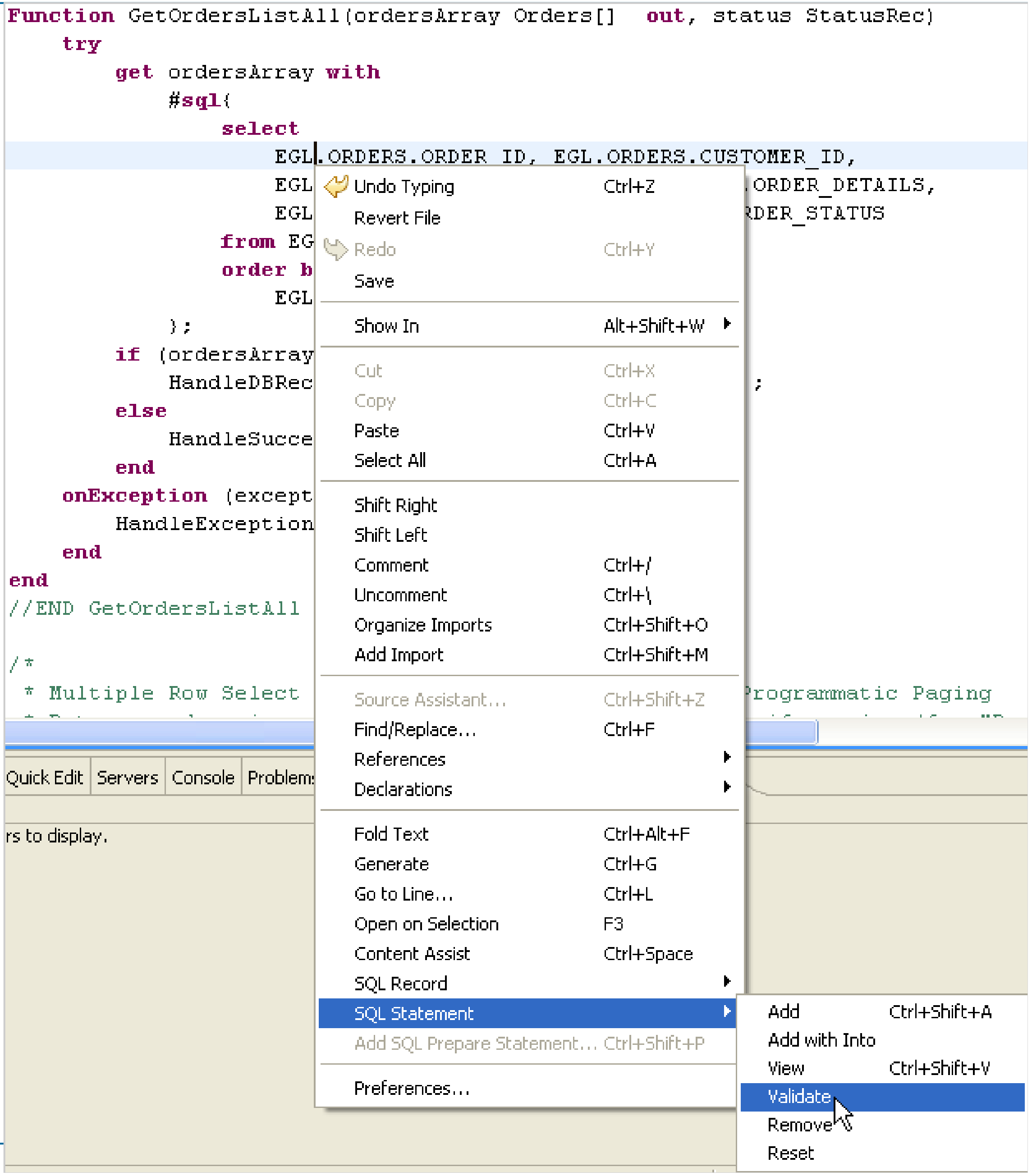
Test the Customized SQL Statement – 2 of 2

- **Optional Exercise:** Click your mouse inside the **select** statement.
- Right-click and select:

  SQL Statement ▶ Validate

  Click **OK** at the SQL User ID/Password prompt.

- If there are SQL problems, you will find out now upon Validating your statement.
- If there are EGL “host variables” in your statement, and there are problems with your EGL host-variables, you will find out when you save (**Ctrl**/**S**)
  - **If you are not familiar with the term, EGL “host variables” are covered later in this course.**

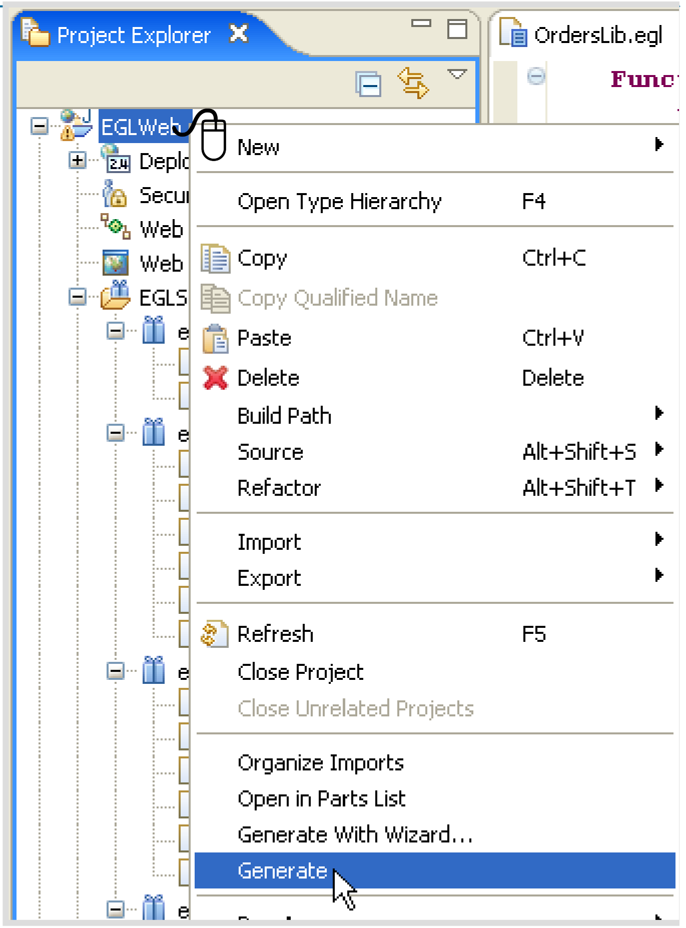


```
Function GetOrdersListAll(ordersArray Orders[]  out, status StatusRec)
    try
        get ordersArray with
            #sql{
                select
                    EGL.ORDERS.ORDER_ID, EGL.ORDERS.CUSTOMER_ID,
                    EGL                              .ORDER_DETAILS,
                    EGL                              RDER_STATUS
                from EG
                order b
                    EGL
            };
        if (ordersArray                               ;
            HandleDBRec
        else
            HandleSucce
        end
    onException (except
        HandleException
    end
end
//END GetOrdersListAll

/*
 * Multiple Row Select                      rogrammatic Paging
```

⏭ Generate the Code Changes

(From Project Explorer)

- Right-click over **EGLWeb**
- Select **Generate**



👓 This will generate new Java code for your changes to the default code created by the Database Access Application Wizard

Topic Summary

- Now that you have completed this topic, you should have:
  - Used the EGL Data Access wizard to import – from your Derby database:
    - Data Access Libraries
    - Data Access Records
    - DataItems (an operational data dictionary)
    - SQL call control records
  - Customized:
    - The DataItems
    - A Data Access Library Function
  - Successfully re-generated all of the Data Access import wizard code.

UNIT

EGL/Web QuickStart

Topics:

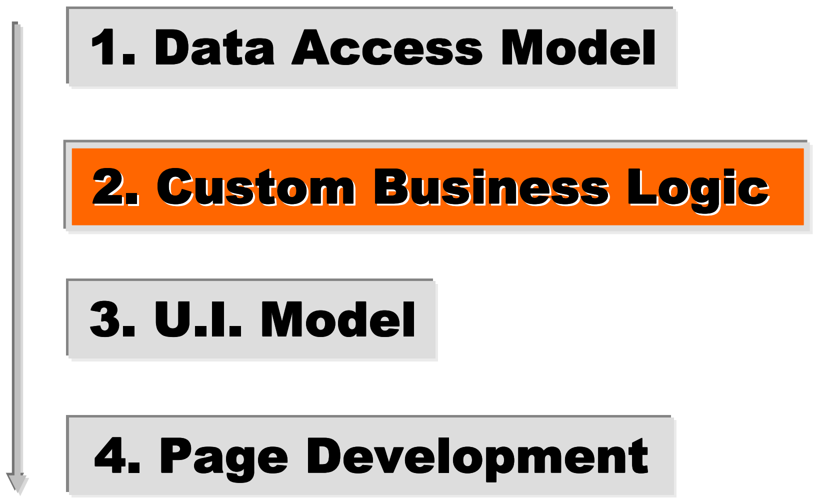
- Development Model – Terms and Concepts
- Importing Database Schema
- **Creating Custom Business Logic**
- Creating the U.I. Model
- Creating Dynamic Content Web Pages

The EGL Project Model – Where we are, from 10,000 Feet

After generating the Data Access model – and customizing the SQL statements and DatatItems, you will probably begin coding your business logic. We say probably, because you could do this in parallel with Data Access Modeling – so the order of the steps is not dependent. It's only convenient to discuss it as such for this simple web application.

- For your business logic you may want to create:
  - **Services or Web Services** – to move towards a Service Oriented Architecture
  - **Libraries** – to hold independent, granular business logic functions
  - **Programs** – to hold larger-scale business logic (reports, calls to the mainframe, etc.)

- We will be covering all of these types of EGL logic parts in the Unit that follows the QuickStart, for now, we'll create a simple new function in Library – so that you get the 10,000 foot view.



Create EGL Business Logic

Follow the steps beginning on the next slide to create the following simple test for zip code. The steps are:

- Edit an existing library
- Copy/Paste an existing function
- Modify the function name and parameter
- Use Content Assist to add new business logic
- Save and Generate

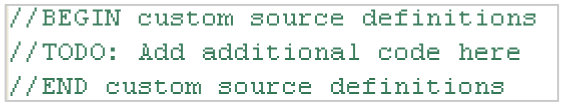
```
Function zipTest(zip string) returns( boolean )
    if (zip is numeric)
        return(true);
    else
        return(false);
    end
end
```

Note: In this workshop you will create this simple business logic function inside of an existing library. But the same process and simple EGL programming model will be used (as you'll see in the next Unit on the EGL Language) to:

- Create a service or Web Service
- Consume (**call** a service or Web Service)
- Call an external program:
    - Mainframe COBOL or RPG
    - Java or C++ class

Adding Custom EGL to the Data Access Wizard’s Generated Code

- Throughout the course (and especially in your production projects) you will add lots of custom EGL to the library code generated by the data access application wizard.
- You may be asking yourself: *“What if my database schema changes and I have to re-import the database?”*
- Rest assured, there is a way to re-import part or all of your database schema definitions without losing any custom defined code work you’ve done.
  - **Scroll to the bottom** of any of the generated EGL files.
  - You should see some commented text
    - Notice an area of comments which indicate that you should: “Add additional code here”



```
//BEGIN custom source definitions
//TODO: Add additional code here
//END custom source definitions
```

  - Add all of your custom code at this location between: //**BEGIN** – and – //**END**

```
//BEGIN custom function definitions

    …your new custom EGL code and functions go here…

//END custom function definitions
```

Save and close your data definitions file

- By doing this, you can re-import your entire database or that specific table and not have to worry about losing your custom EGL functions.

***Notes

Copy and Paste a Function

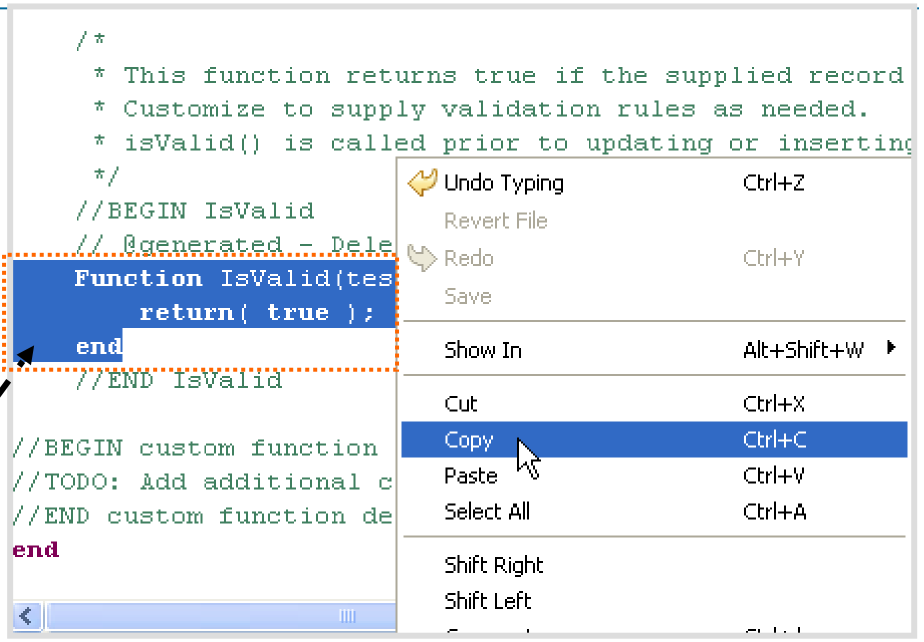
From Project Explorer,

1. Open **CustomerLib.egl**

2. Scroll down to near the end, and find the function named: **`isValid`**

3. Select the entire function (the three lines shown)

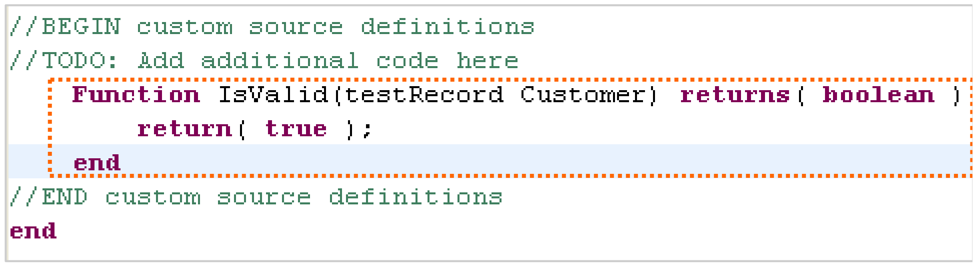
   - **Copy (Ctrl/C)**



```
/*
 * This function returns true if the supplied record
 * Customize to supply validation rules as needed.
 * isValid() is called prior to updating or insertinç
 */
//BEGIN IsValid
// @generated - Dele
   Function IsValid(tes
      return( true );
   end
//END IsValid

//BEGIN custom function
//TODO: Add additional c
//END custom function de
end
```

…….and **Paste (Ctrl/V)** as shown here ➔
Below: `//TODO: Add additional code here`

Above the final **end** statement in the file



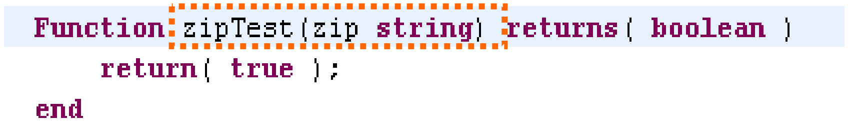
```
//BEGIN custom source definitions
//TODO: Add additional code here
   Function IsValid(testRecord Customer) returns( boolean )
      return( true );
   end
//END custom source definitions
end
```

- This will duplicate the function in the library – in the area where you can add custom functions (which won't be over-written)

Use Content Assist to Modify an EGL Function - 1 of 2

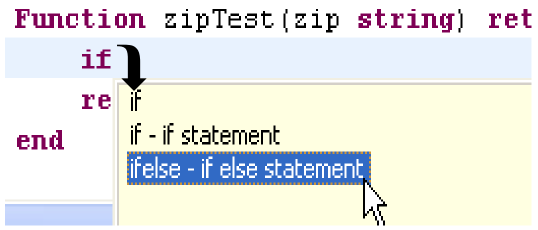
You will use the EGL Editor's "Content Assist" to modify the function by following these steps. Note that Content Assist is an "intelligent editor" – it allows you to select some value that makes sense to code within the context of EGL business logic.

4. Change the function name and input parameter ➔



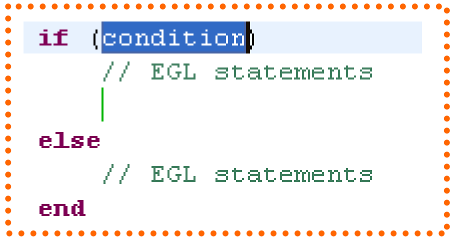
```
Function zipTest(zip string) returns( boolean )
    return( true );
end
```

5. Enter a new line
   - Type the word: **`if`**
     - Immediately press **Ctrl+Spacebar**
       – Select **ifelse-if else statement** ➔



```
Function zipTest(zip string) ret
    if
    re
end
```

   - This will create a *templated* **if** statement ➔
   - The selected `condition` means that Content Assist is still active (press **Ctrl/Spacebar**, select a variable name)

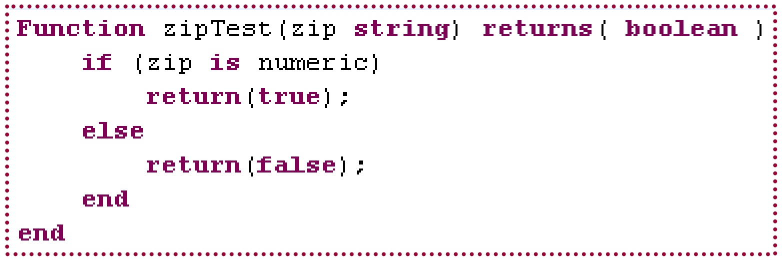


```
if (condition)
    // EGL statements

else
    // EGL statements
end
```

And of course you can simply edit/type at will

Use Content Assist to Modify an EGL Function - 2 of 2

6. Using Content Assist or just plain old editing/typing, complete, **save** and **generate** the function as shown below.



```
Function zipTest(zip string) returns( boolean )
    if (zip is numeric)
        return(true);
    else
        return(false);
    end
end
```

🡐 **Finished Function**

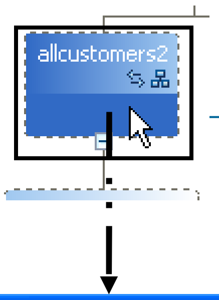
To save and generate:

- **Ctrl/S** saves and validates your EGL statements
  - Note – if you make any typos, small red x's will appear in your source next to the line in question.
    - Mouse-over the red-x's to see a detailed explanation of the problem. Correct it. Press Ctrl/S again, and continue.
- **Ctrl/G** generates Java (assuming that the EGL parser found no syntax errors when you pressed **Ctrl/S**)

Topic Summary

- Now that you have completed this topic, you should have:
  - Coded EGL Business logic by;
    - Editing an existing Library function
    - Duplicating a function, and using it as the basis for new business logic
    - (Using Content Assist) created a new logic function that tests a field for all numeric values, and returned a true/false
  - Saved and Generated Java

⏭ 1. Create the allcustomers2.jsp page



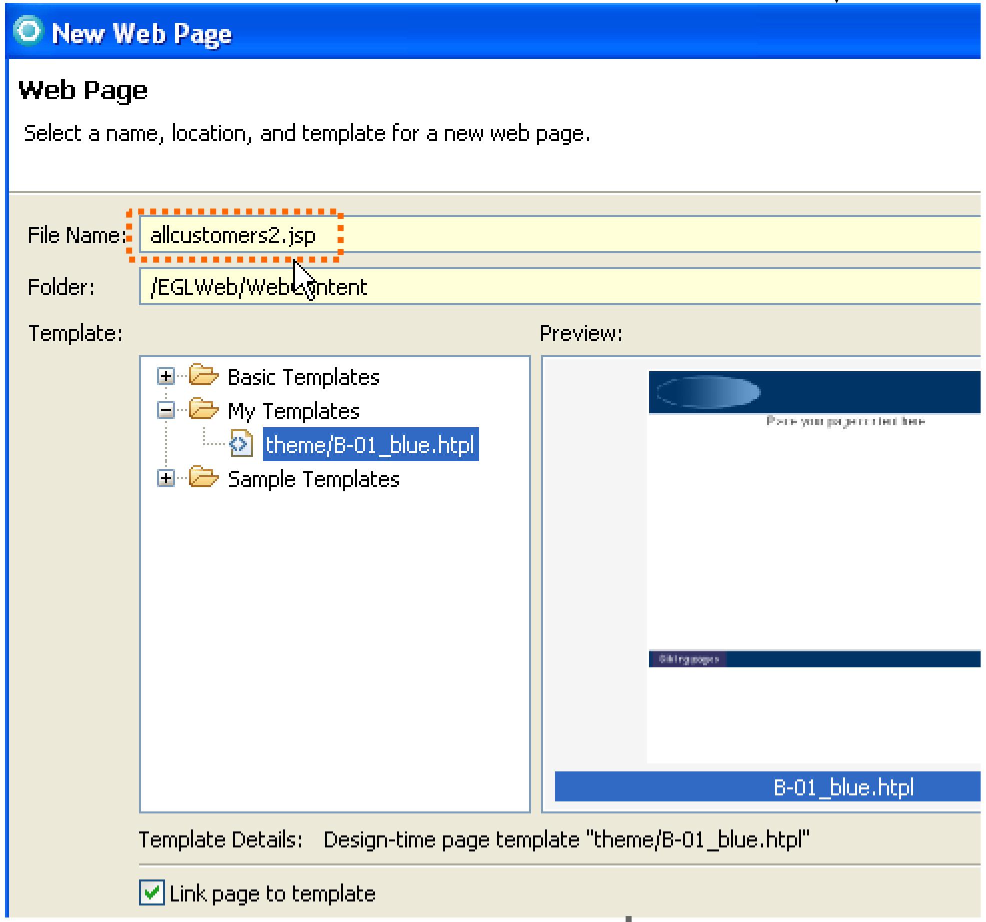
Okay, that was easy ☺ Time to try our first dynamic content web page.

1a. From the Site Map, **Double-Click allcustomers2**

1b. Add the **.jsp** extension to the page name:

`allcustomers2.jsp`

1c. Click **Finish**



Web Page

Select a name, location, and template for a new web page.

File Name: allcustomers2.jsp

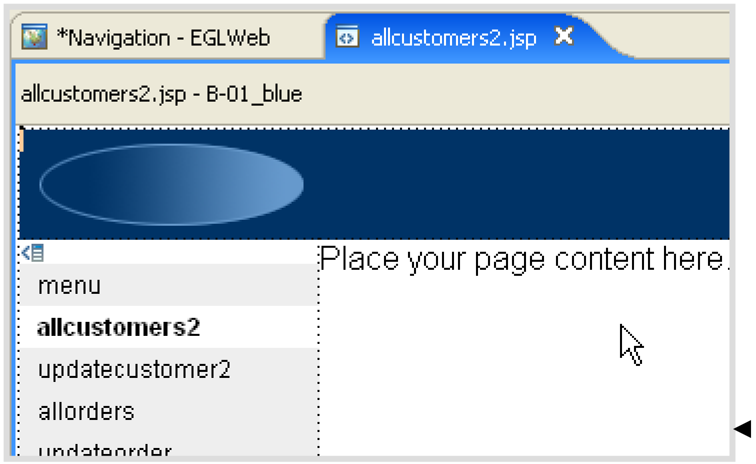
Folder: /EGLWeb/WebContent

Template:

Preview:

Template Details: Design-time page template "theme/B-01_blue.htpl"

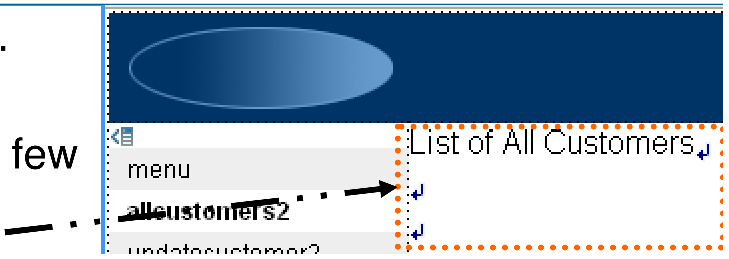
✓ Link page to template



⏭ 2. Add text, graphics or EGL variables – 1 of 3

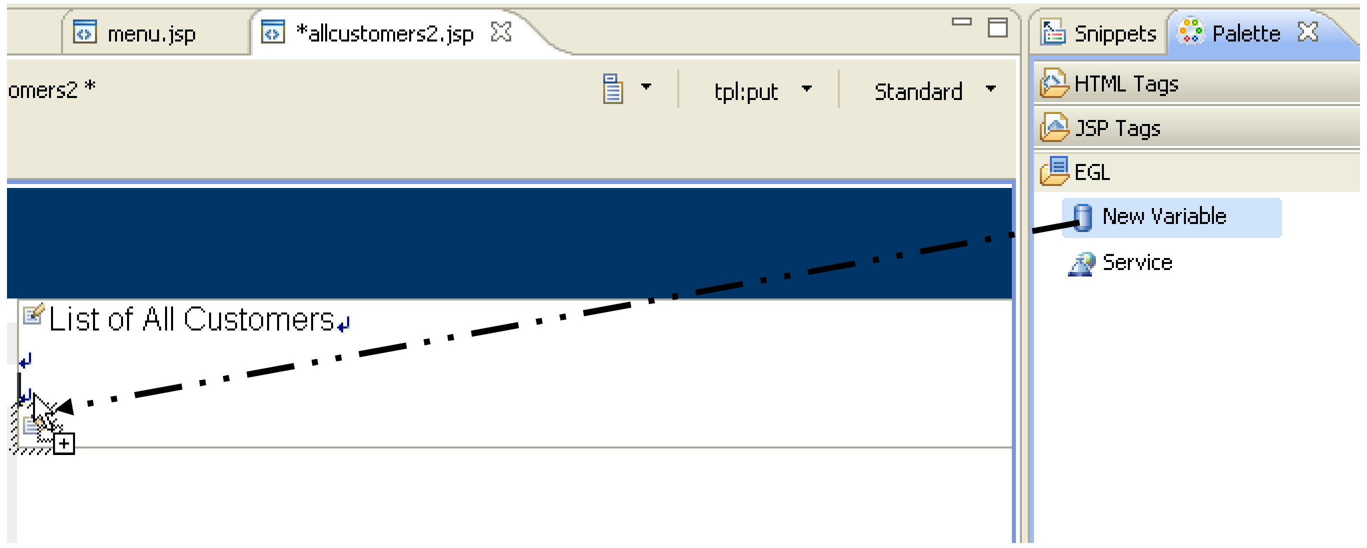
We won't add much in the way of graphics to **allcustomers2**…

2a. Modify the default page header. Add some new text and a few line breaks



2b. From the Palette, Expand the **EGL** drawer

Left-Click (select) Drag & Drop **New Variable** onto the page



⏭ 2. Add EGL variables – 2 of 3

2d. The Create a New EGL Data Variable wizard allows you to select the type of variable to add to your page, and customize its use.

Specify the following

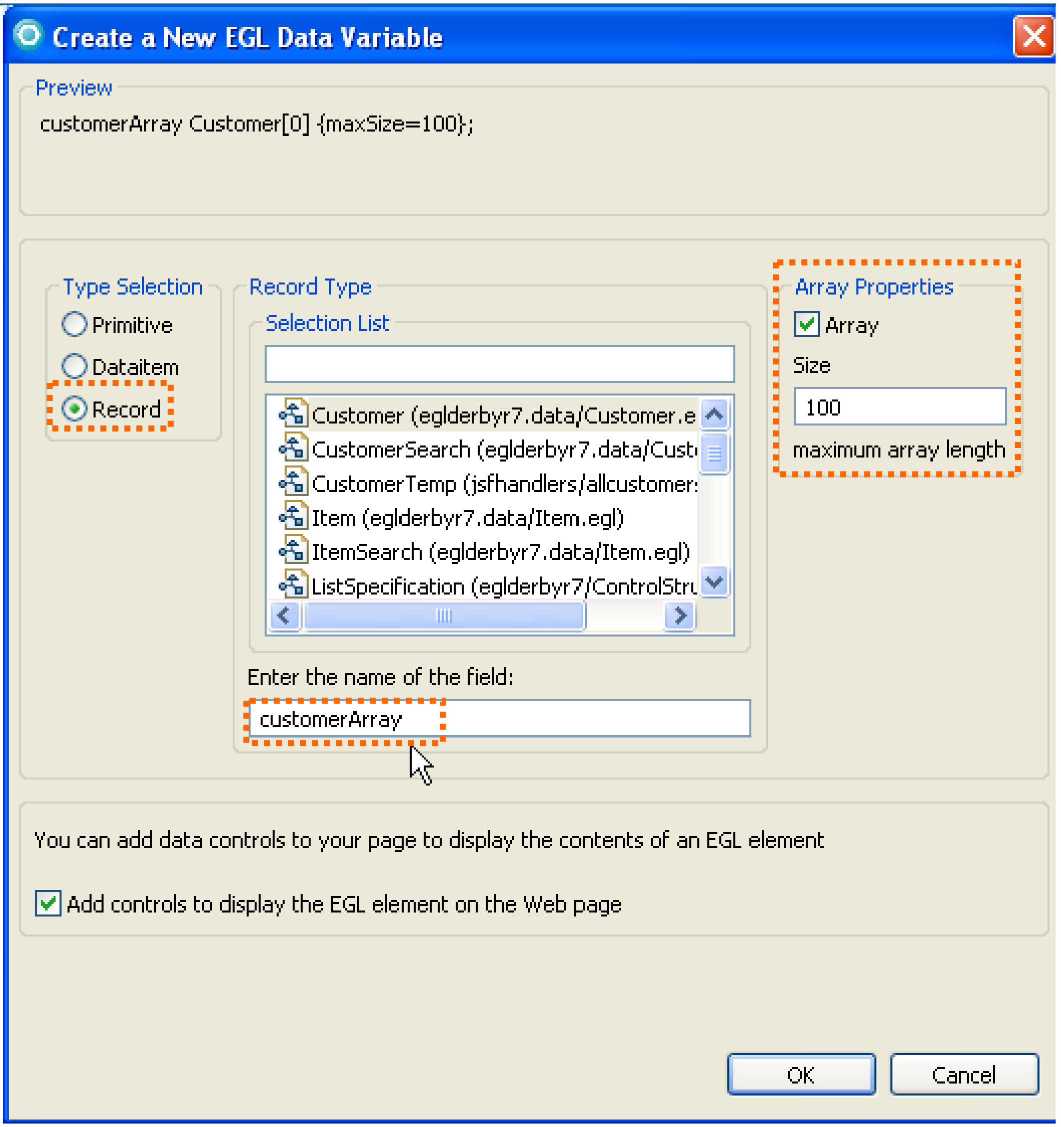
⦿ **Record**

☑ **Array** – Size 100

(Note the size specification will be used during SQL generation to halt cursor fetching after 100 rows)

Name the field: **`customerArray`**

Click **OK**



Create a New EGL Data Variable

Preview

```
customerArray Customer[0] {maxSize=100};
```

Type Selection: Primitive [✓ Record] Dataitem

Record Type – Selection List:
- Customer (eglderbyr7.data/Customer.e
- CustomerSearch (eglderbyr7.data/Cust
- CustomerTemp (jsfhandlers/allcustomer
- Item (eglderbyr7.data/Item.egl)
- ItemSearch (eglderbyr7.data/Item.egl)
- ListSpecification (eglderbyr7/ControlStru

Array Properties: [✓] Array

Size: 100 maximum array length

Enter the name of the field: customerArray

You can add data controls to your page to display the contents of an EGL element

[✓] Add controls to display the EGL element on the Web page

⏭ 2. Add EGL variables – 3 of 3

2e. The Configure Data Controls wizard allows you to select input vs. output controls for your data, as well as which columns of a record to display.

Specify the following

⦿ Displaying an existing record (read-only)

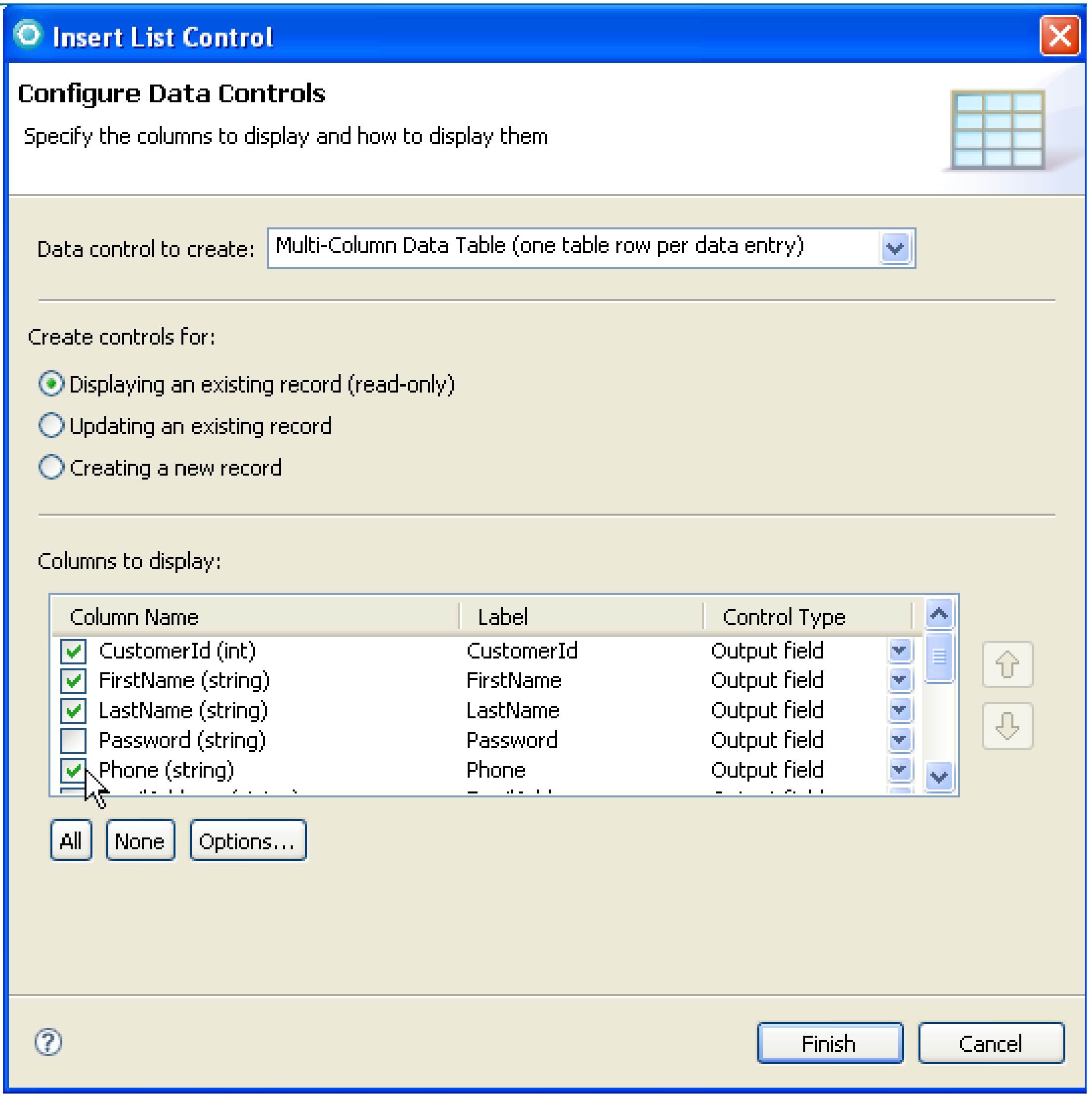
For Columns to display:

Click None – to uncheck all

Then check individually:

- ☑ **CustomerID**
- ☑ **FirstName**
- ☑ **LastName**
- ☑ **Phone**

Click **Finish**



Configure Data Controls

Specify the columns to display and how to display them

Data control to create: Multi-Column Data Table (one table row per data entry)

Create controls for:

[✓ Displaying an existing record (read-only)]
Updating an existing record
Creating a new record

Columns to display:

	Column Name	Label	Control Type
✓	CustomerId (int)	CustomerId	Output field
✓	FirstName (string)	FirstName	Output field
✓	LastName (string)	LastName	Output field
	Password (string)	Password	Output field
✓	Phone (string)	Phone	Output field

⏭ 2. Finished – Adding EGL Variables

- So, by adding EGL variables to a web page, Page Designer creates default labels and JSF Components on the page.

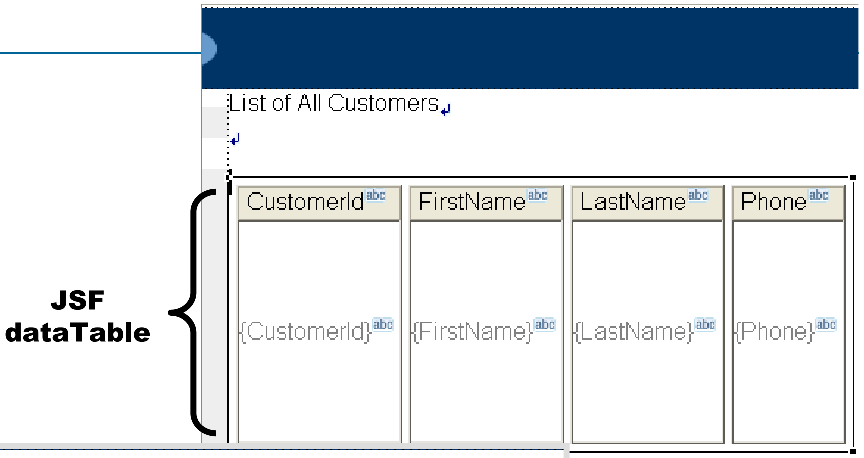
In this case, it created a JSF dataTable which will display your customer rows through an EGL array, once the rows have been accessed from the database.

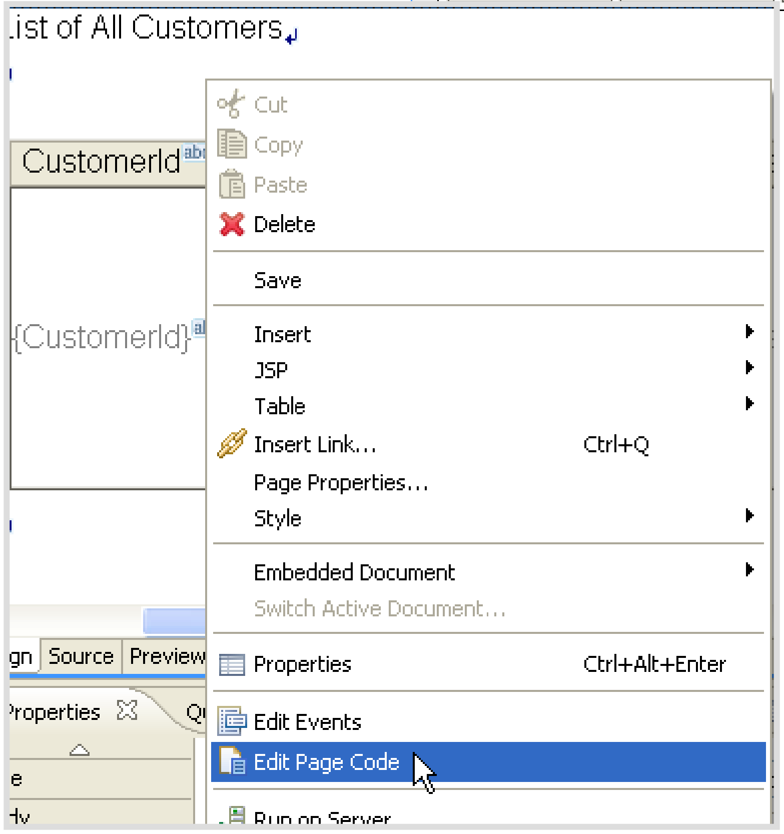
To do that, we'll call one of the Data Access functions you just imported, using the EGL Data Access wizard.

To do that, we'll need to edit our EGL JSFHandler for this page.

⏭ Right-click over the page, and select:

Edit Page Code





⏭ 3. Add EGL Business and Back-End Logic – 1 of 2

Recall that in a previous step, we imported resources from the database, which included Records and Database Access Functions. It's time to put those to use – by coding with the help of Content Assist.

3a. Enter a new line in the **`onConstruction()`** function

3b. Type: **`cust`** and press **Ctrl+Spacebar**

3c. Select **`CustomerLib` – eglderbyr7.access** (library)

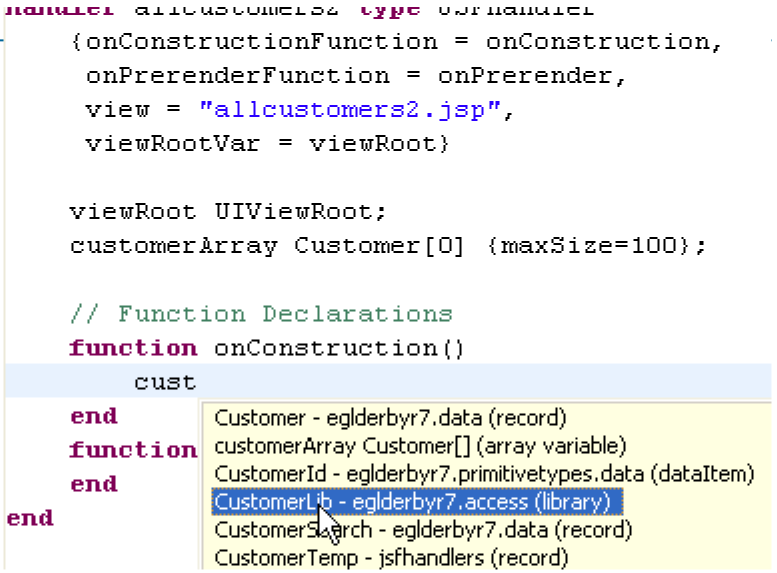
This will add the library name at your cursor focus point

3d. Next, type a period after **`CustomerLib.`**

and press Ctrl+Spacebar

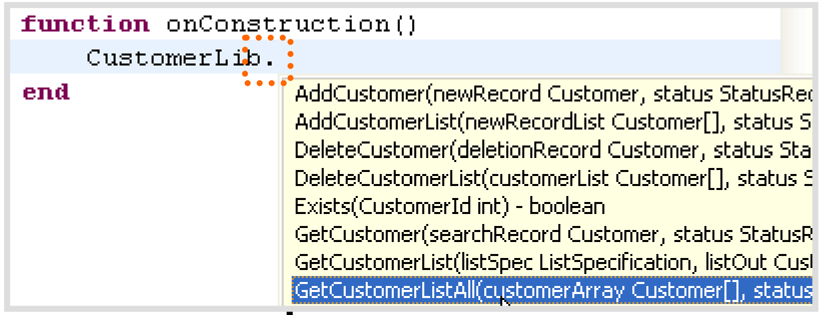
3e. Select **`GetCustomerListAll(customerArray…)`**

This will add the complete **`CustomerLib`** call – minus the ending semi-colon; which you should type ➔



```
handler allcustomers2 type JSFHandler
    {onConstructionFunction = onConstruction,
     onPrerenderFunction = onPrerender,
     view = "allcustomers2.jsp",
     viewRootVar = viewRoot}

    viewRoot UIViewRoot;
    customerArray Customer[0] {maxSize=100};

    // Function Declarations
    function onConstruction()
        cust
    end
    function
    end
end
```

```
Customer - eglderbyr7.data (record)
customerArray Customer[] (array variable)
CustomerId - eglderbyr7.primitivetypes.data (dataItem)
CustomerLib - eglderbyr7.access (library)
CustomerSearch - eglderbyr7.data (record)
CustomerTemp - jsfhandlers (record)
```



```
function onConstruction()
    CustomerLib.
end
```

```
AddCustomer(newRecord Customer, status StatusRec
AddCustomerList(newRecordList Customer[], status S
DeleteCustomer(deletionRecord Customer, status Sta
DeleteCustomerList(customerList Customer[], status S
Exists(CustomerId int) - boolean
GetCustomer(searchRecord Customer, status StatusR
GetCustomerList(listSpec ListSpecification, listOut Cus
GetCustomerListAll(customerArray Customer[], status
```

```
CustomerLib.GetCustomerListAll(customerArray, status);
```

⏭ 3. Add EGL Business and Back-End Logic – 2 of 2

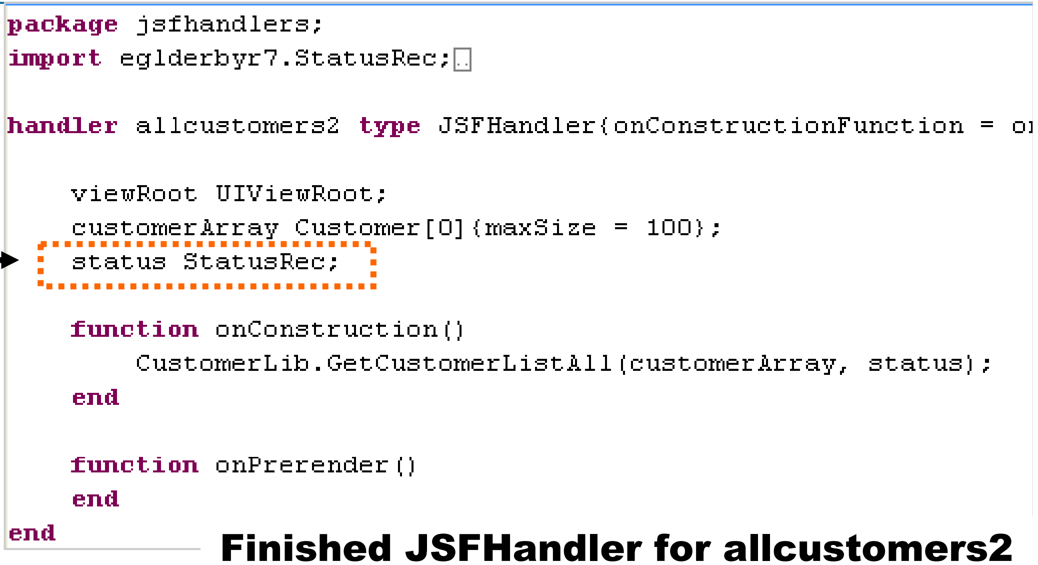
⏭ Finish coding the JSFHandler for this page, by adding an EGL variable for the **`status`** record, as shown in this screen capture

**** IMPORTANT ****

Do **not** just type. ☹

Use Content Assist – as much as possible.

By doing so, the tooling will automatically add the necessary **`import`** statements for you (and it will be SOOO much easier ☺ and you will make so many fewer typos)

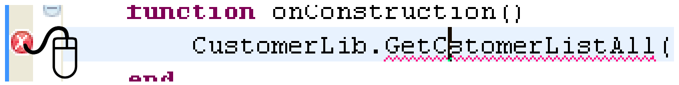


```
package jsfhandlers;
import eglderbyr7.StatusRec;

handler allcustomers2 type JSFHandler(onConstructionFunction = o

    viewRoot UIViewRoot;
    customerArray Customer[0]{maxSize = 100};
    status StatusRec;

    function onConstruction()
        CustomerLib.GetCustomerListAll(customerArray, status);
    end

    function onPrerender()
    end
end
```

Finished JSFHandler for allcustomers2 (note code has also been formatted)

⏭ **Press Ctrl+Shift+F to format your code as shown in the screen capture**

⏭ **Press Ctrl/S when you are done** – and fix any syntax errors (which will display as red circular X's in the editor border)



```
    function onConstruction()
        CustomerLib.GetCstomerListAll(
    end
```

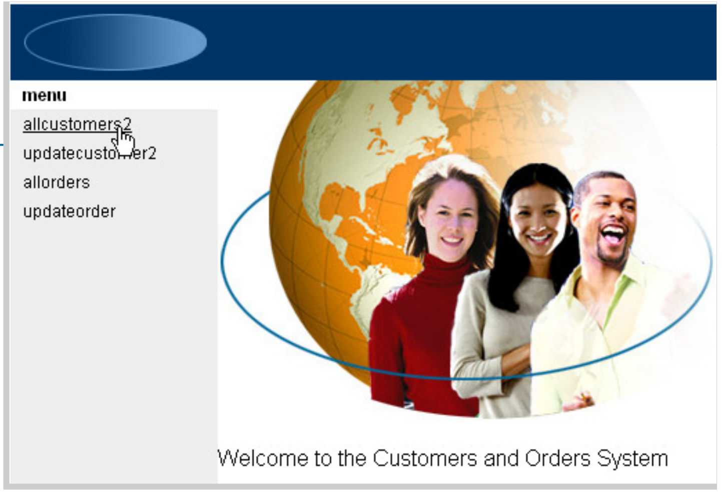
Note that you can mouse-over the Red X-s to obtain additional syntax-related assistance

⏭ 4. Run the page on the server

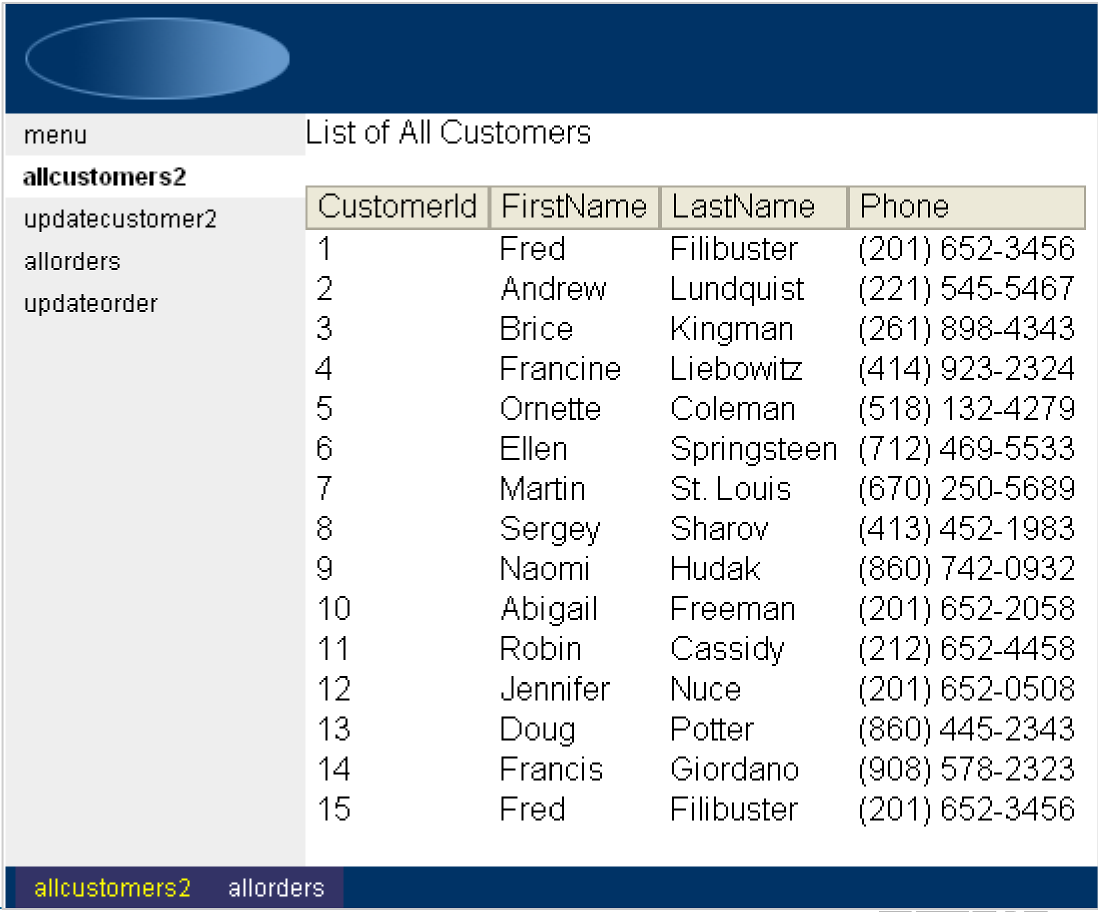
Time to test out what we've done!

4a. (From Project Explorer) Right-click over **menu.jsp** and select: **Run As > Run on Server…**

Save all files when prompted.



4b. Click the menu link to **allcustomers2**



List of All Customers

CustomerId	FirstName	LastName	Phone
1	Fred	Filibuster	(201) 652-3456
2	Andrew	Lundquist	(221) 545-5467
3	Brice	Kingman	(261) 898-4343
4	Francine	Liebowitz	(414) 923-2324
5	Ornette	Coleman	(518) 132-4279
6	Ellen	Springsteen	(712) 469-5533
7	Martin	St. Louis	(670) 250-5689
8	Sergey	Sharov	(413) 452-1983
9	Naomi	Hudak	(860) 742-0932
10	Abigail	Freeman	(201) 652-2058
11	Robin	Cassidy	(212) 652-4458
12	Jennifer	Nuce	(201) 652-0508
13	Doug	Potter	(860) 445-2343
14	Francis	Giordano	(908) 578-2323
15	Fred	Filibuster	(201) 652-3456

So – now that we've done one "list page", let's consolidate what we've learned, and create allorders.

Close all of the pages in the Content Area except for: **Navigation – EGLWeb**

⏭ Create updatecustomer2.jsp

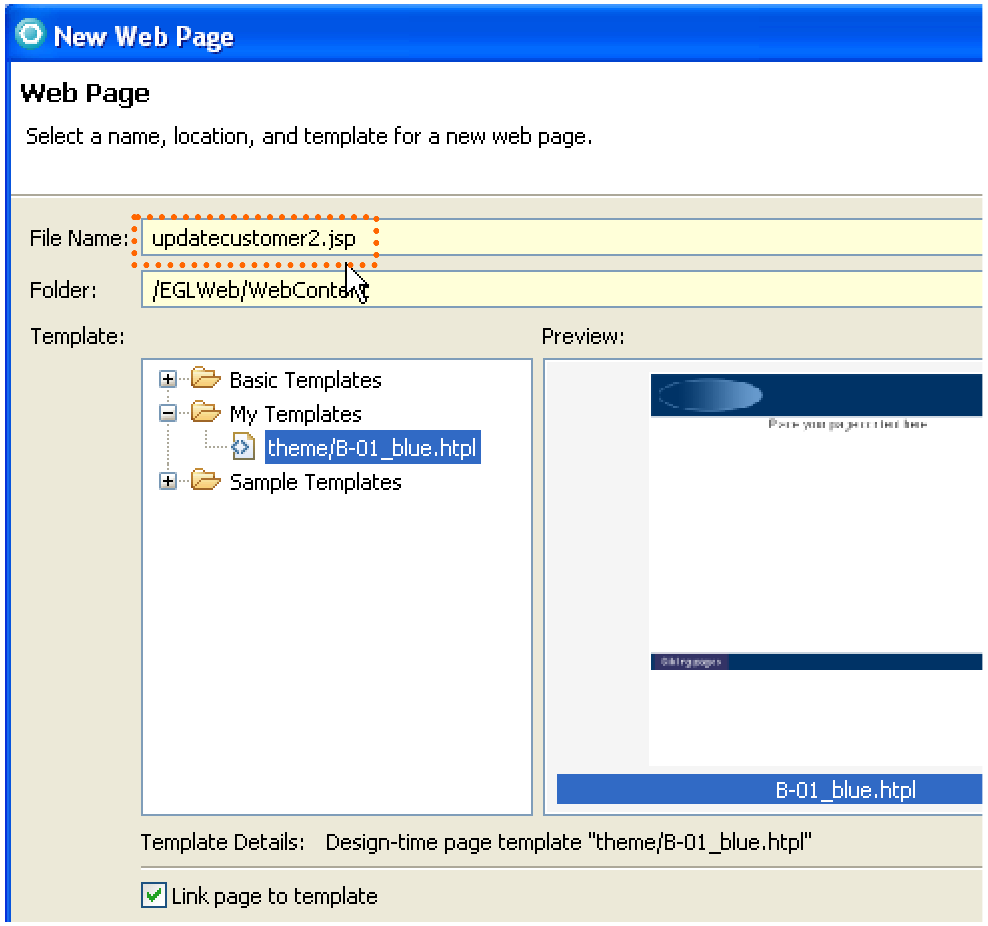
- From the Site Map:
  Double-Click updatecustomer2

- Add the **.jsp** extension to the page name:

  `updatecustomer2.jsp`

- Click **Finish**

- Modify the default heading text, and add a few line breaks.


New Web Page

Web Page

Select a name, location, and template for a new web page.

File Name: updatecustomer2.jsp

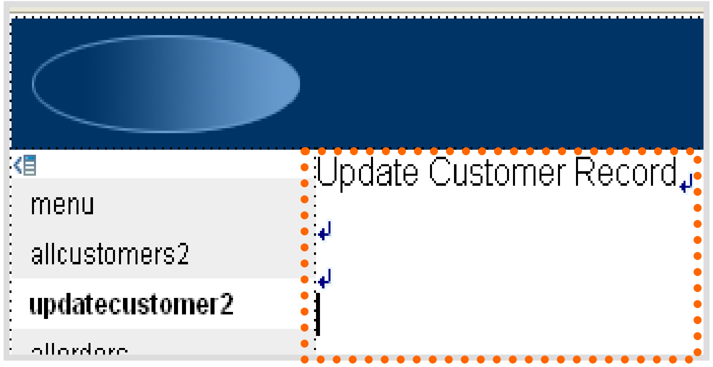
Folder: /EGLWeb/WebContent

Template:

Preview:

Template Details: Design-time page template "theme/B-01_blue.htpl"

☑ Link page to template



⏭ From the Palette, Drag & Drop an EGL Variable on the page

⏭ 2. Add EGL Variables – 2 of 3

2d. From the Create a New EGL Data Variable wizard specify the following (actually, you can probably just take all of the defaults!)

⦿ **Record**

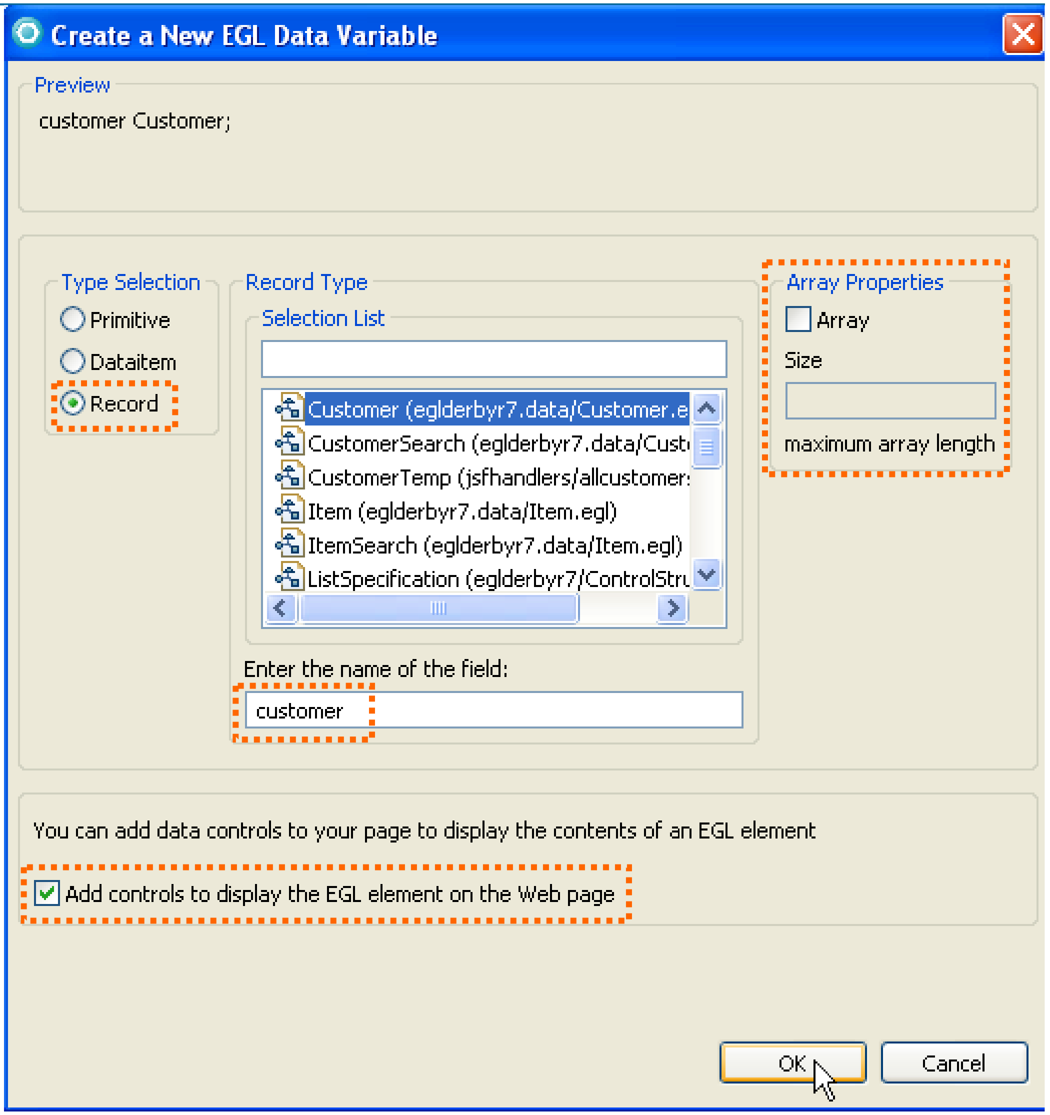
❑ **Un-check Array**

Name the field: **customer**

Leave checked:

☑ **Add controls to display the …**

Click **OK**


Create a New EGL Data Variable

Preview

customer Customer;

Type Selection: Primitive [ ] DataItem [ ] [✓ Record]

Record Type – Selection List:

- Customer (eglderbyr7.data/Customer.e
- CustomerSearch (eglderbyr7.data/Custo
- CustomerTemp (jsfhandlers/allcustomer:
- Item (eglderbyr7.data/Item.egl)
- ItemSearch (eglderbyr7.data/Item.egl)
- ListSpecification (eglderbyr7/ControlStru

Array Properties: ☐ Array

Size: (maximum array length)

Enter the name of the field: customer

You can add data controls to your page to display the contents of an EGL element

☑ Add controls to display the EGL element on the Web page

⏭ 2. Add EGL Variables – 3 of 3

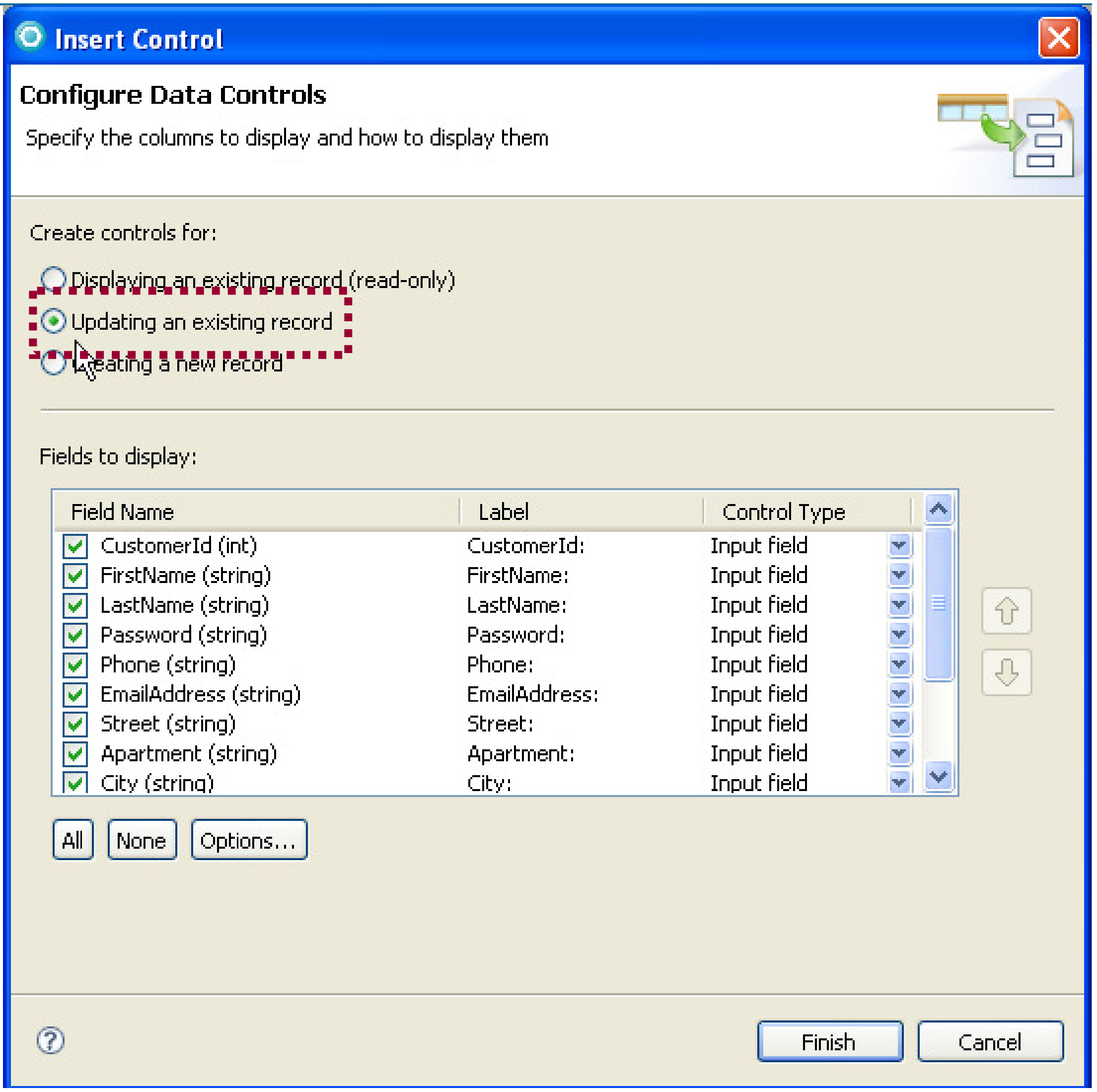
2e. From Configure Data Controls …specify the following

⊙ **Updating an existing record**

Display all of the columns
(Leave them all ☑ checked)

- Click **Options…** and ensure that at least one ☑ **Submit Button** is checked

- Click **Finish**



Insert Control

Configure Data Controls

Specify the columns to display and how to display them

Create controls for:

( ) Displaying an existing record (read-only)
(●) Updating an existing record
( ) Creating a new record

Fields to display:

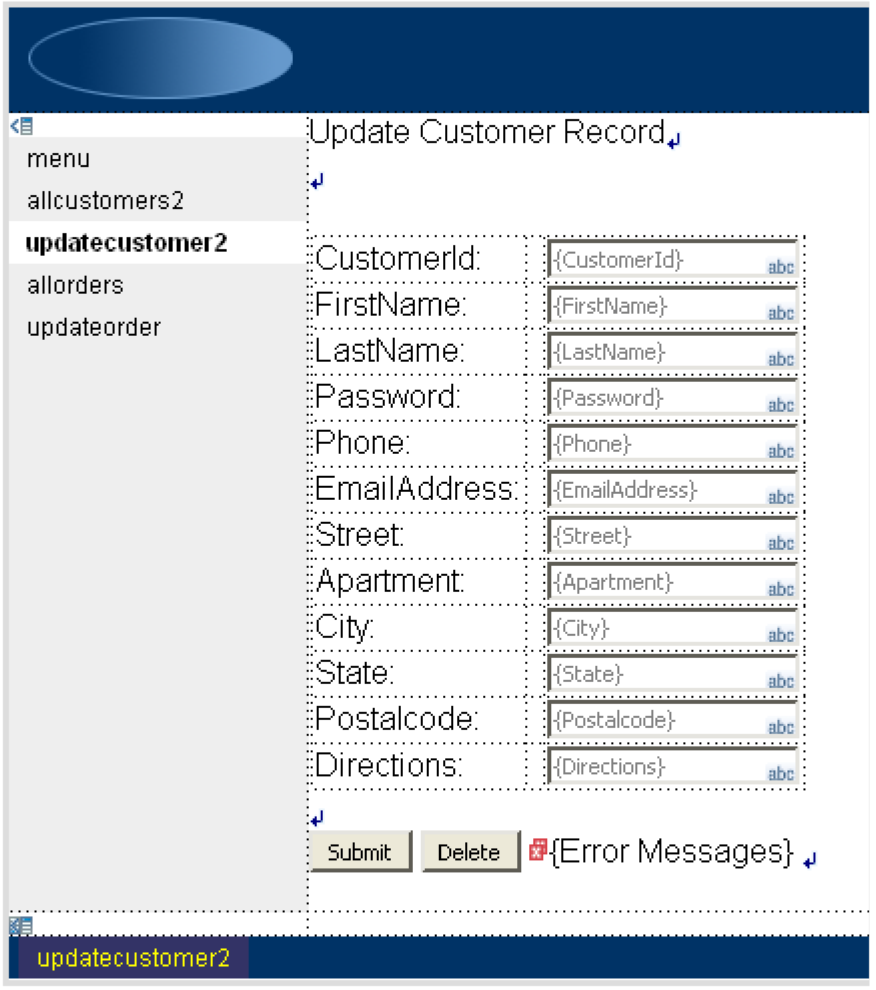
	Field Name	Label	Control Type
✓	CustomerId (int)	CustomerId:	Input field
✓	FirstName (string)	FirstName:	Input field
✓	LastName (string)	LastName:	Input field
✓	Password (string)	Password:	Input field
✓	Phone (string)	Phone:	Input field
✓	EmailAddress (string)	EmailAddress:	Input field
✓	Street (string)	Street:	Input field
✓	Apartment (string)	Apartment:	Input field
✓	City (string)	City:	Input field

⏭ 2. Finished – Adding EGL Variables

👓 The default Page Design layout for a non-array record is a vertically spaced, form with submit buttons.

- Let's add the code to make this page run.

⏭ Right-click over the page, and select: **Edit Page Code**



Update Customer Record

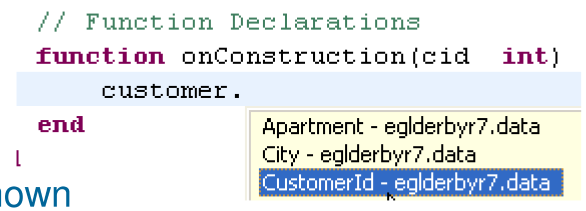
CustomerId:	{CustomerId}
FirstName:	{FirstName}
LastName:	{LastName}
Password:	{Password}
Phone:	{Phone}
EmailAddress:	{EmailAddress}
Street:	{Street}
Apartment:	{Apartment}
City:	{City}
State:	{State}
Postalcode:	{Postalcode}
Directions:	{Directions}

Submit Delete {Error Messages}

⏭ 3. Add EGL Business and Back-End Logic

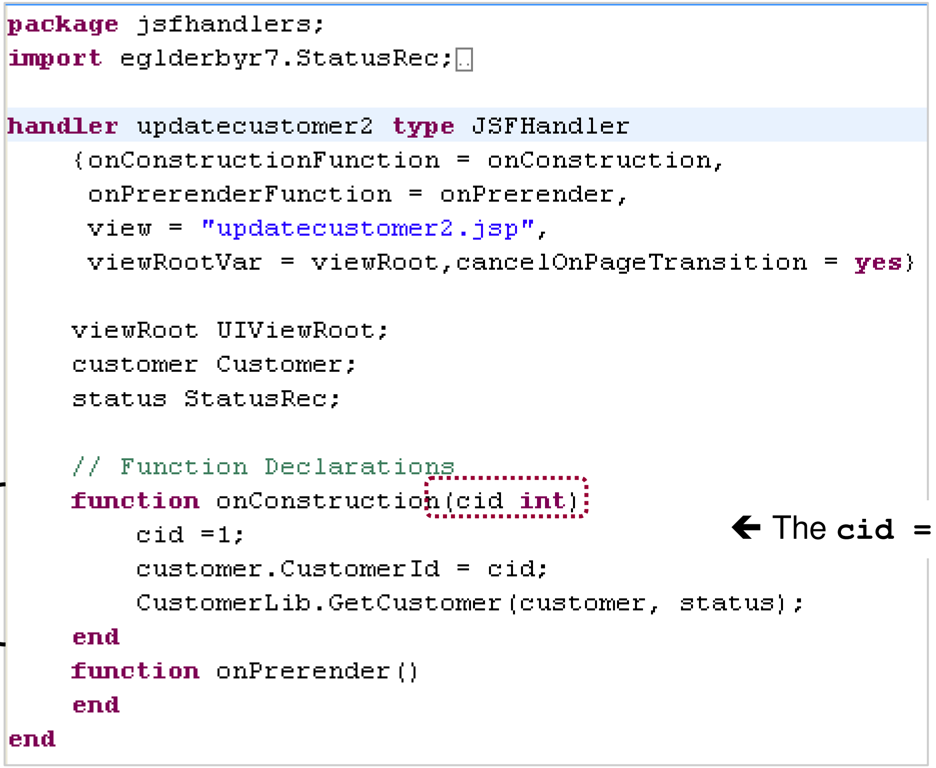
As shown in the finished JSFHandler code below, **use Content Assist***** *(Ctrl/Spacebar)* to add the following (save with Ctrl/S when done):

1. A JSFHandler property: `cancelOnPageTransition = yes`
2. A `status` record variable declaration
3. Modify the `onConstruction` function, add the parameter: `cid` as shown
4. Add an assignment statement, to hard-codes `cid's` value to 1
5. And an assignment statement to `customer.customerID` variable
6. Finally, a call to the `CustomerLib.GetCustomer(…)` function



```
// Function Declarations
function onConstruction(cid  int)
    customer.
end
```

Content Assist



```
package jsfhandlers;
import eglderbyr7.StatusRec;

handler updatecustomer2 type JSFHandler
    {onConstructionFunction = onConstruction,
     onPrerenderFunction = onPrerender,
     view = "updatecustomer2.jsp",
     viewRootVar = viewRoot,cancelOnPageTransition = yes}

    viewRoot UIViewRoot;
    customer Customer;
    status StatusRec;

    // Function Declarations
    function onConstruction(cid int)
        cid =1;
        customer.CustomerId = cid;
        CustomerLib.GetCustomer(customer, status);
    end
    function onPrerender()
    end
end
```

🡐 The `cid = 1;` statement is here so that we can test the page standalone

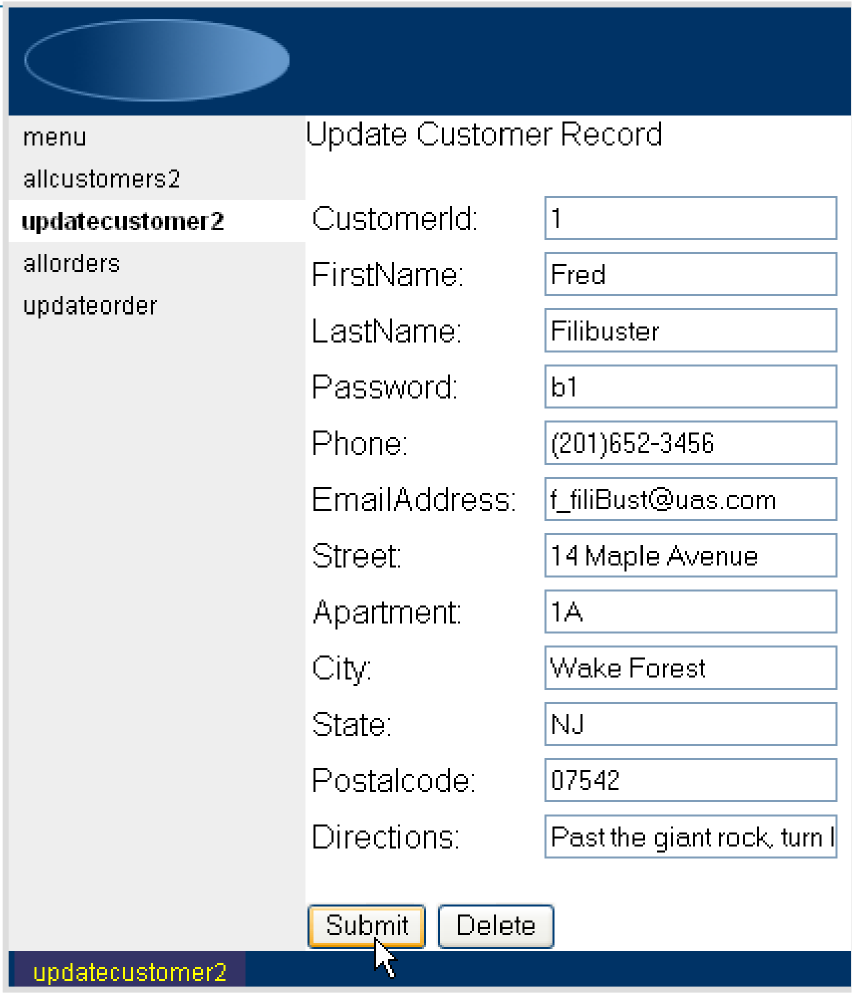
*** Note: By using Content Assist the **import** statements needed for all this code will automatically be added to your JSFHandler!

⏭ 4. Run the Page on the Server

Time to test out what we've done!

(From Project Explorer)

- Right-click over **updatecustomer2.jsp** and
- Select: **Run As > Run on Server…**



Update Customer Record

CustomerId:	1
FirstName:	Fred
LastName:	Filibuster
Password:	b1
Phone:	(201)652-3456
EmailAddress:	f_filiBust@uas.com
Street:	14 Maple Avenue
Apartment:	1A
City:	Wake Forest
State:	NJ
Postalcode:	07542
Directions:	Past the giant rock, turn l

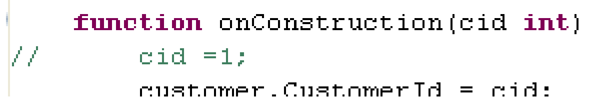
But we have a lot of work left to do to this page.

- Cosmetic page design enhancements
- EGL (business logic) and more.

Let's get started by commenting out the statement in the JSFHandler that hard-codes `cid` to a 1 value

⏭ Right-Click over updatecustomer2.jsp and select **Edit Page Code**

⏭ **Type two forward slashes** in front of the `//cid = 1;` statement



```
    function onConstruction(cid int)
//      cid =1;
        customer.CustomerId = cid;
```

5. Refine the U.I. Model

👓 So we're ready now after doing the basic Web Site and EGL work to make the pages ready to show our users.

- Here's what's left to be done – you will:
  - Add a link from the list to the detail pages
  - Refine the U.I. for all of the dynamic content pages
  - Refine the business logic for the update page
    - Add update and delete data access functions
    - Add business logic calls to our data validation routine

- Sounds like a lot, but it's all straightforward work. We'll start by adding a link from **allcustomers2** to **updatecustomer2**, to pass a clicked row value from the list to the detail page, so users can select the row they want to update.

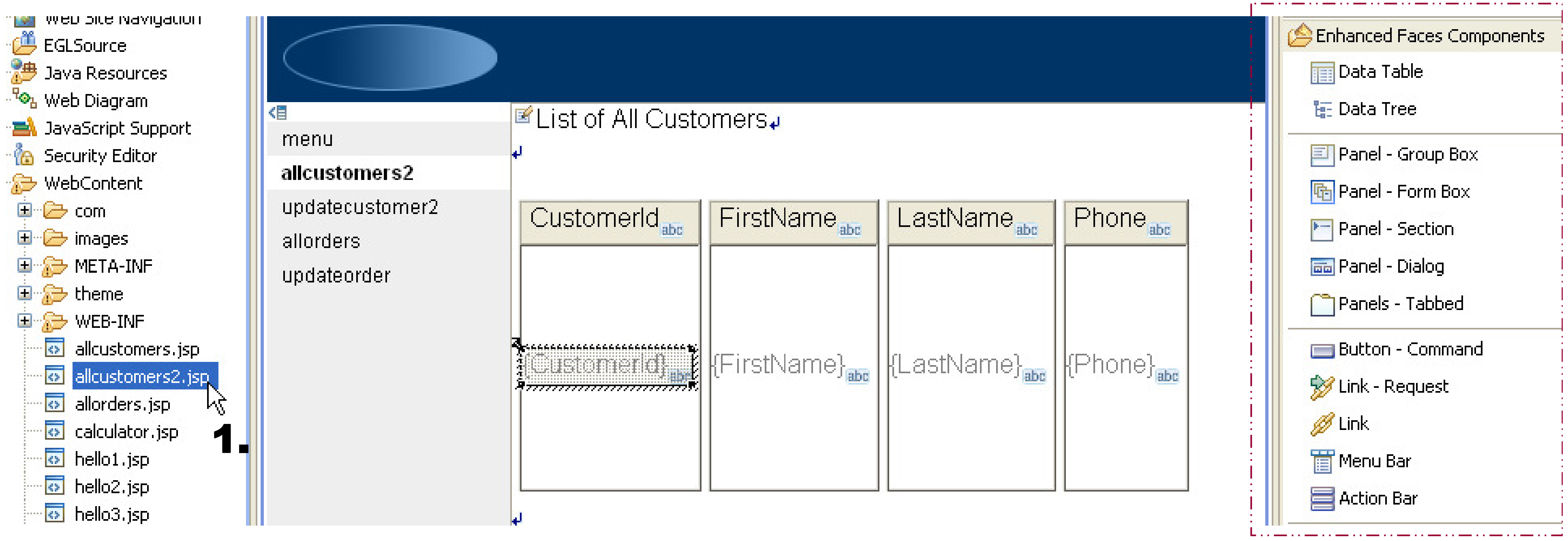
⏭ Adding a Dynamic Content Hyperlink – 1 of 3

(From Project Explorer)

1. Double-Click the **allcustomers2.jsp**

(From the Palette)

2. Open the Enhanced Faces Components drawer



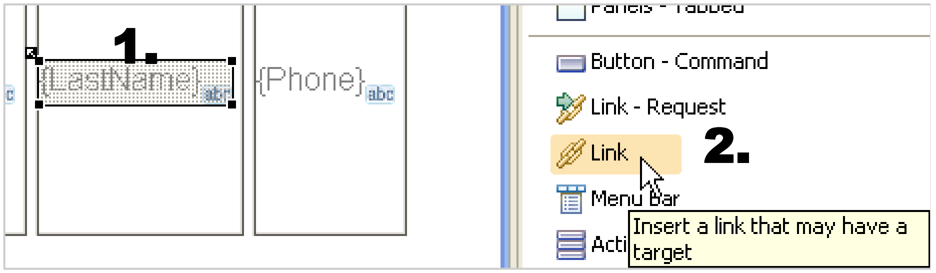
⏭ Adding a Dynamic Content Hyperlink – 2 of 3

(From Page Designer)

1. (Left) Single-Click the **lastName** control in the middle of the column

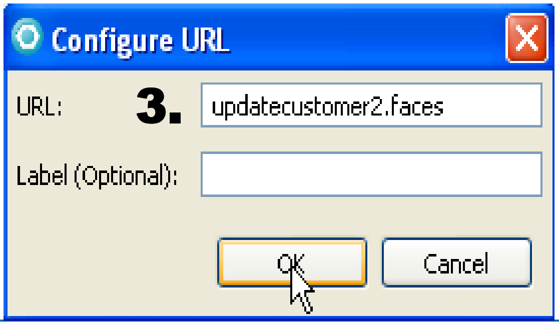
(From the Palette)

2. Double-Click a Link component



(From Configure URL)

3. Enter the URL: **updatecustomer2.faces**

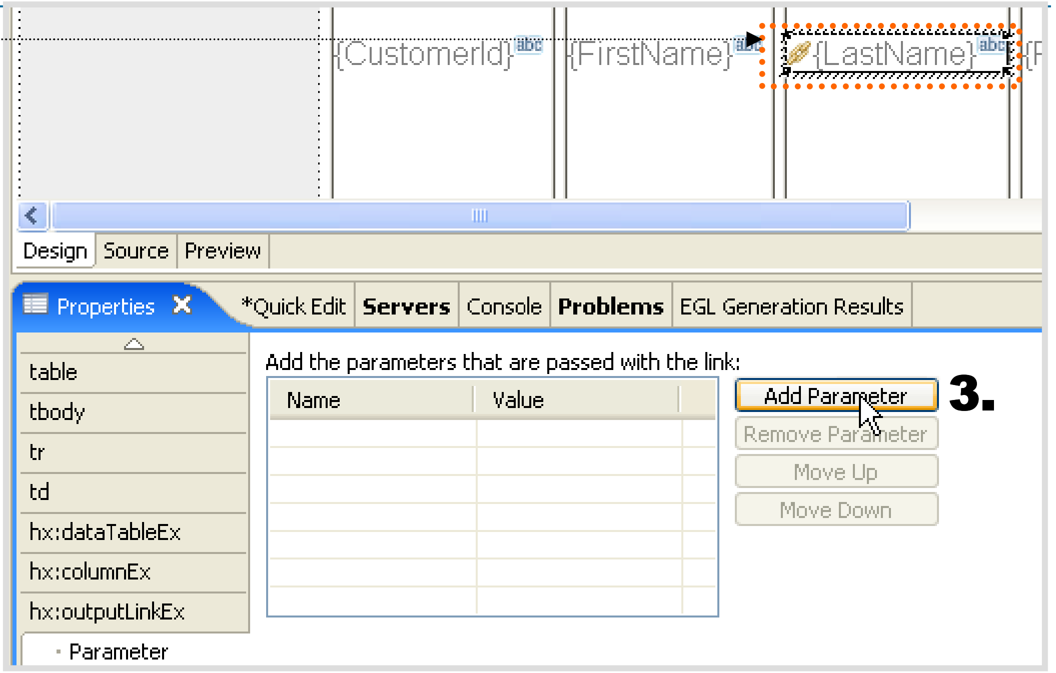


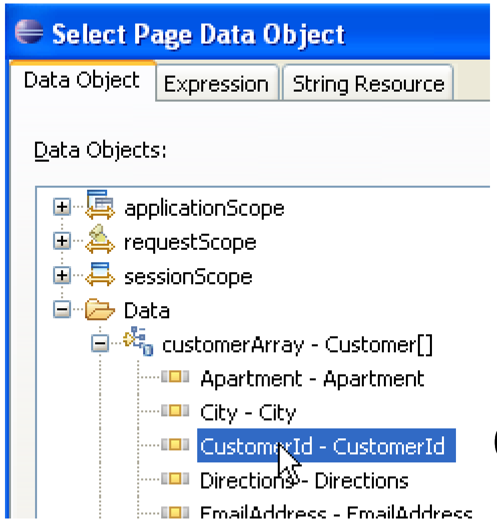
Click **OK**

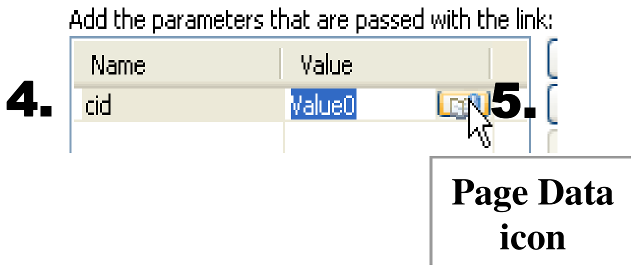
⏭ Adding a Dynamic Content Hyperlink – 3 of 3

With the **link** next to LastName selected:

1. Open the **Properties View**
2. Click the – **Parameter** tab
3. Click: **Add Parameter**
4. Name the new Parameter: **`cid`**
5. Tab into then click the **Page Data icon**
6. From Select Page Data Object, expand **customerArray** and select: **`CustomerId – CustomerId`**

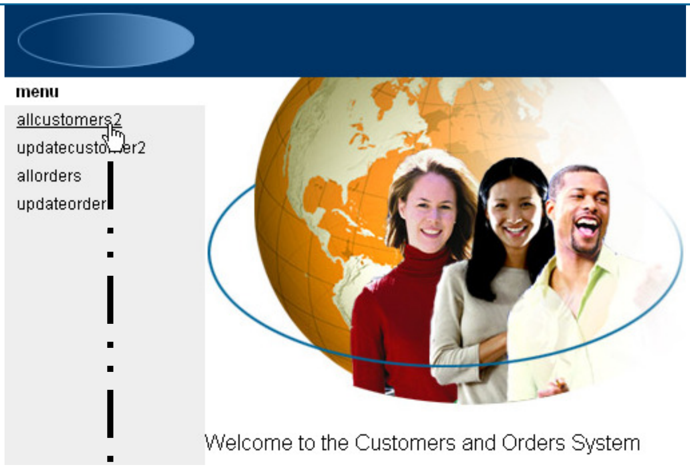






7. Click **OK**

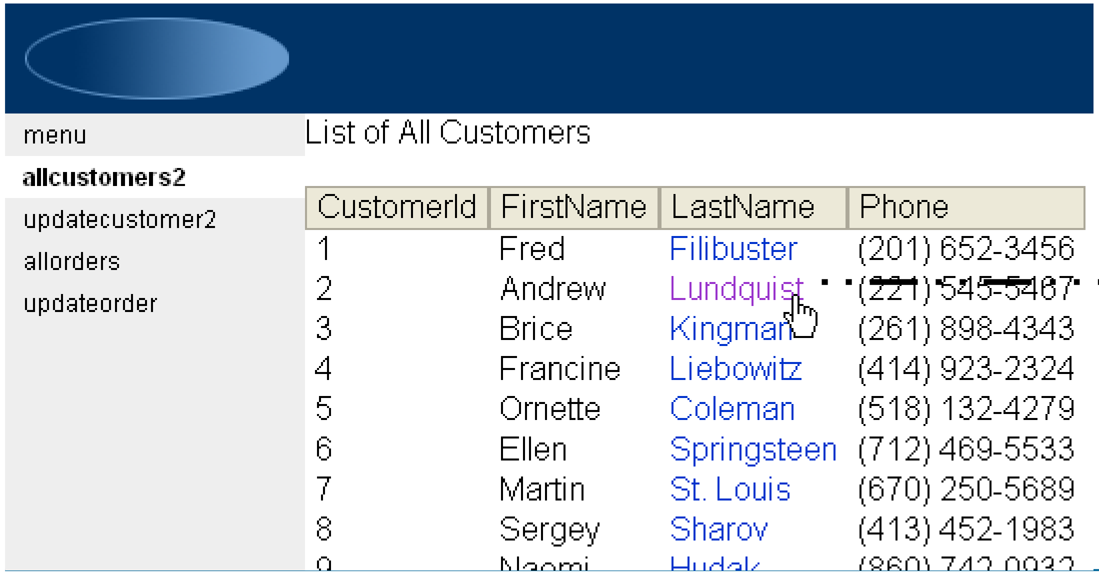
⏭ Test Your Work



Welcome to the Customers and Orders System

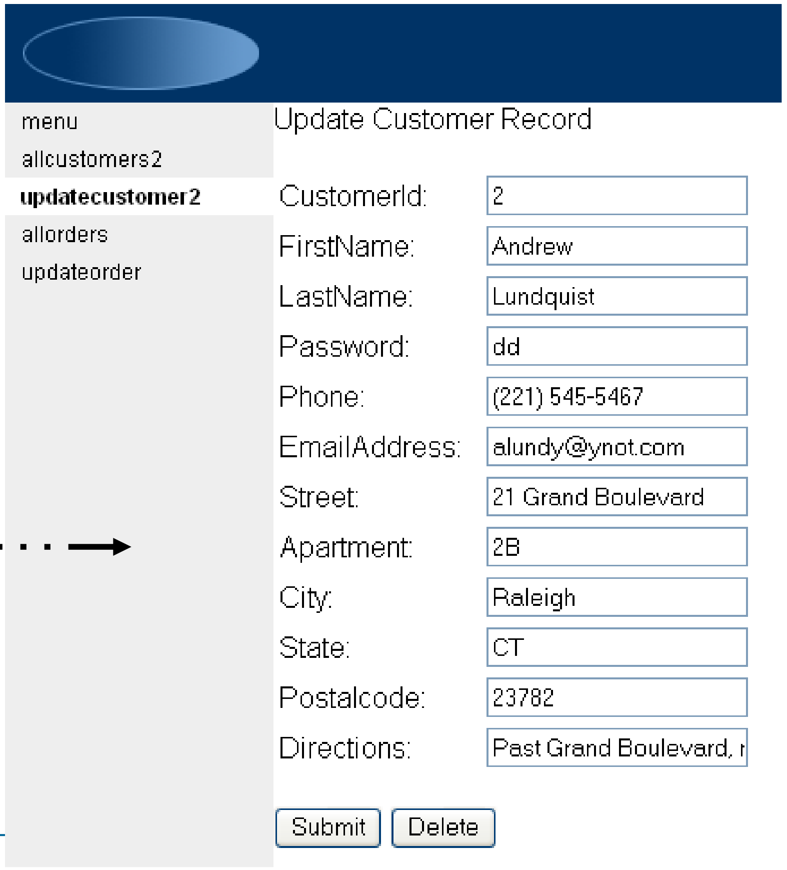
(From Project Explorer)

- Right-Click over **menu.jsp** and select **Run As > Run on Server…**
- From the menu, launch **allcustomers2**
- From allcustomers2, select (click) a hyper-linked LastName



List of All Customers

CustomerId	FirstName	LastName	Phone
1	Fred	Filibuster	(201) 652-3456
2	Andrew	Lundquist	(221) 545-5467
3	Brice	Kingman	(261) 898-4343
4	Francine	Liebowitz	(414) 923-2324
5	Ornette	Coleman	(518) 132-4279
6	Ellen	Springsteen	(712) 469-5533
7	Martin	St. Louis	(670) 250-5689
8	Sergey	Sharov	(413) 452-1983
9	Naomi	Hudak	(860) 742-0932



Update Customer Record

CustomerId:	2
FirstName:	Andrew
LastName:	Lundquist
Password:	dd
Phone:	(221) 545-5467
EmailAddress:	alundy@ynot.com
Street:	21 Grand Boulevard
Apartment:	2B
City:	Raleigh
State:	CT
Postalcode:	23782
Directions:	Past Grand Boulevard, r

5. Refine the U.I. Model – for Dynamic Content Pages

So, there's still some left to be done to our prototype. We need to add more EGL business and data access logic, and we need to make the pages look a little nicer. We'll start by doing the latter, and then wrap back around on the EGL coding for database updates.

- **List pages** (allcustomers2 and allorders)**:**
  - Add a border to the data tables on both pages.
  - Add alternate row colors
  - Add paging (for allcustomers2)
    - Add inline update form for allcustomers2 (OPTIONAL)
  - Row Categorization (for allorders)

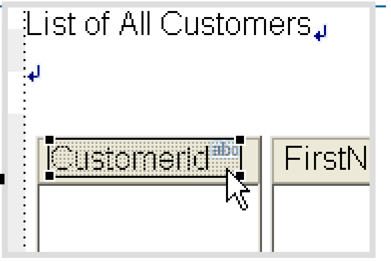
- **For the detail page we'll:**
  - Make the key field read-only (so users aren't tempted to update it)
  - Change the color of the cells and text justification

⏭ Adding a Border to a dataTable

(From Project Explorer: Open allcustomers2.jsp into the Content Area)

1. **Click a column header**, and **press the up arrow key on your PC** 🡅
   - This selects the entire dataTable and makes its properties available

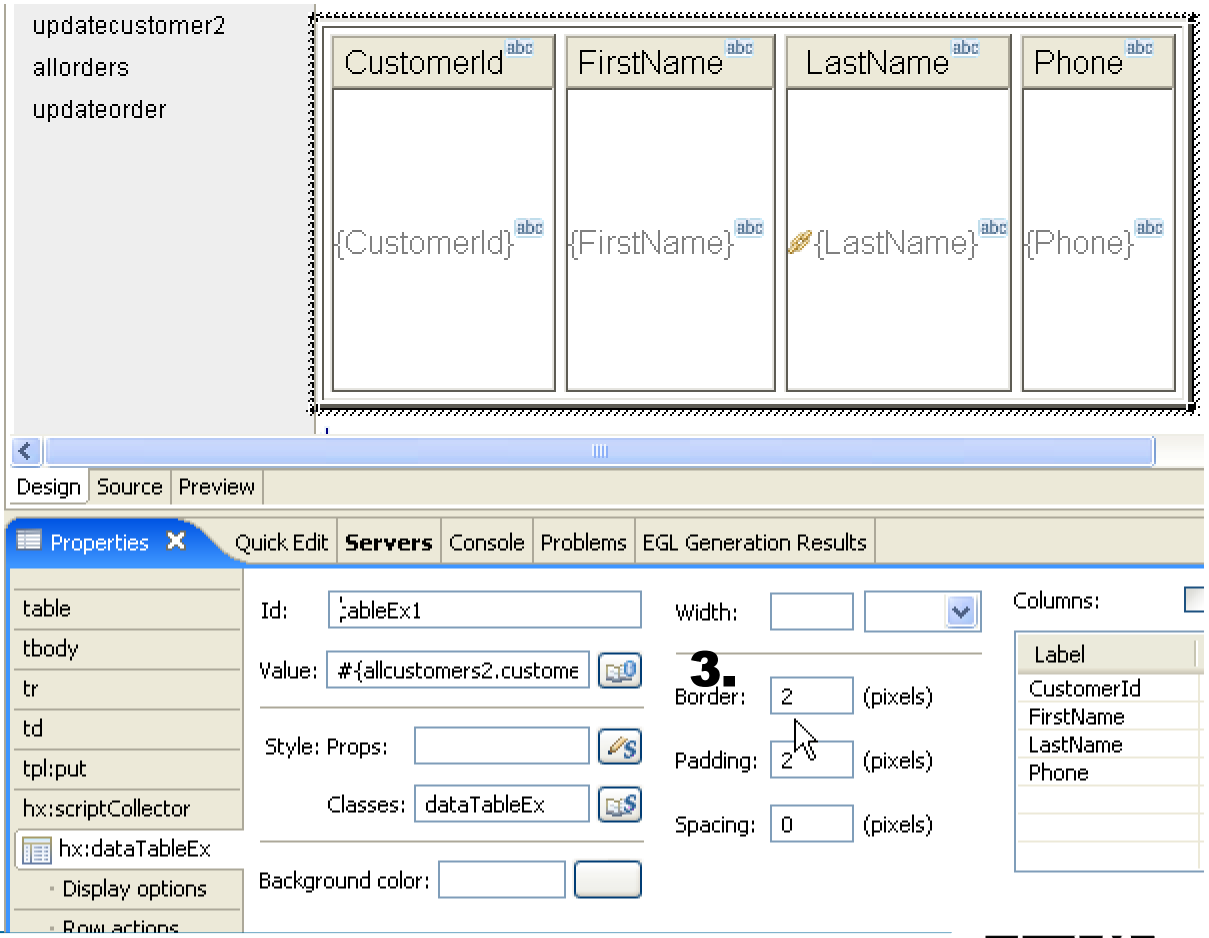
1.



2. Click the **Properties View tab**
3. Specify **Border: 2**

Run the page on the server, to see the effect of adding a border.

2.



Id: tableEx1
Value: #{allcustomers2.custome
Style: Props:
Classes: dataTableEx
Background color:
Width:
Border: 2 (pixels)
Padding: 2 (pixels)
Spacing: 0 (pixels)

⏭ Adding Alternate Row Colors to a JSF dataTable

(Still editing allcustomers2.jsp into the Content Area – with the entire dataTable selected)

1. **(From Properties) click the - Display options tab**
2. **Delete the content of Column classes:**

Run the page to see results of alternate row colors

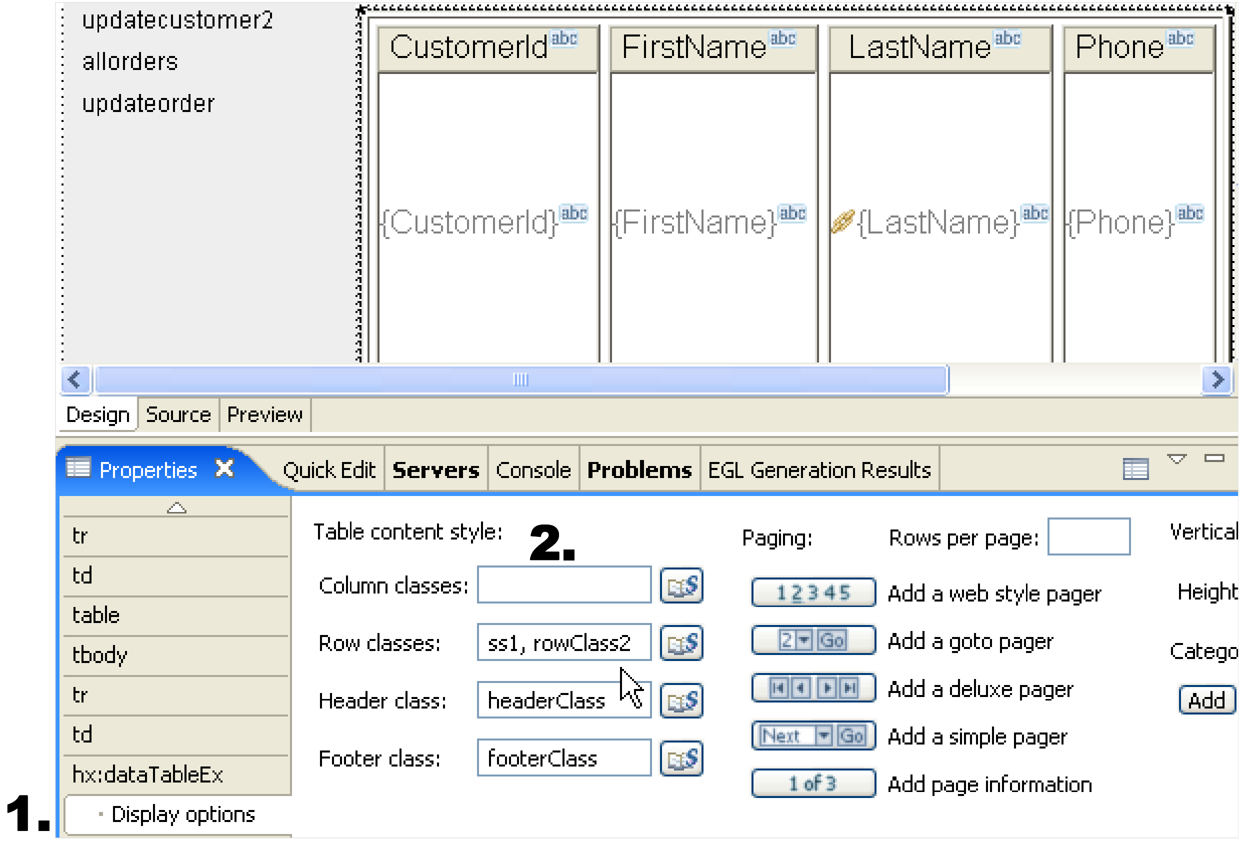


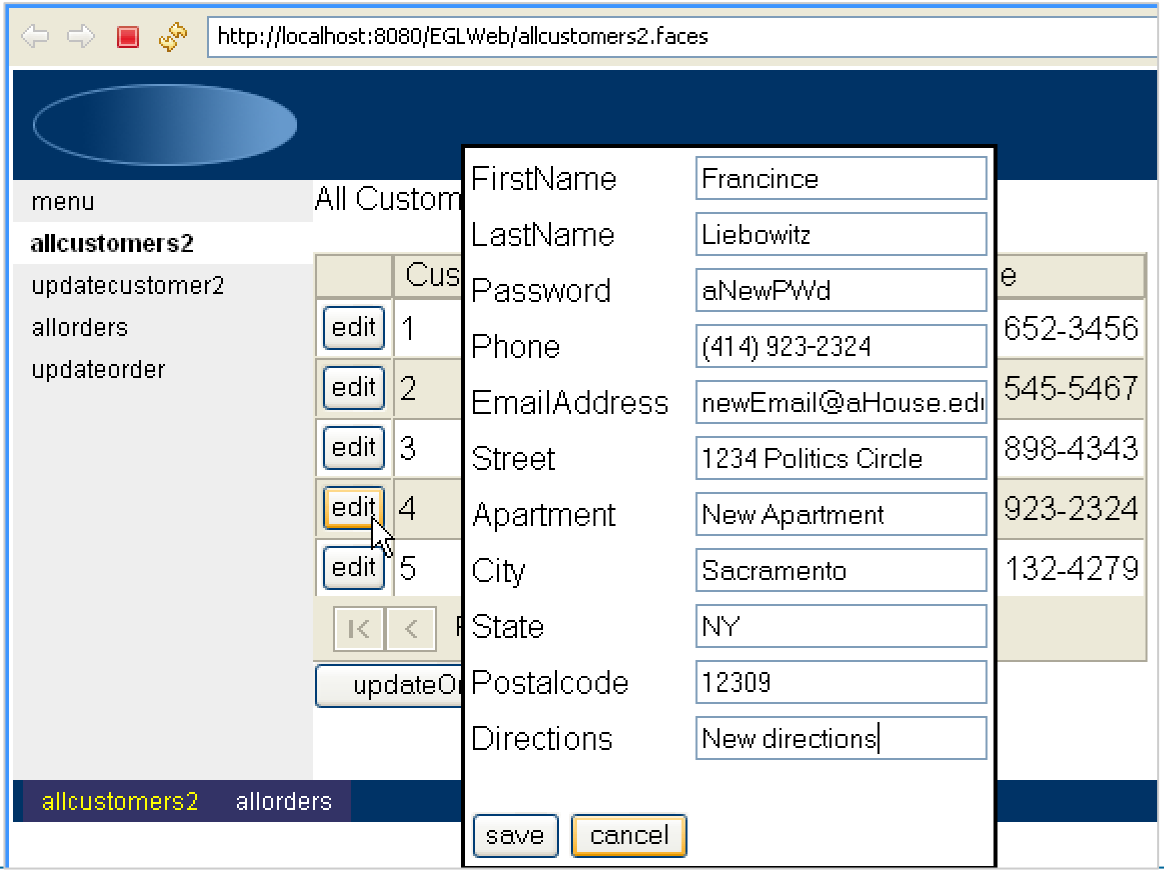
Table content style:

Column classes:	
Row classes:	ss1, rowClass2
Header class:	headerClass
Footer class:	footerClass

(OPTIONAL WORKSHOP) Add an Inline Form to a dataTable

You might wish to allow users to actually update a customer record "inline" from the .JSP page's dataTable (as shown below). This useful web U.I. technique is not that hard to achieve using EGL and JSF technology. However:

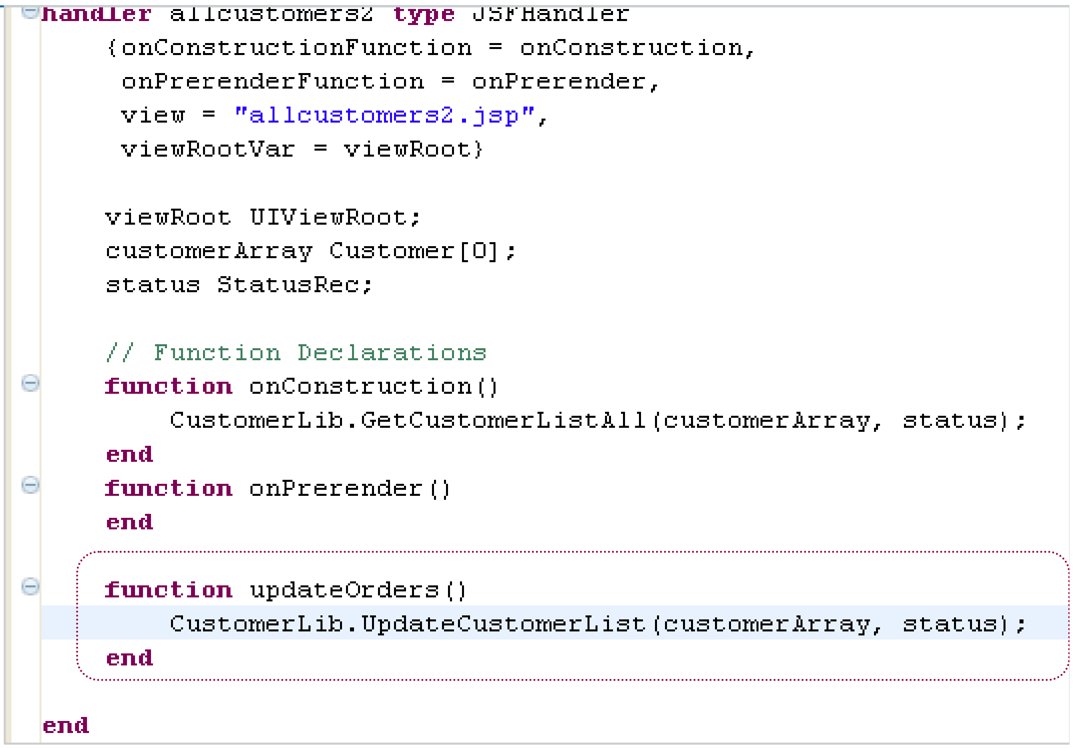
1. Please consider this workshop optional – finish it only if you're ahead of schedule in the class (it will take 10 – 15 minutes)
2. If you do complete this workshop, your allcustomers2 functionality will not match the course's (as we are not expecting everyone to accomplish this). And screen-captures might not match, etc.
3. You can always back out your changes by pressing Ctrl/Z – repeatedly


http://localhost:8080/EGLWeb/allcustomers2.faces

FirstName	Francince
LastName	Liebowitz
Password	aNewPWd
Phone	(414) 923-2324
EmailAddress	newEmail@aHouse.edu
Street	1234 Politics Circle
Apartment	New Apartment
City	Sacramento
State	NY
Postalcode	12309
Directions	New directions

⏭ (**OPTIONAL** Lab) Adding an Inline Form to a dataTable – 1 of 6

(If not still , open **allcustomers2.egl** in the Content Area)



```
handler allcustomers2 type JSFHandler
    {onConstructionFunction = onConstruction,
     onPrerenderFunction = onPrerender,
     view = "allcustomers2.jsp",
     viewRootVar = viewRoot}

    viewRoot UIViewRoot;
    customerArray Customer[0];
    status StatusRec;

    // Function Declarations
    function onConstruction()
        CustomerLib.GetCustomerListAll(customerArray, status);
    end
    function onPrerender()
    end

    function updateOrders()
        CustomerLib.UpdateCustomerList(customerArray, status);
    end

end
```

- Using Content Assist, add the following updateOrders() function and call to your CustomerLib.UpdateCustomerList(…) function (which was created by the Data Access Wizard)

- **Save (Ctrl/S)** your coding – and remove any/all syntax errors in your EGL source.

⏭ (OPTIONAL Lab) Adding an Inline Form to a dataTable – 2 of 6

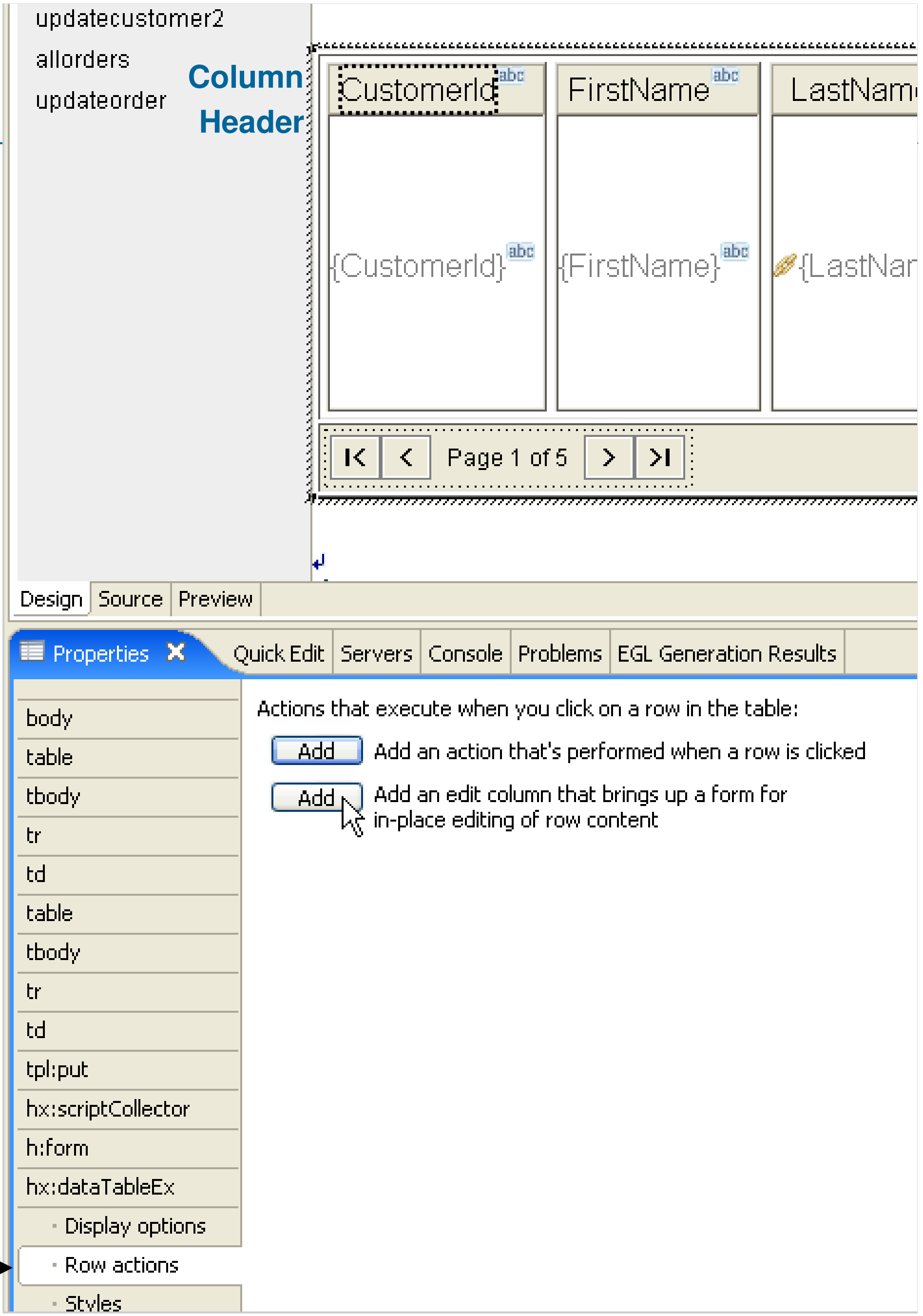
(Return to work on **allcustomers2.jsp** in the Content Area)

1. **Click a column header in the dataTable, and press the up arrow key** on your PC:
   - this selects the entire dataTable

(From **Properties** / **- Row actions**)

Click: **Add an edit column that brings up a form for in-place editing of row content**

This will add a new column to the left of your dataTable, with an area in which you can drag and drop an EGL variable



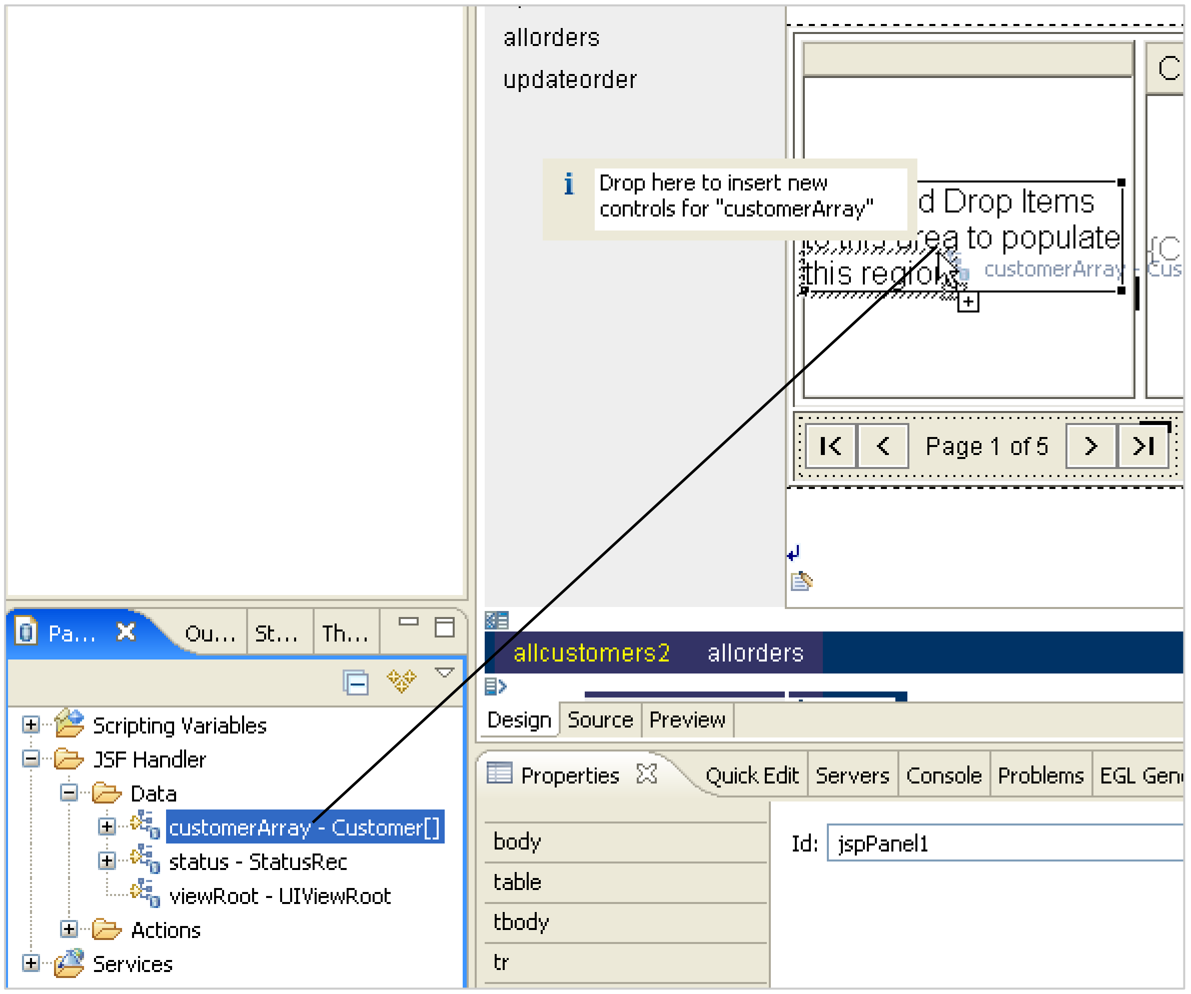
Actions that execute when you click on a row in the table:

Add an action that's performed when a row is clicked

Add an edit column that brings up a form for in-place editing of row content

⏭ (OPTIONAL Lab) Adding an Inline Form to a dataTable – 3 of 6

(From Page Data) **Select** (left-click and hold), then **Drag and Drop customerArray into the inline form area -** *this will bring up Configure Data Controls*



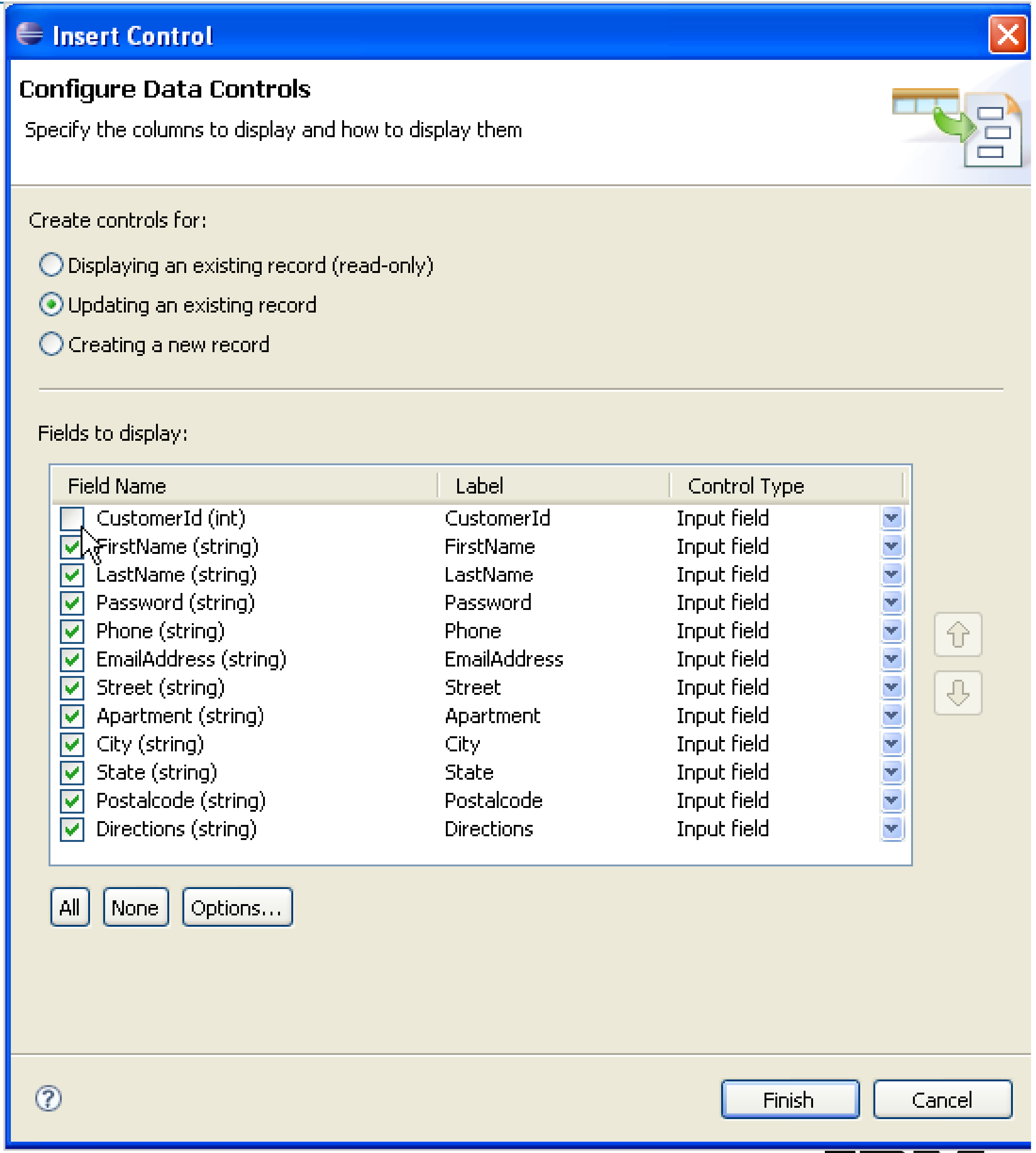
⏭ (OPTIONAL Lab) Adding an Inline Form to a dataTable – 4 of 6

(From Configure Data Controls)

Specify: ⦿ **Updating an existing record**

Un-check: **CustomerId**

Click: **Finish**



Configure Data Controls

Specify the columns to display and how to display them

Create controls for:

( ) Displaying an existing record (read-only)
[✓ Updating an existing record]
( ) Creating a new record

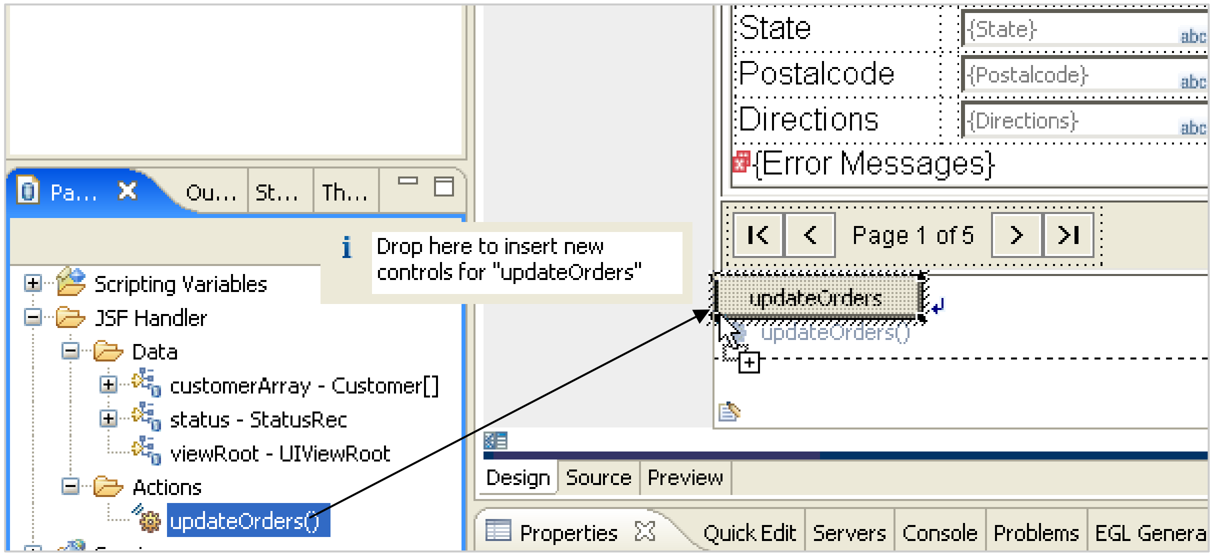
Fields to display:

	Field Name	Label	Control Type
☐	CustomerId (int)	CustomerId	Input field
✓	FirstName (string)	FirstName	Input field
✓	LastName (string)	LastName	Input field
✓	Password (string)	Password	Input field
✓	Phone (string)	Phone	Input field
✓	EmailAddress (string)	EmailAddress	Input field
✓	Street (string)	Street	Input field
✓	Apartment (string)	Apartment	Input field
✓	City (string)	City	Input field
✓	State (string)	State	Input field
✓	Postalcode (string)	Postalcode	Input field
✓	Directions (string)	Directions	Input field

⏭ (OPTIONAL Lab) Adding an Inline Form to a dataTable – 5 of 6

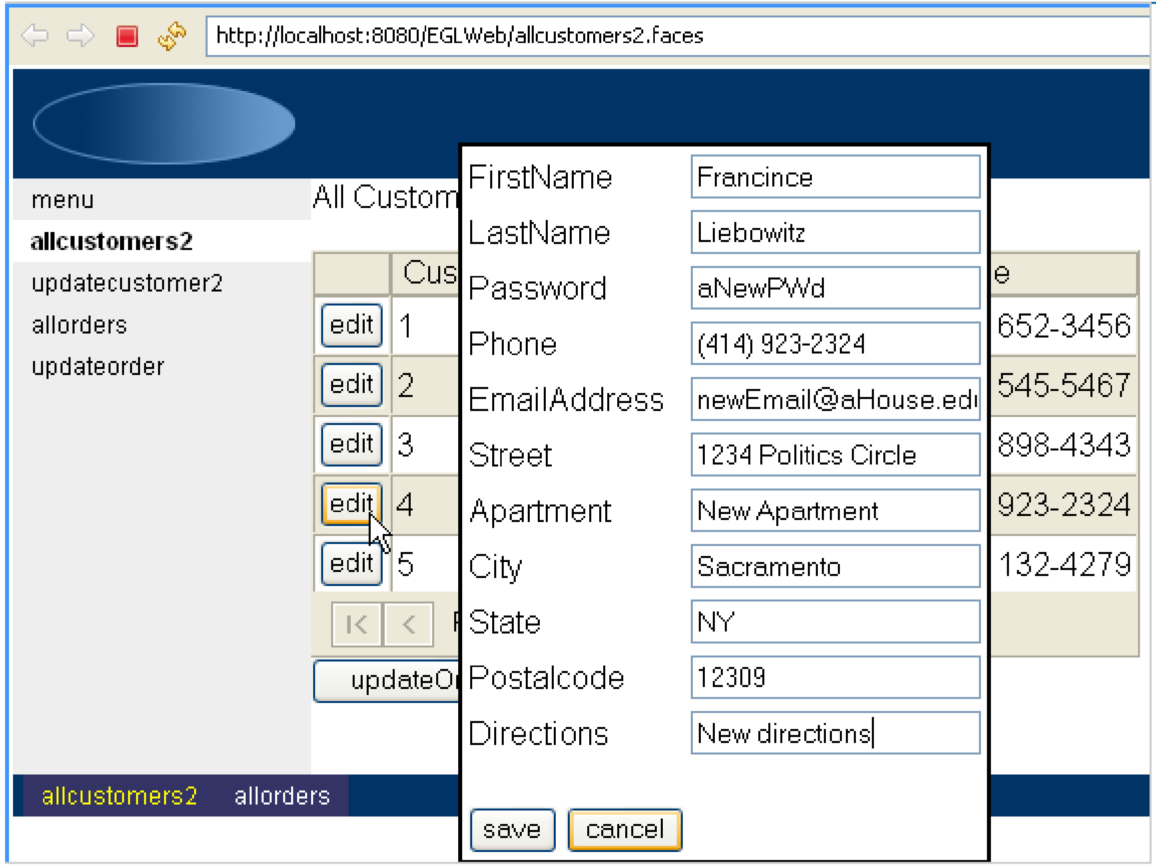
(From Page Data)

- **Select** (left-click and hold) then **Drag and Drop the updateOrders()** onto the page
  - **This will create a submit button control – "bound" to the updateOrders() EGL function**



Drop here to insert new controls for "updateOrders"

⏭ (OPTIONAL Lab) Adding an Inline Form to a dataTable – 6 of 6



FirstName	Francince
LastName	Liebowitz
Password	aNewPWd
Phone	(414) 923-2324
EmailAddress	newEmail@aHouse.edı
Street	1234 Politics Circle
Apartment	New Apartment
City	Sacramento
State	NY
Postalcode	12309
Directions	New directions

- **Run the page on the server**
- Click a row's **edit** button
- Modify some data value(s) in the form, and click the **save** button to test this new functionality.
  - Try making changes to one of the rows and clicking cancel as well.
- Click the **updateOrders** button – to save your edits to the database
- Re-run **allcustomers2**

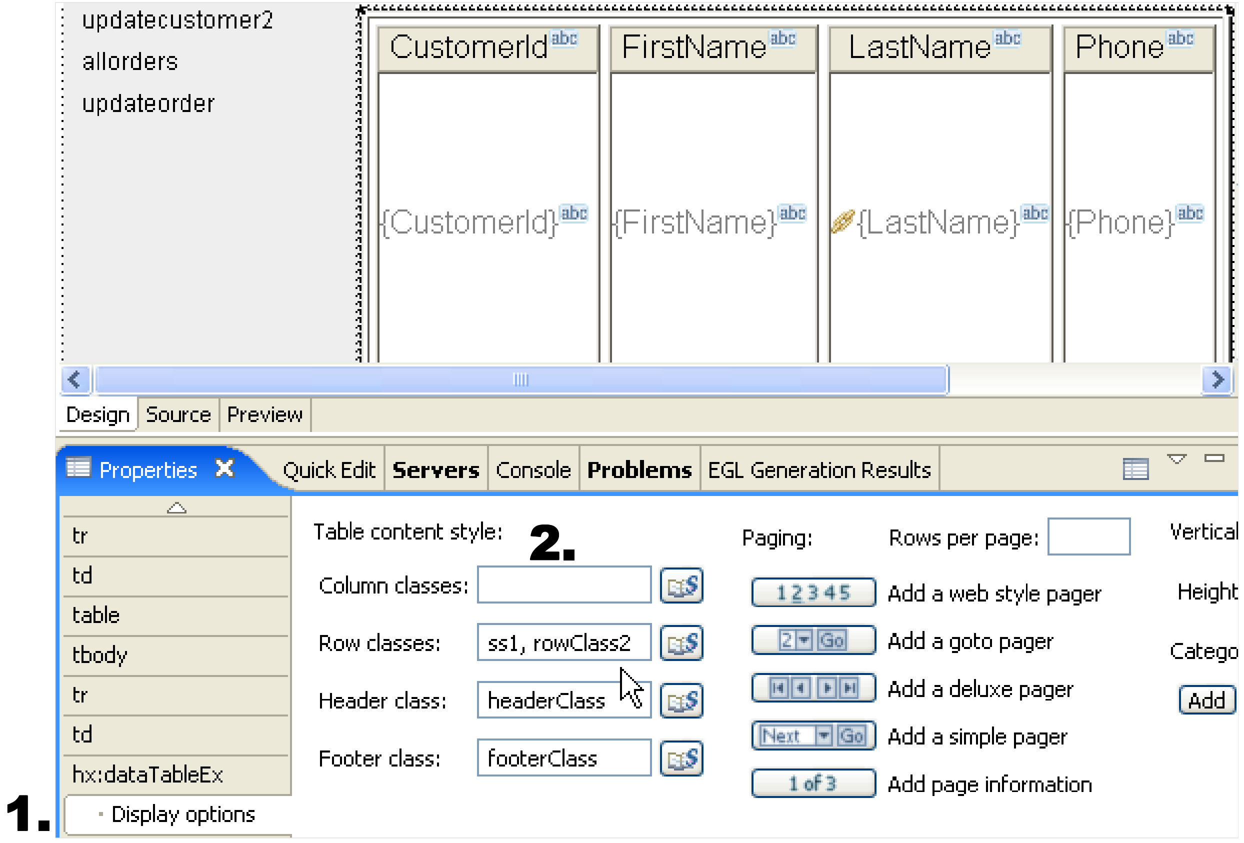
⏭ Adding Alternate Row Colors to a JSF dataTable

(Still working on allcustomers2.jsp into the Content Area – with the entire dataTable selected)

1. **(From Properties) click the - Display options tab**

2. **Delete the content of Column classes:**

Run the page to see results of alternate row colors



updatecustomer2
allorders
updateorder

CustomerId	FirstName	LastName	Phone
{CustomerId}	{FirstName}	{LastName}	{Phone}

Design | Source | Preview

Properties | Quick Edit | Servers | Console | Problems | EGL Generation Results

tr
td
table
tbody
tr
td
hx:dataTableEx
1. Display options

Table content style: 2.

Column classes:	
Row classes:	ss1, rowClass2
Header class:	headerClass
Footer class:	footerClass

Paging: Rows per page:

Add a web style pager
Add a goto pager
Add a deluxe pager
Add a simple pager
Add page information

Topic

EGL and Remote Programs

Sub-Topics:

- EGL Calls to Remote Programs (Overview)
- Calling Java (terms and concepts)
- Calling Java using ExternalType
- Calling Java using JavaLib
- Calling COBOL Programs
- **Calling RPG Programs**

Calling RPG Programs

- In this section you will learn how to call RPG programs running on an IBM iSeries server.
- This lab requires you to have open Internet (TCP/IP) access on your PC
- You will be calling (up to) four RPG transactions, running in an IBM System i in Dallas/Fort Worth.
- The steps you learn here should be generalize-able to your specific shop's enablement

EGL Logic Part
call program…

Here's what you'll do:

1. Add the iSeries **JT400** toolkit to your project
2. (For each RPG program to call) **Add to the EGL Linkage Part for your Build File**
3. Create the business logic to call the RPG Program

First we will start with an overview of the system architecture and process ➔

Calling RPG Programs - Overview

- Calling a RPG, COBOL, CL, or other program on the System i from EGL is as simple as coding:

```
call "myProgram"  (parm1, parmN);
```

- The facility is however extremely flexible and supports the many options that may be required to support many possible options and configurations. EGL takes into account the many possible runtime options that may affect how you run your jobs and call the RPG programs.

- EGL uses the System i Toolbox for Java™ to call programs on the System i. The toolbox in turn uses the i5/OS Remote Command server that is a part of the i5/OS Host Servers.

- The Remote Command server in i5/OS consists of a server daemon program that listens for TCP/IP requests from "clients". This program is QZRCSRVSD which runs in the QSYSWRK sub-system.

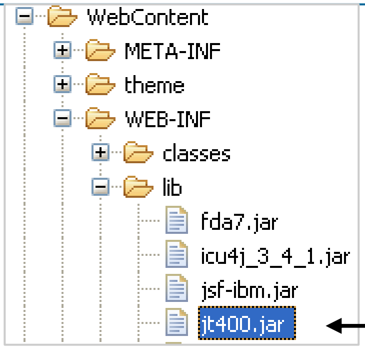
- When the Remote Command Server starts, it starts a number of "worker" jobs that process requests in the QUSRSYS sub-system. These jobs are all named QZRCSRVS.

- The next slide describes this process in graphical and technical (in the Notes section) detail

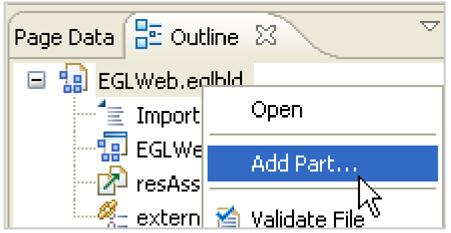
Calling RPG Programs – Steps (Revisited)

Here's what you'll do:

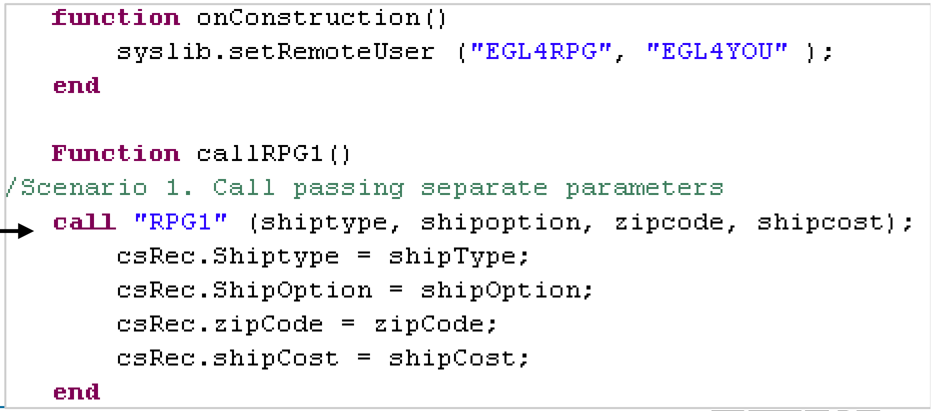
- One time step
  - Add the iSeries JT400 toolkit to your project and Java Build Path



- Once for each RPG program to call
  - Add an EGL Linkage Part for the program



- For each call
  - Code the business logic to call the RPG Program, from an EGL "client" logic part:
    - Service
    - JSFHandler
    - Program
    - Library



```
   function onConstruction()
       syslib.setRemoteUser ("EGL4RPG", "EGL4YOU" );
   end

   Function callRPG1()
/Scenario 1. Call passing separate parameters
   call "RPG1" (shiptype, shipoption, zipcode, shipcost);
       csRec.Shiptype = shipType;
       csRec.ShipOption = shipOption;
       csRec.zipCode = zipCode;
       csRec.shipCost = shipCost;
   end
```

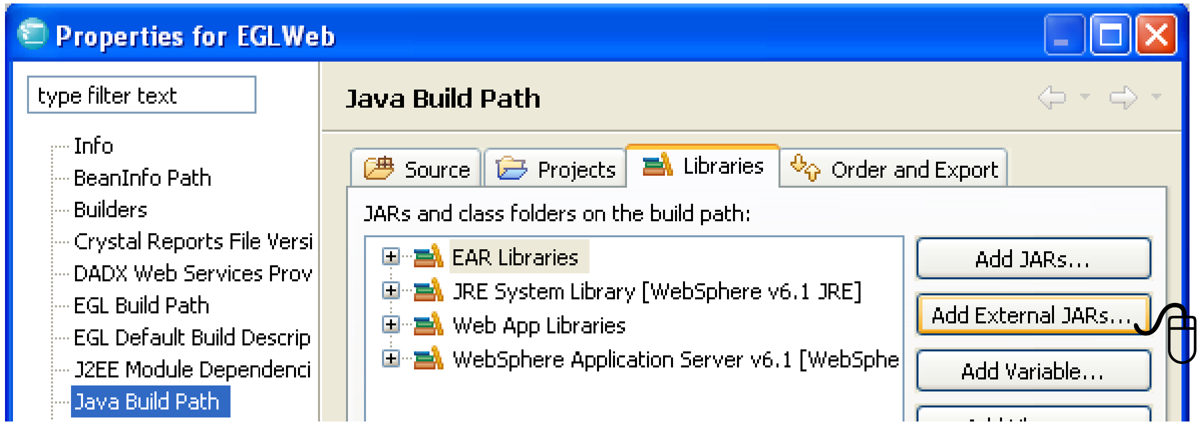
⏭ Add to the JT400 Toolkit to your Project

Calling RPG programs on an iSeries box requires the JT400 toolkit in your build path.

- There are 2 ways to do this (you only need to do one of these)

1. Add the file to the build path

- Right-click over your project and select **Properties**
- In the window that pops up, select **Java Build Path** on the left side of the screen
- Select **Add External Jars…** Find the **jt400.jar** file on your pc, select **Open**, Select **OK** to close the properties window.


Properties for EGLWeb

type filter text

Info
BeanInfo Path
Builders
Crystal Reports File Versi
DADX Web Services Prov
EGL Build Path
EGL Default Build Descrip
J2EE Module Dependenci
Java Build Path

Java Build Path

Source | Projects | Libraries | Order and Export

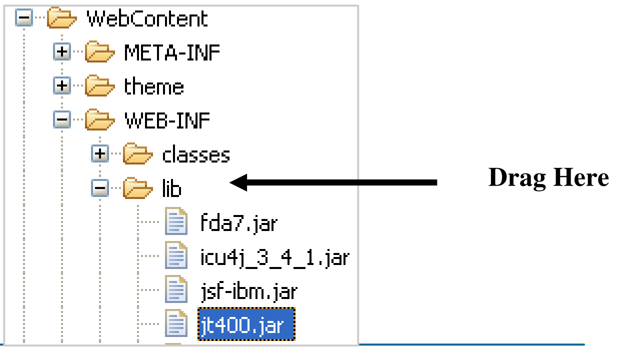
JARs and class folders on the build path:

EAR Libraries
JRE System Library [WebSphere v6.1 JRE]
Web App Libraries
WebSphere Application Server v6.1 [WebSphe

Add JARs…
Add External JARs…
Add Variable…

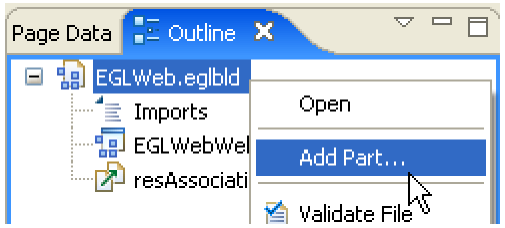
2. Put the file in the WEB-INF→Lib folder.

- Switch to the **Resource** perspective Resource
- Expand **Web Content → WEB-INF → lib** folder
  - **Drag jt400.jar into this folder**


WebContent
META-INF
theme
WEB-INF
classes
lib
fda7.jar
icu4j_3_4_1.jar
jsf-ibm.jar
jt400.jar

2. Create a Linkage Options Part – 1 of 3

- From Project Explorer,
  Open **EGLWeb.eglbld using the EGL Build Parts Editor**
- From the **Window** menu, open an **Outline** View
- From the Outline View (bottom left corner of the Workbench)
  - Right-click over EGLWeb.eglbld, select **Add Part…**
- Select ⊙**Linkage Options** and click: **Next**
- Name the part: **externalPrograms** click: **Finish**



From the externalPrograms part:

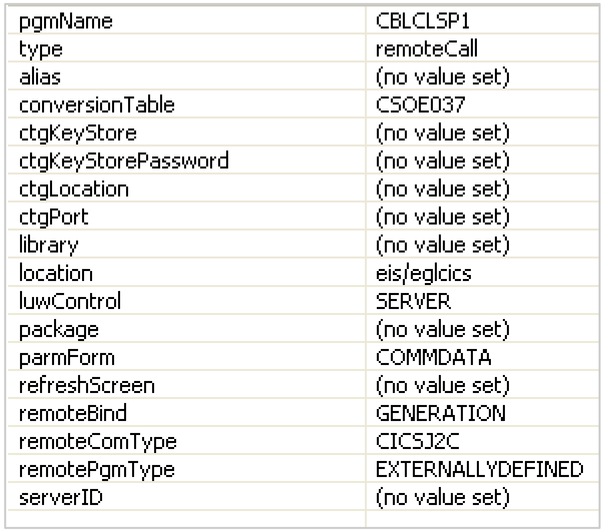
- Click: **Add**

In the CallLink Elements:

- Name the program: **CBLCLSP1**
- Type: **remoteCall**

Enter the following properties:

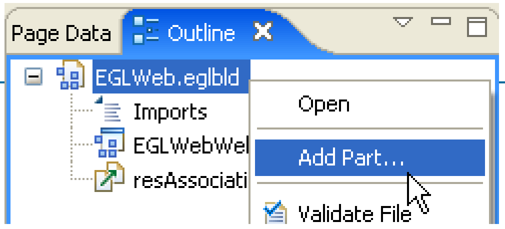
- conversionTable (type): **CSOE037**
- location (type): **eis/eglcics**
- luwControl (select): **SERVER**
- parmForm (select): **COMMDATA**
- remoteBind: **GENERATION**
- remoteComType (select): **CICSJ2C**
- remotePgmType (select): **EXTERNALLYDEFINED**



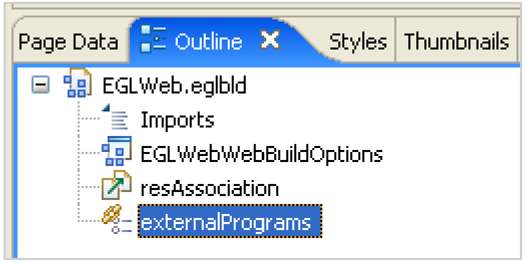
pgmName	CBLCLSP1
type	remoteCall
alias	(no value set)
conversionTable	CSOE037
ctgKeyStore	(no value set)
ctgKeyStorePassword	(no value set)
ctgLocation	(no value set)
ctgPort	(no value set)
library	(no value set)
location	eis/eglcics
luwControl	SERVER
package	(no value set)
parmForm	COMMDATA
refreshScreen	(no value set)
remoteBind	GENERATION
remoteComType	CICSJ2C
remotePgmType	EXTERNALLYDEFINED
serverID	(no value set)

- **Close** and **Save** your Linkage Options – and **Close/Save** the **Build File**

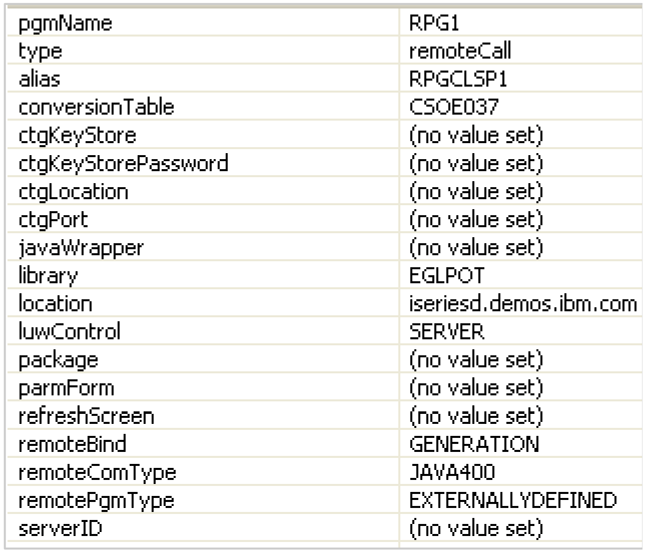
⏭ Add to the Build File's Linkage Option – 1 of 2



- From Project Explorer, **EGLWeb.eglbld using the EGL Build Parts Editor**
- From the **Window** menu, open an **Outline** View
- From the Outline View (bottom left corner of the Workbench)
  - Right-click over EGLWeb.eglbld, select **Add Part…**
- Select ⦿**Linkage Options** and click: **Next**
- Name the part: **externalPrograms** click: **Finish**



- From the externalPrograms part:
- Click: **Add**



pgmName	RPG1
type	remoteCall
alias	RPGCLSP1
conversionTable	CSOE037
ctgKeyStore	(no value set)
ctgKeyStorePassword	(no value set)
ctgLocation	(no value set)
ctgPort	(no value set)
javaWrapper	(no value set)
library	EGLPOT
location	iseriesd.demos.ibm.com
luwControl	SERVER
package	(no value set)
parmForm	(no value set)
refreshScreen	(no value set)
remoteBind	GENERATION
remoteComType	JAVA400
remotePgmType	EXTERNALLYDEFINED
serverID	(no value set)

In the CallLink Elements:
- Name the program: **RPG1**
- Type: **remoteCall**

Enter the following properties:
- Alias: **RPGCLSP1**
- conversionTable (type): **CSOE037**
- Library: **EGLPOT**
- location (type): **iseriesd.demos.ibm.com**
- luwControl (select): **SERVER**
- remoteBind: **GENERATION**
- remoteComType (select): **JAVA400**
- remotePgmType (select): **EXTERNALLYDEFINED**

- **Close** and **Save** your Linkage Options – and **Close/Save** the **Build File**

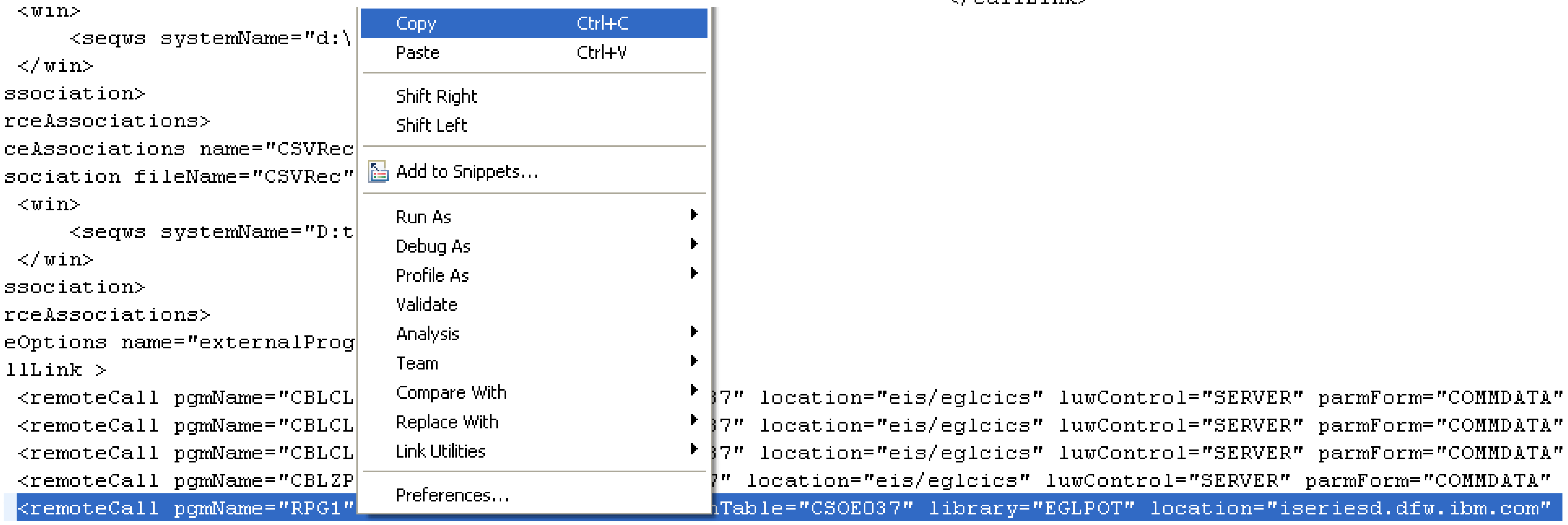
⏭ Add to the Build File's Linkage Option – 2 of 2

- From Project Explorer, Right-Click over **EGLWeb.eglbld** and select: **Open with Text editor**
- Scroll down to the line `<remoteCall pgmName="RPG1"…`
- **Click your mouse at the beginning of the line** (to set focus)
- **Press Shift/End** – to *select* the entire line (see reverse video line below)
- **Press Ctrl/C** – to copy the line
- **Enter a new line, and press Ctrl/V** 3 times – to copy/paste the entire line three times
- Modify the pgmName's and Alias' in the new lines to:

RPG2	**RPGCLSP2**
RPG3	**RPGCLSP3**
RPG4	**RPGZIPCK**

```
    <remoteCall pgmName="RPG1" alias="RPGCLSP1"
    <remoteCall pgmName="RPG2" alias="RPGCLSP2"
    <remoteCall pgmName="RPG3" alias="RPGCLSP3"
    <remoteCall pgmName="RPG4" alias="RPGZIPCK"
</callLink>
```

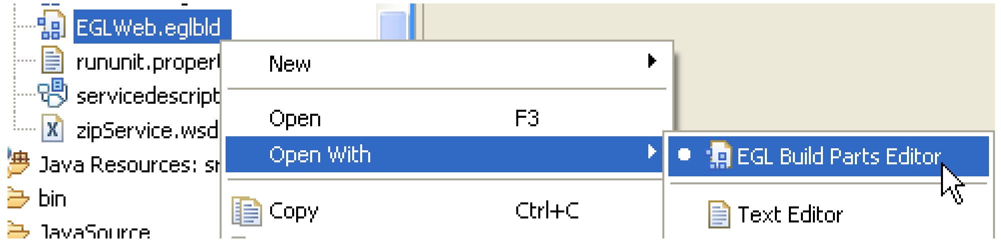
- **Close and save your edits**



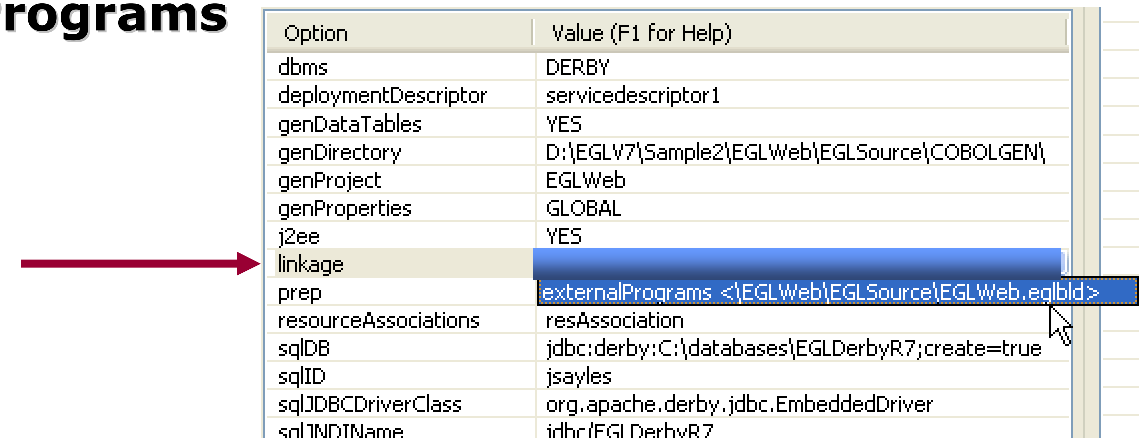
```
 <win>
     <seqws systemName="d:\
 </win>
ssociation>
rceAssociations>
ceAssociations name="CSVRec
sociation fileName="CSVRec"
 <win>
     <seqws systemName="D:t
 </win>
ssociation>
rceAssociations>
eOptions name="externalProg
llLink >
 <remoteCall pgmName="CBLCL                            37" location="eis/eglcics" luwControl="SERVER" parmForm="COMMDATA"
 <remoteCall pgmName="CBLCL                            37" location="eis/eglcics" luwControl="SERVER" parmForm="COMMDATA"
 <remoteCall pgmName="CBLCL                            37" location="eis/eglcics" luwControl="SERVER" parmForm="COMMDATA"
 <remoteCall pgmName="CBLZP                            7" location="eis/eglcics" luwControl="SERVER" parmForm="COMMDATA"
 <remoteCall pgmName="RPG1"                            nTable="CSOE037" library="EGLPOT" location="iseriesd.dfw.ibm.com"
```

2. Replicate the Linkage Options for Additional Programs – 3 of 3

- Close and save your edits to the Linkage Options



- Open EGLWeb.eglbld with the **EGL Build Parts Editor**
- Un-check: Show only specified options
- From the **linkage** Option, open the drop-down list and select: **externalPrograms**



Option	Value (F1 for Help)
dbms	DERBY
deploymentDescriptor	servicedescriptor1
genDataTables	YES
genDirectory	D:\EGLV7\Sample2\EGLWeb\EGLSource\COBOLGEN\
genProject	EGLWeb
genProperties	GLOBAL
j2ee	YES
linkage	
prep	externalPrograms <\EGLWeb\EGLSource\EGLWeb.eglbld>
resourceAssociations	resAssociation
sqlDB	jdbc:derby:C:\databases\EGLDerbyR7;create=true
sqlID	jsayles
sqlJDBCDriverClass	org.apache.derby.jdbc.EmbeddedDriver
sqlJNDIName	jdbc/EGLDerbyR7

- **Close** and **save** your Build file
- Right-click over your EGLWeb project and **Generate**

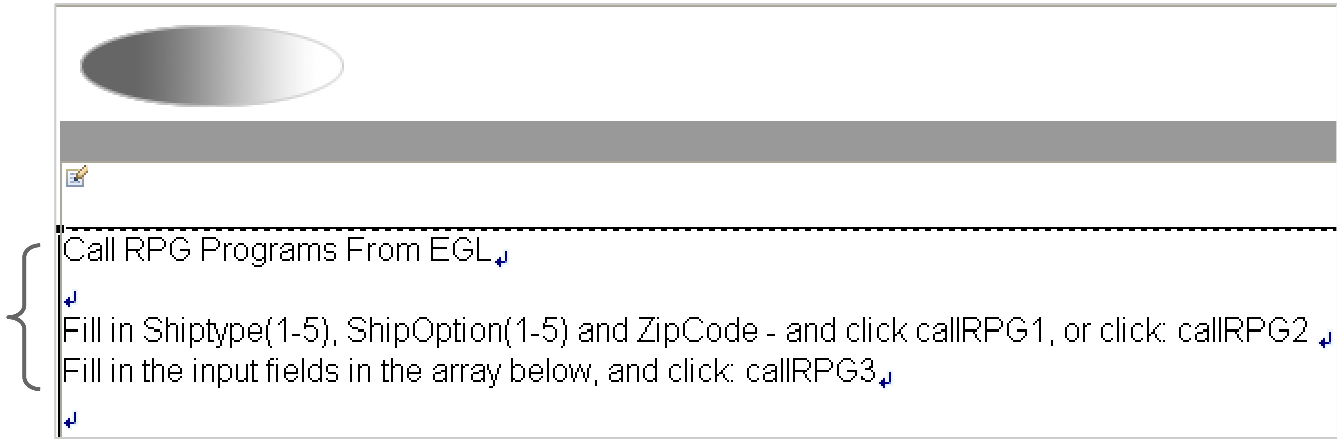
3. Create a New Page to Test Your RPG Program Calls

Before creating the page you need to generate the new Build-file entries to Java

- Right-click over the \EGLSource\ folder and select **Generate**

Now let’s create a page that calls your COBOL programs

- Right-click over \WebContent\ and select, New > Web Page
- Name the page: **callRPG.jsp**
- Change the boiler-plate page heading text as shown here…



Call RPG Programs From EGL

Fill in Shiptype(1-5), ShipOption(1-5) and ZipCode - and click callRPG1, or click: callRPG2
Fill in the input fields in the array below, and click: callRPG3

- Right-click and edit the Page Code

From inside the JSFHandler

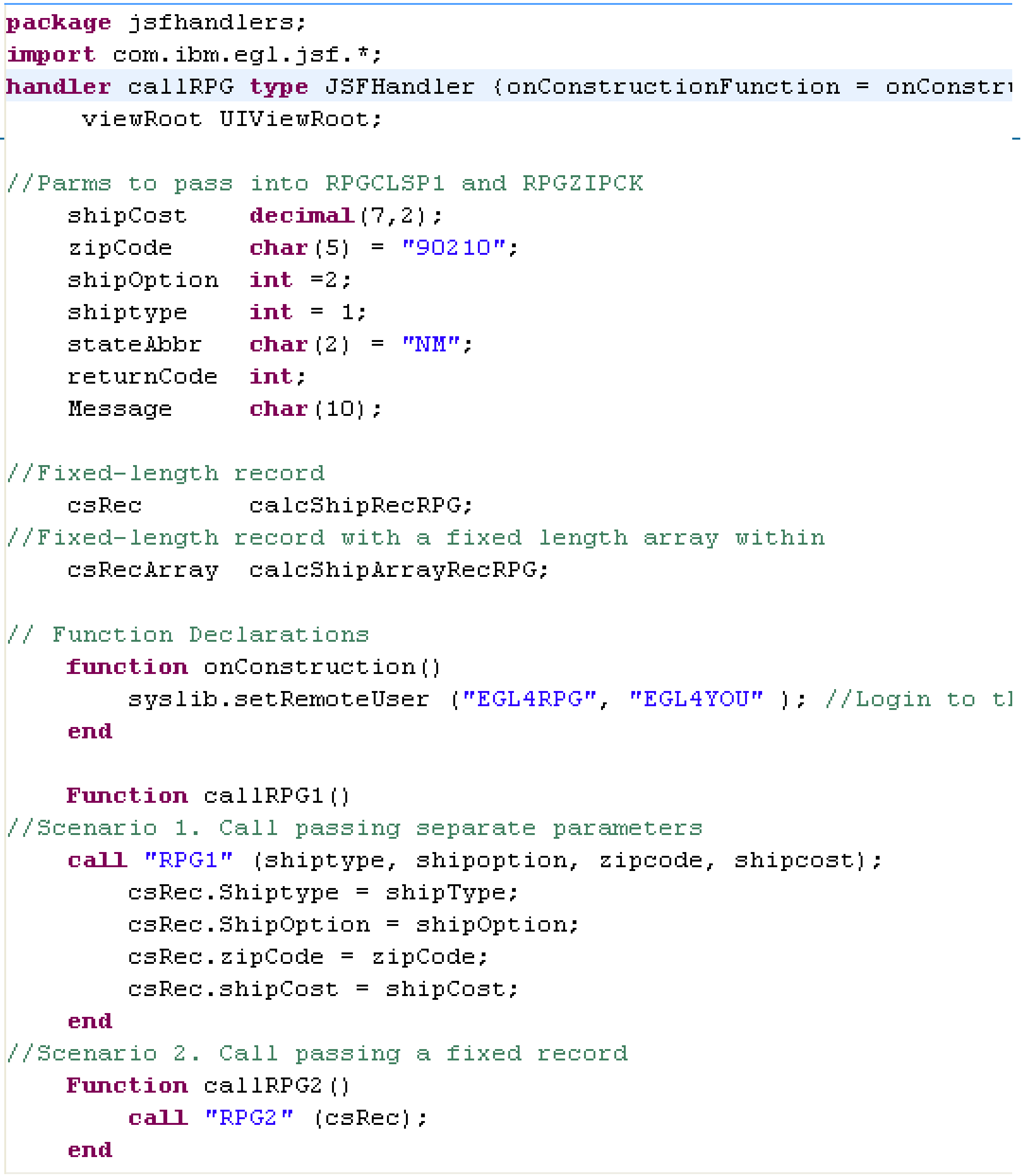
- Select and replace all of the boiler-plate code with the code in the Notes section of this slide
  - Note that there is a lot of source code, as the four RPG programs are included as comments at the bottom
  - Details of the program calls are covered on the next slide

3. EGL RPG Program Calls

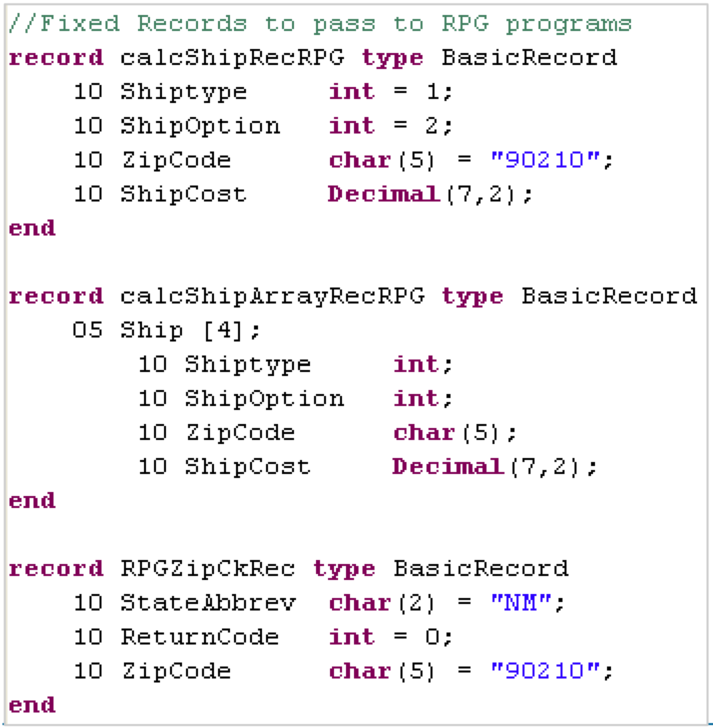
So – you can see from this example, that there is nothing complex whatsoever about calling RPG programs from EGL. This JSFHandler has several different scenarios:

- **Passing individual parameters to RPG**
- **Passing a single record**
- **Passing a record that contains an array**

- Here are the record definitions (nothing new…hope you're not disappointed ☺)



```
package jsfhandlers;
import com.ibm.egl.jsf.*;
handler callRPG type JSFHandler {onConstructionFunction = onConstru
     viewRoot UIViewRoot;

//Parms to pass into RPGCLSP1 and RPGZIPCK
    shipCost     decimal(7,2);
    zipCode      char(5)  = "90210";
    shipOption   int =2;
    shiptype     int = 1;
    stateAbbr    char(2)  = "NM";
    returnCode   int;
    Message      char(10);

//Fixed-length record
    csRec        calcShipRecRPG;
//Fixed-length record with a fixed length array within
    csRecArray   calcShipArrayRecRPG;

//  Function Declarations
    function onConstruction()
        syslib.setRemoteUser ("EGL4RPG", "EGL4YOU" ); //Login to tl
    end

    Function callRPG1()
//Scenario 1. Call passing separate parameters
    call "RPG1" (shiptype, shipoption, zipcode, shipcost);
        csRec.Shiptype = shipType;
        csRec.ShipOption = shipOption;
        csRec.zipCode = zipCode;
        csRec.shipCost = shipCost;
    end
//Scenario 2. Call passing a fixed record
    Function callRPG2()
        call "RPG2" (csRec);
    end
```



```
//Fixed Records to pass to RPG programs
record calcShipRecRPG type BasicRecord
    10 Shiptype      int = 1;
    10 ShipOption    int = 2;
    10 ZipCode       char(5)  = "90210";
    10 ShipCost      Decimal(7,2);
end

record calcShipArrayRecRPG type BasicRecord
    05 Ship [4];
        10 Shiptype      int;
        10 ShipOption    int;
        10 ZipCode       char(5);
        10 ShipCost      Decimal(7,2);
end

record RPGZipCkRec type BasicRecord
    10 StateAbbrev   char(2)  = "NM";
    10 ReturnCode    int = 0;
    10 ZipCode       char(5)  = "90210";
end
```

Press Ctrl/S (save) your JSFHandler

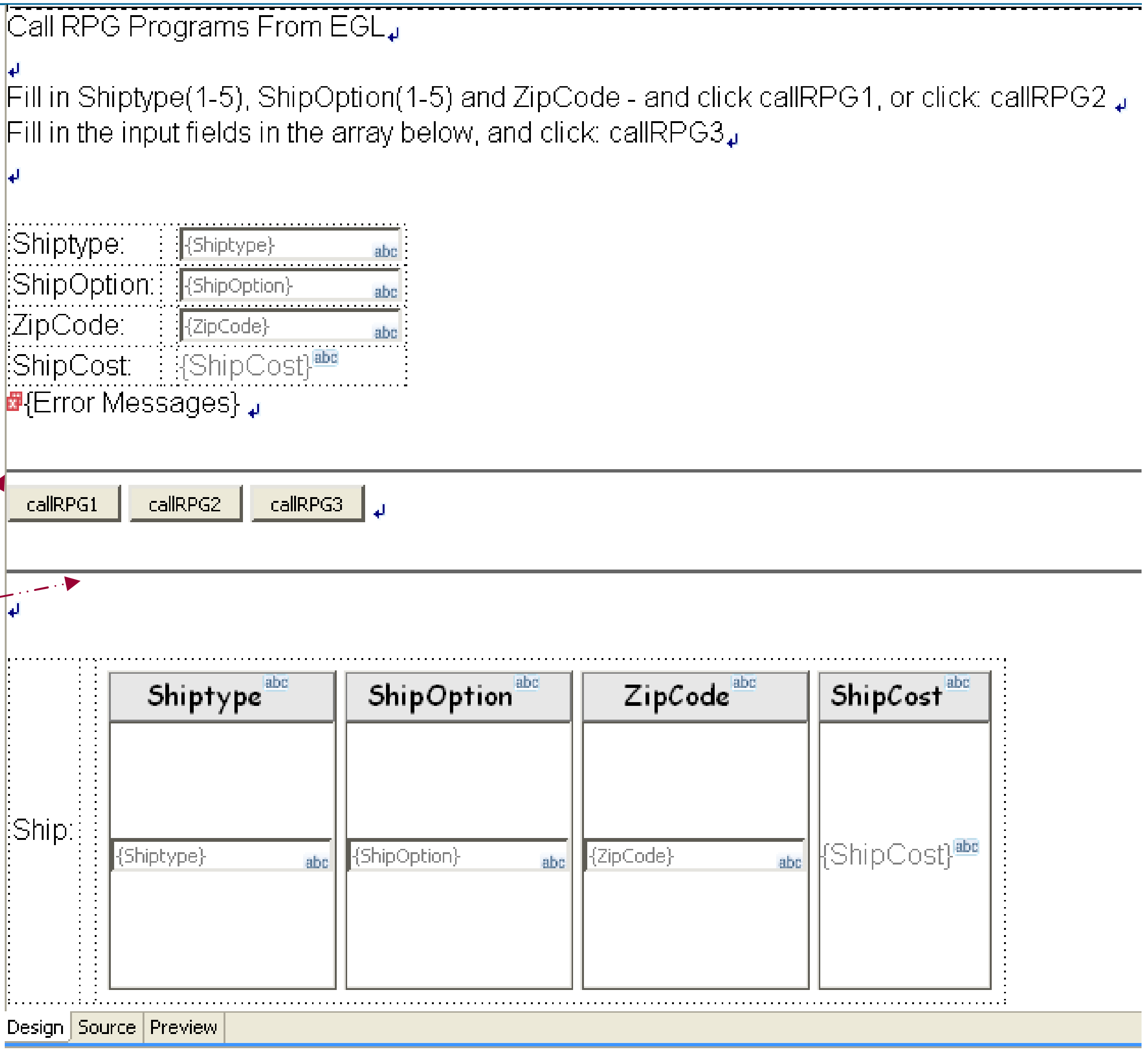
4. Create the Page Results

From Page Designer:

- From Page Data – drag **csRec** on to the page ➔
  - Make the top three fields input
  - Make **ShipCost** output
  - From options, specify no Submit buttons (un-check)
- From Page Data – select all three (`callRPG1, callRPG2` and `callRPG3`) functions, and drag them onto the page, to create three submit buttons.

You can optionally add HTML Horizontal Rule tags between the controls

- From Page Data – drag **csRecArray** onto the page ➔
  - Make the first three columns **Input Text** type
  - Make ShipCost **outputText**


Call RPG Programs From EGL

Fill in Shiptype(1-5), ShipOption(1-5) and ZipCode - and click callRPG1, or click: callRPG2
Fill in the input fields in the array below, and click: callRPG3

Shiptype:	{Shiptype}
ShipOption:	{ShipOption}
ZipCode:	{ZipCode}
ShipCost:	{ShipCost}

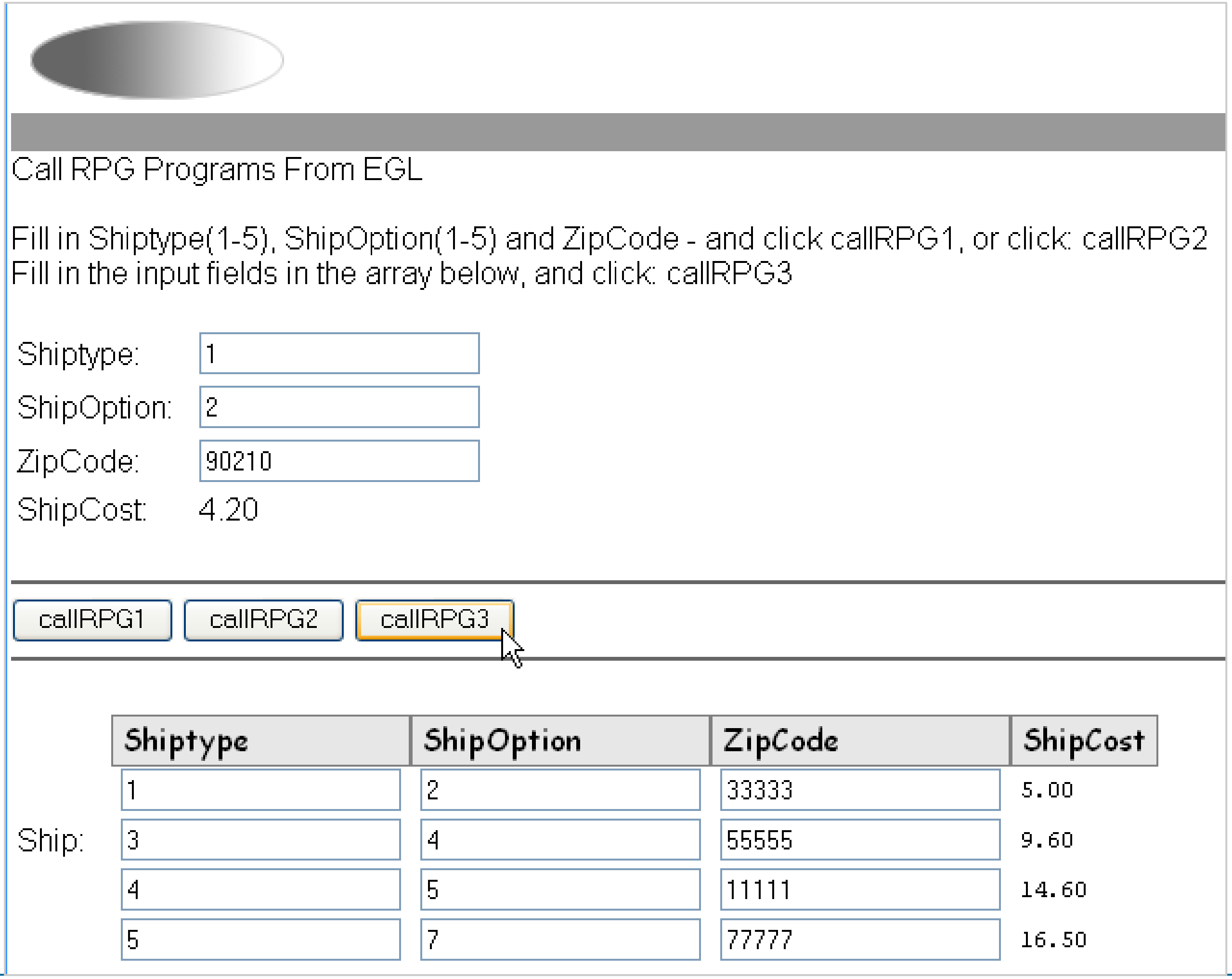
{Error Messages}

Ship:

Shiptype	ShipOption	ZipCode	ShipCost
{Shiptype}	{ShipOption}	{ZipCode}	{ShipCost}

⏭ Run the Page – and Call RPG

- Run the page on the server.
- Enter the values shown below – and click the buttons.
  - Note that callRPG2 passes the same data (just using a fixed record instead of individual parms)
  - Note also that you must fill in the input values in the array and click callRPG3 to return the 4 ShipCosts



Call RPG Programs From EGL

Fill in Shiptype(1-5), ShipOption(1-5) and ZipCode - and click callRPG1, or click: callRPG2
Fill in the input fields in the array below, and click: callRPG3

Shiptype:	1
ShipOption:	2
ZipCode:	90210
ShipCost:	4.20

	Shiptype	ShipOption	ZipCode	ShipCost
	1	2	33333	5.00
Ship:	3	4	55555	9.60
	4	5	11111	14.60
	5	7	77777	16.50

***Notes

Unit

EGL and Services (SOA)

Topics:

- **EGL and Services Overview**
- Creating and Consuming Local Services
- Setting up for Web Services
- Creating and consuming an EGL Web Service
- Consuming a 3rd Party Web Service

The Promise of Services and SOA (Service Oriented Architecture)

Services are the driving business system design paradigm of the day.

Services – implemented in EGL provide a cross platform language for business oriented development

Services and SOA are based on the concept of "Service Oriented Design"

At development time…

- Focus on the business logic
- Implement SOA design elements: services and interfaces
- Leverage existing business developers for new SOA development
- Ignore deployment targets/technology while coding/testing

Leverage external web services…

- EGL Interfaces
  - Represent external web services
  - Are created via import from WSDL
  - Allow the EGL developer to stay within the context of the EGL programming model

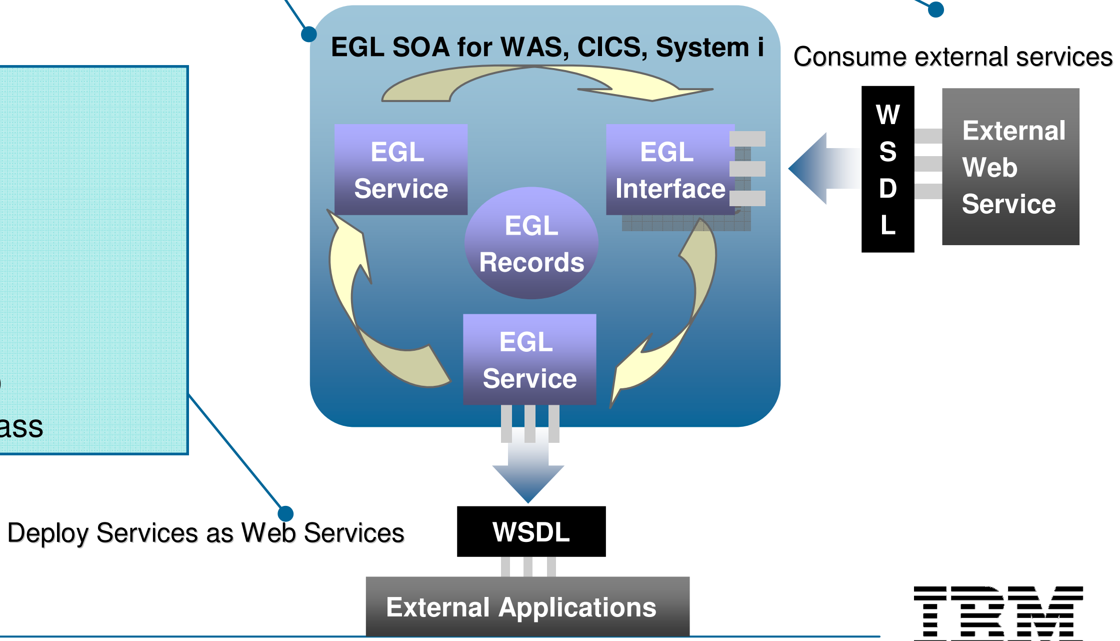
Deploy EGL services…

To any platform

- Java to WAS/Tomcat/etc.
- COBOL to CICS, iSeries

As…

- A Web service (uses SOAP)
- A private service (uses CICS ECI, J2C, or TCP)
- Other SOA runtimes when they reach critical mass



Service Oriented Design – Concepts

Service Orientated Design is a software design paradigm that encourages the creation of functionally discrete (modular) business logic, composed of standardized software components (logic routines and subroutines with standard interfaces)

- Services are software components that:
  - Are logically separated, along the lines of a "business service" – such as:
    - Rent a car – which is further de-composed into:
      - Reserve a vehicle – consisting of lower-level services such as:
        - Confirm a location
        - Confirm the customer information
        - Get payment information
        - Confirm customer and payment information
        - Create the reservation

      …

- Services are formally declared with a standard network interface that allows them to be invoked by other services that only know the interface.
- Also, services:
  - May be created at various levels of business/technical granularity (i.e. low-level vs. high-level)
  - Can be assembled/combined ("choreographed") to create higher-level business functionality
  - Can be deployed to any platform – independently – and called from any platform through their network interface

Network Interface (and More Services Concepts)

- *“A Service is a* ***network interface*** *to programs that live on a particular machine implemented in such a manner that any other program (or many programs) running on any other machine, written in any language, and running on any OS can call it.”*
  Bob Cancilla, IBM.

- Additionally:
  - Services are an abstract design pattern - they are not the same as Web Services. In fact, a service can be implemented using a number of different technologies, including:
    - **RPC** (Remote Program Call) - which you implement with an EGL:
      - Local Service …or…
      - Library call …or…
      - Program
    - **DCOM** (Distributed Component Object Model)
    - **REST** (Representational State Transfer)
    - **CORBA** (Common Object Request Broker)
    - **Web Services** – which you implement with an EGL Service part

- The “network interface” for a Web Service is formally termed a **WSDL**
  - **W**eb **S**ervices **D**escription **L**anguage

Creating and Consuming Services – WSDL Files and EGL Interfaces

- **WSDL** is an XML document used to describe and locate Web Services.
- A WSDL:
  - Specifies the run-time location as a URL
  - Specifies operations and data in the Web Service
  - Is written in XML using the SOAP transport protocol

- But you don't want to:
  - Learn XML or SOAP or…
  - Create the WSDL files manually ☹

- You want to code EGL business logic – exactly as you've done all along in this course ☺

- So in fact, the RBD tooling will generate all of the "plumbing" for you – when you create or consume Web Services:
  - When you create an EGL service, the tooling generates WSDL files, so you will not need to learn XML (and other low-level languages, terms and concepts)
  - When you consume a service, RBD generates EGL Interfaces from existing WSDL files for you – automatically

Topic

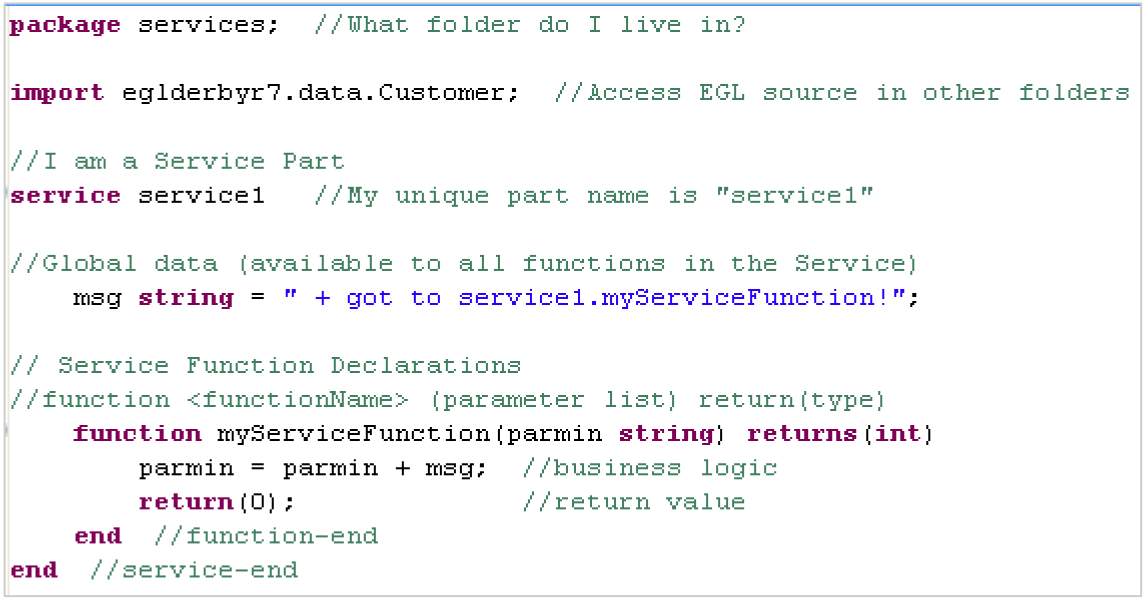
EGL and Services Overview

Sub-Topics:

- EGL and Services Overview
- **Creating and Consuming Local Services**
- Setting up for Web Services
- Creating and consuming an EGL Web Service
- Consuming a 3rd Party Web Service

A Simple EGL Service

- A service (Local or Web) is coded from the same basic EGL programming model you've used all along:
  - **package** statement
  - **import** statement(s)
  - **service** <UniqueName> {optional properties}
  - **Global data** (area)
  - **Functions** – including a **returns**(type) on the signature



```
package services;  //What folder do I live in?

import eglderbyr7.data.Customer;  //Access EGL source in other folders

//I am a Service Part
service service1   //My unique part name is "service1"

//Global data (available to all functions in the Service)
    msg string = " + got to service1.myServiceFunction!";

// Service Function Declarations
//function <functionName> (parameter list) return(type)
    function myServiceFunction(parmin string) returns(int)
        parmin = parmin + msg;   //business logic
        return(0);               //return value
    end  //function-end
end  //service-end
```

EGL Services – Steps for Creating and Consuming Local Services

Here are the high-level steps to create a Local Service.

- To **create** a local Service using EGL
  1. Code the service
  2. Generate the service
  3. (if not available) – Create a new services descriptor file
  4. Add the service as a local service
  5. (if not done) Point to the services descriptor file, from the build-file
  6. Generate the descriptor file

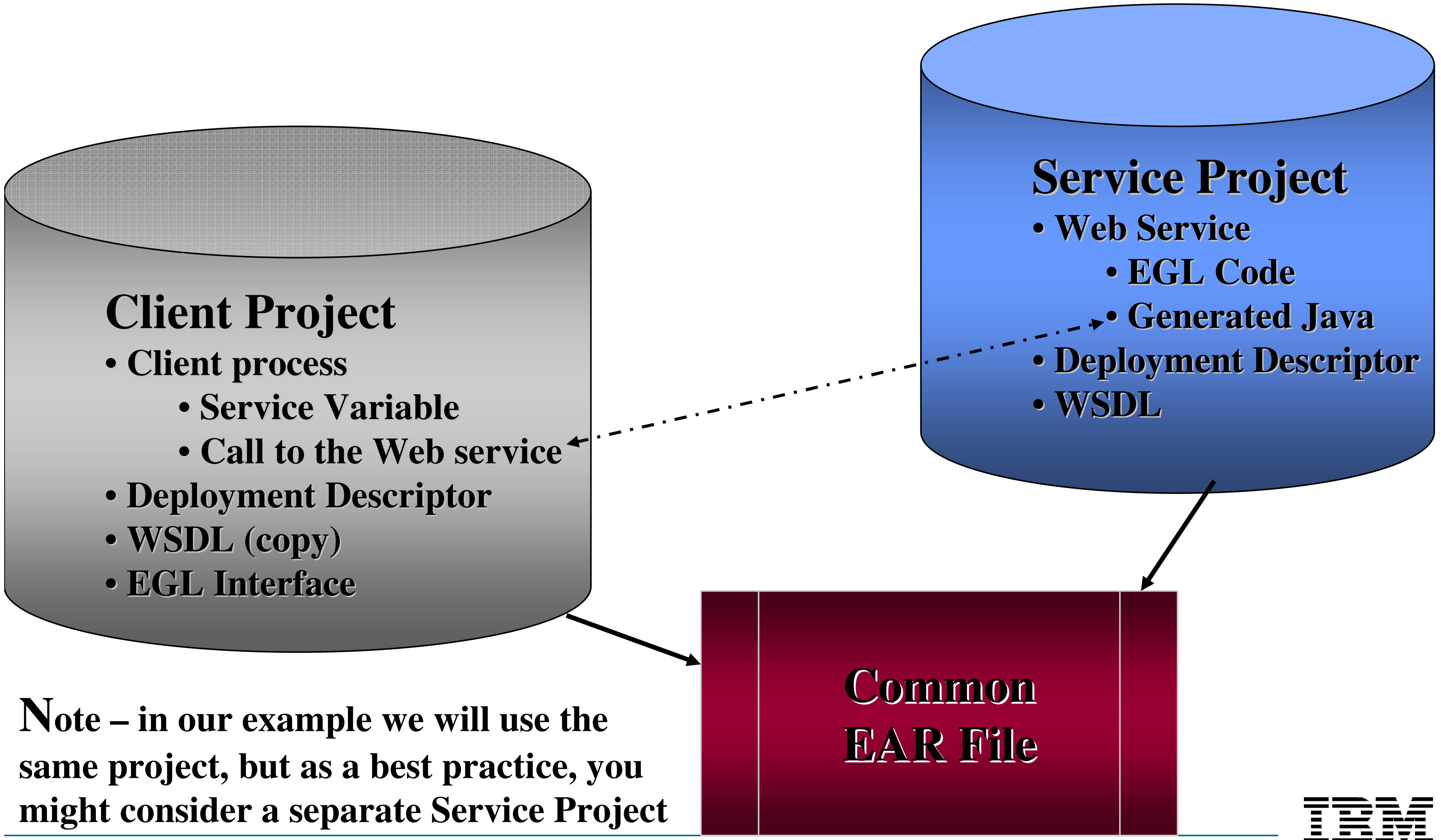
- To **consume** a local service from an EGL client
  1. Code a variable of <serviceName> type in the client logic
     - Add a Bindservice property with the appropriate bindkey (not shown, and optional if service binding name is the same as the service part name)

     ```
     serviceVar  service1  {@Bindservice{}};
     ```
  2. Code the call to the ServiceVar.function(…) – passing parameters

     ```
     retCode int = serviceVar.nyTest(zipIn);
     ```

EGL Services – Conceptual View

- To do a Web Service as opposed to a Local Service, you may want to architect a separate project for your Web Services. You can think of this as a “service” project – and you can think of calls to the web services as coming from “client” projects



Note – in our example we will use the same project, but as a best practice, you might consider a separate Service Project

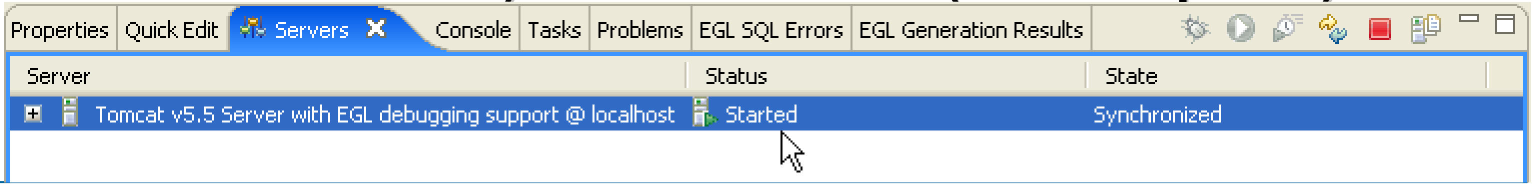
EGL Services – Steps for Creating and Consuming Web Services

Here are the steps we'll take together, to create a new EGL Web project (from scratch) and use it as our Web Service project.

1. Start Tomcat – or **WebSphere** (see *****Notes**)
2. Customize your project's Build File
3. Create and generate a Web Service
   - Code the service
   - Generate the service
4. Generate the WSDL for your Service – may need to customize the service port (end point)
5. Test the WSDL Using Web Services Explorer

6. Consume the Web Service
   - (If using a 3rd Party WSDL) Import or copy the WSDL to your Client/Project
   - Add a WSDL entry in the services descriptor file for all Web Services
   - Generate the services descriptor file
   - In the client process - code a variable of <serviceName> type
     - Add a Bindservice property with the appropriate bindkey
   - Code the call to the ServiceVar.function – passing parameters

1. (From the Servers Tab) Start Tomcat (or WebSphere)

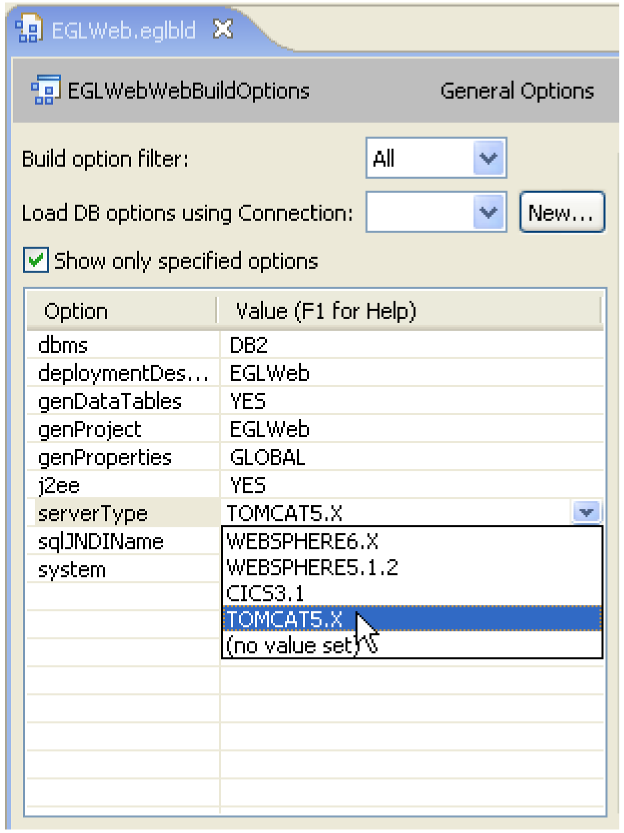


Server	Status	State
Tomcat v5.5 Server with EGL debugging support @ localhost	Started	Synchronized

⏭ 2. Customize Your Project's Build-File

- Next – you need to select the serverType you will be publishing the Web Server to
  - **Open EGLWeb.eglbld**
  - **Un-check Show only specified options**
  - **Scroll down to find the serverType option**
  - **Use the combo-box to select the serverType for your project**
    - **WEBSPHERE**
    - **TOMCAT**

➔ Select the serverType



Build option filter: All

Load DB options using Connection:

[✓] Show only specified options

Option	Value (F1 for Help)
dbms	DB2
deploymentDes...	EGLWeb
genDataTables	YES
genProject	EGLWeb
genProperties	GLOBAL
j2ee	YES
serverType	TOMCAT5.X
sqlJNDIName	
system	

WEBSPHERE6.X
WEBSPHERE5.1.2
CICS3.1
[✓ TOMCAT5.X]
(no value set)

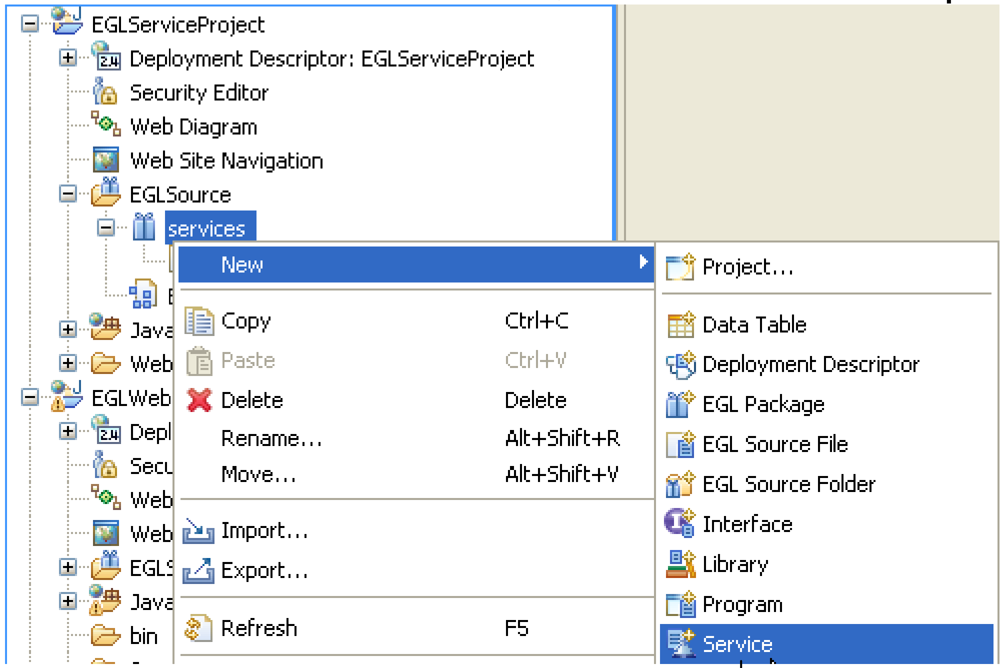
- **From Project Explorer:**
  - **Generate your Project**

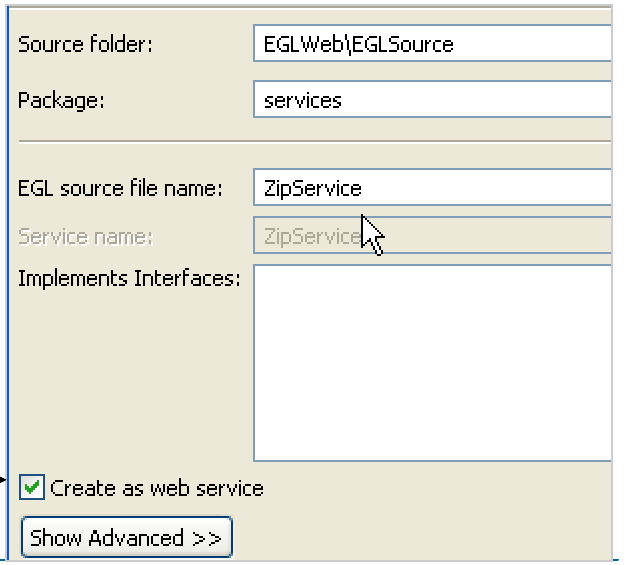
⏭ 3. Create a new EGL Web Service

- *(If not already done)* Create a new EGL **package** in your EGLServiceProject under \EGLSource\ named: **services**

- Create a new Service, in the **\services**\ package named: **ZipService**

See ***Notes





Source folder: EGLWeb\EGLSource

Package: services

EGL source file name: ZipService

Service name: ZipService

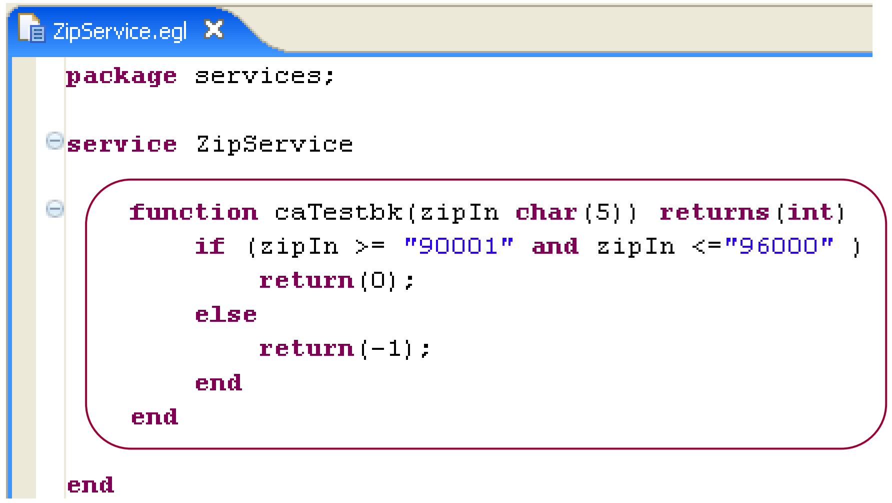
Implements Interfaces:

☑ Create as web service

☑ Check: **Create as web service**

⏭ 3. Create a new EGL Web Service – continued

- Create a new function in **ZipService** named: **caTestbk**



ZipService.egl

```
package services;

service ZipService

    function caTestbk(zipIn char(5)) returns(int)
        if (zipIn >= "90001" and zipIn <="96000" )
            return(0);
        else
            return(-1);
        end
    end

end
```

You can either try typing the above function in (using Content Assist), or you can copy/paste in the code – found in the Notes section.

Save the Web service

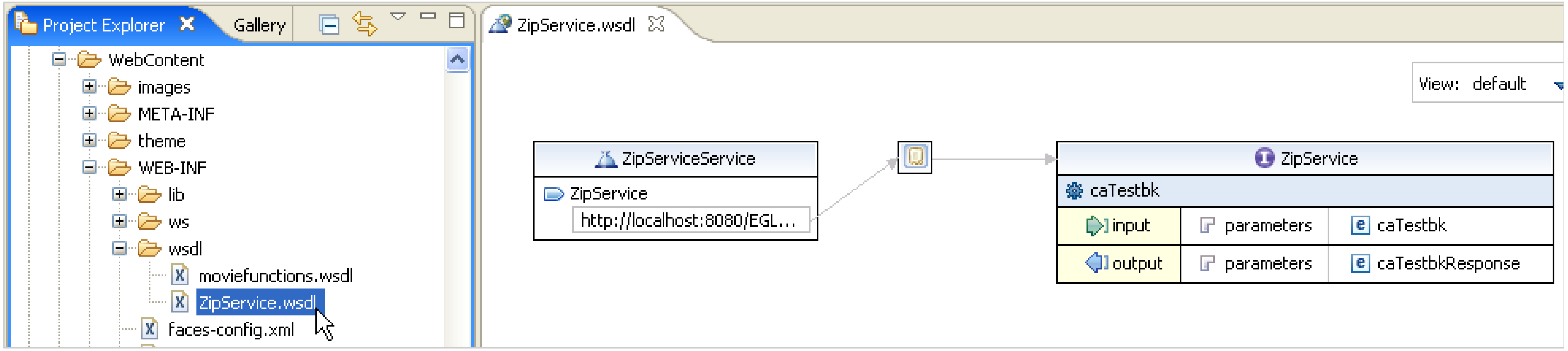
⏭ Press **Ctrl/S** to save and validate the Service

⏭ 4. Generate the WSDL for Your Web Service

⏭ **Generate the Web service -** From Project Explorer

- Select **EGLWeb**
- Right-Click and select **Generate**

After successfully generating the WSDL, you can open it in the Content Area – it's located under: **`\WebContent\WEB-INF\wsdl\`**



- **Close ZipService.WSDL – unless you are using WebSphere** ***Notes

Next we will test the Web Service, then we will work in the Client Project and call this Web Service from both a batch program and your updatecustomer2 web page. But if you are using WebSphere – before continuing please read the Slide Notes – as you may need to customize the WSDL "end-points" – the port# for the WSDL address.

***Notes

⏭ 5. (Optional) Test the Web Service – Interactively

👓 RBD offers a nifty interactive Web Services test facility called the Web Services Explorer.

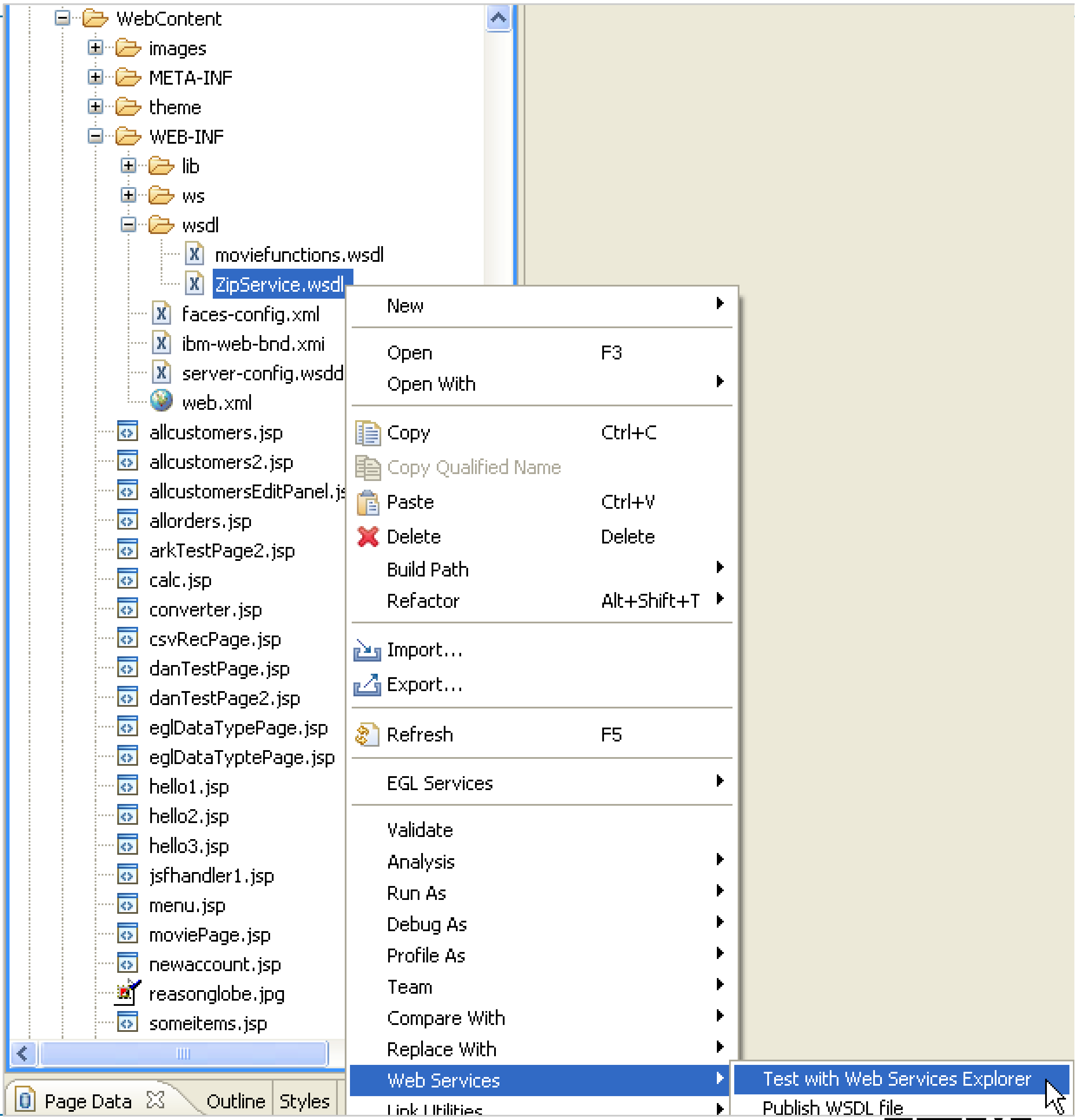
You can use this tool to test your Web Service functionality effectively, before embedding calls to it from your service client.

From Project Explorer – find: **ZipService.wsdl** you just generated under **\WebContent\WEB-INF\wsdl\**

⏭ Right-click over **ZipService.wsdl**

- Select **Web Services**
  - **Test with Web Services Explorer**

*****Notes – please review if you do NOT see the Web Services option shown here**



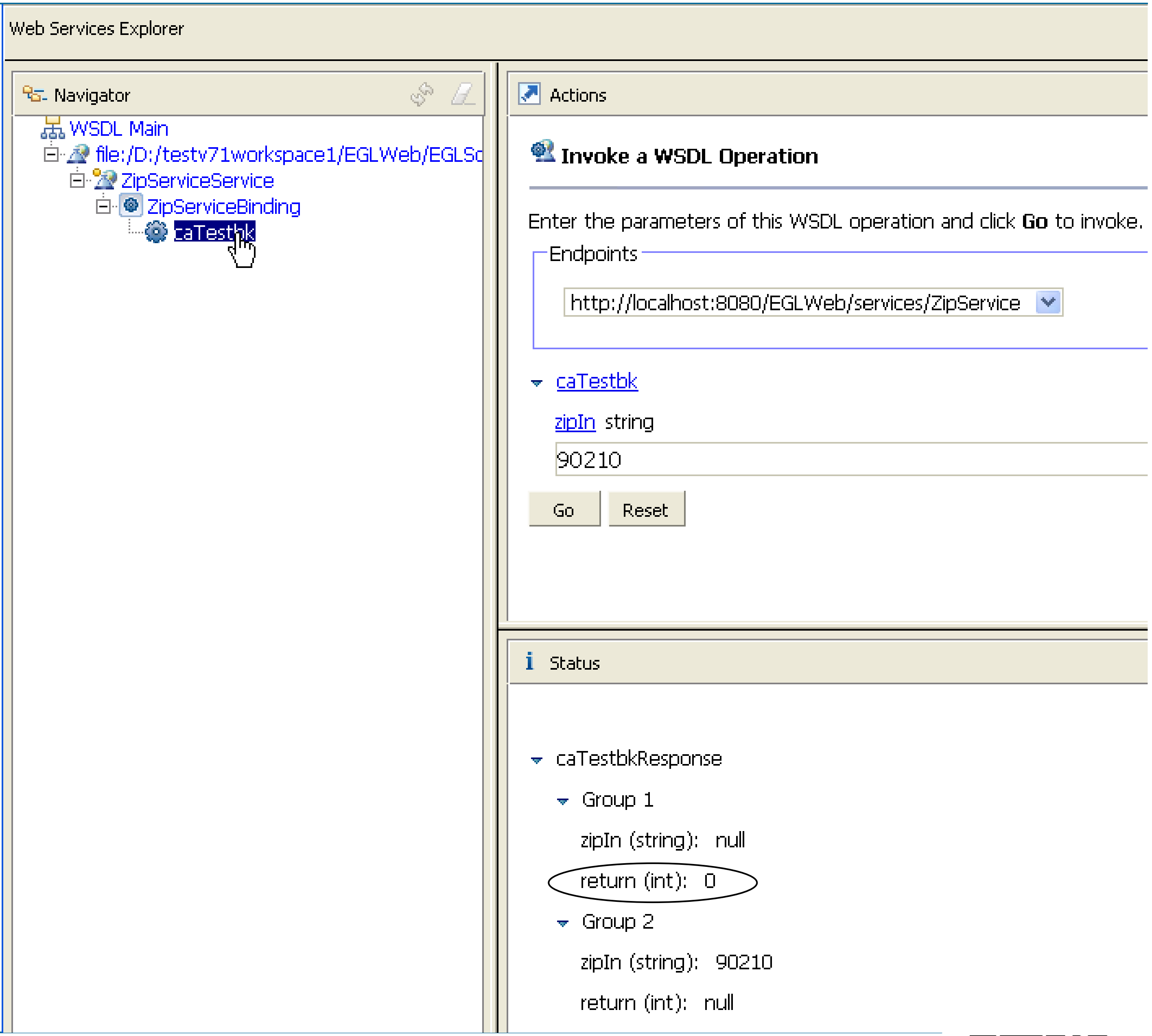
5. Test the Web Service – Interactively

This opens the Web Services Explorer – and specifically, it opens your ZipService in the Navigator – and all of the services functions are exposed as Actions for you to test.

- **Click caTestbk**
- Enter a zipCode:
- Click **Go**
  - Note that **90210** returns: 0
  - Less than 90000 returns: -1

*** Your Application Server must be still be started ***

Close the Explorer


Web Services Explorer

Navigator

WSDL Main
file:/D:/testv71workspace1/EGLWeb/EGLSc
ZipServiceService
ZipServiceBinding
caTestbk

Actions

Invoke a WSDL Operation

Enter the parameters of this WSDL operation and click **Go** to invoke.

Endpoints

http://localhost:8080/EGLWeb/services/ZipService

caTestbk

zipIn string

90210

Go Reset

Status

caTestbkResponse
Group 1
zipIn (string): null
return (int): 0
Group 2
zipIn (string): 90210
return (int): null

Steps for Creating and Consuming Web Services – Client Project

Review of the steps. By following the previous instructions you have created the Service Project, Service and deployed it as a Web Service.

1. Start Tomcat (see ***Notes)
2. Customize your project's Build File
3. Create and generate a Web Service
   - Code the service
   - Generate the service
4. Generate the WSDL for your Service
5. Test the WSDL Using Web Services Explorer

6. **Consume the Web Service**

   (If using a 3rd Party WSDL…next workshop)
   - Import or copy the WSDL to your Client/Project
   - Add a WSDL entry in the services descriptor file for all Web Services

   (If you've created an EGL Web Service…this workshop)
   - Create an EGL Client Interface to the WSDL

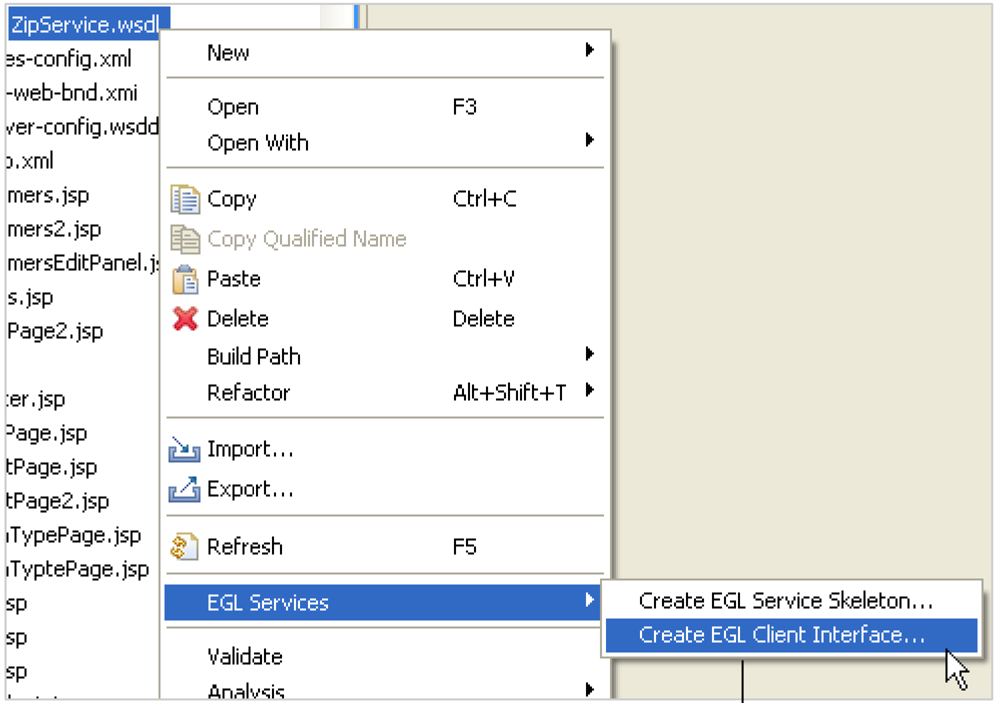
   - Generate the services descriptor file
   - In the client process - code a variable of <serviceName> type
     - Add a Bindservice property with the appropriate bindkey
   - Code the call to the ServiceVar.function – passing parameters

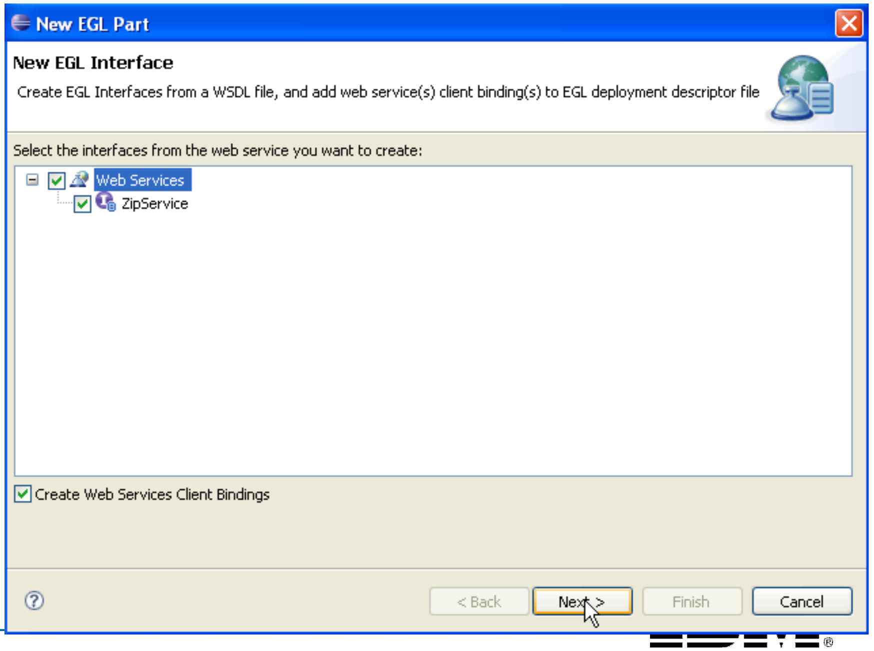
Note – we are not using a 3rd Party **WSDL** in this workshop, so we begin with the step to create an EGL Client Interface to the WSDL

⏭ 6. Create an EGL Client Interface to the WSDL

(from Project Explorer)

- **Right-click** over **ZipService.wsdl**
- Select: **EGL Services > Create EGL Client Interface…**





New EGL Interface

Create EGL Interfaces from a WSDL file, and add web service(s) client binding(s) to EGL deployment descriptor file

Select the interfaces from the web service you want to create:

☑ Create Web Services Client Bindings

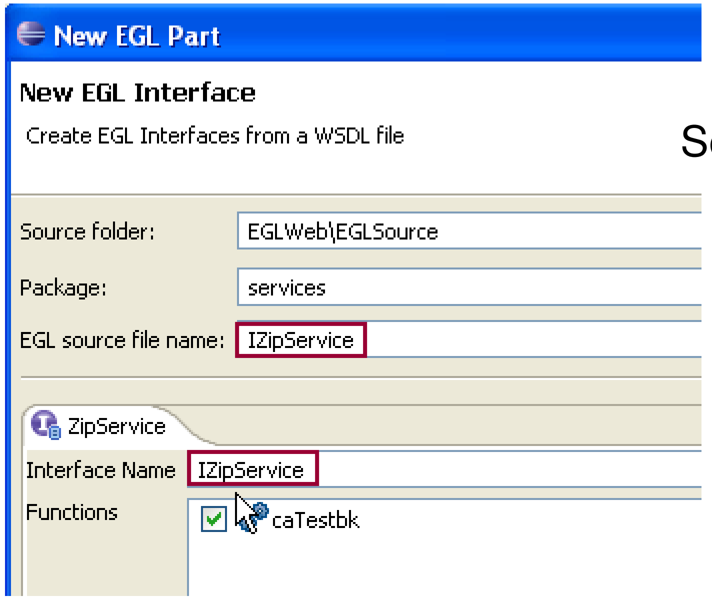
- **Click Next >**

⏭ 6. Create an EGL Client Interface to the WSDL

From New EGL Interface

- Rename the EGL source file name: **IZipService**
- Rename the Interface name: **IZipService**

See ***Notes



New EGL Part

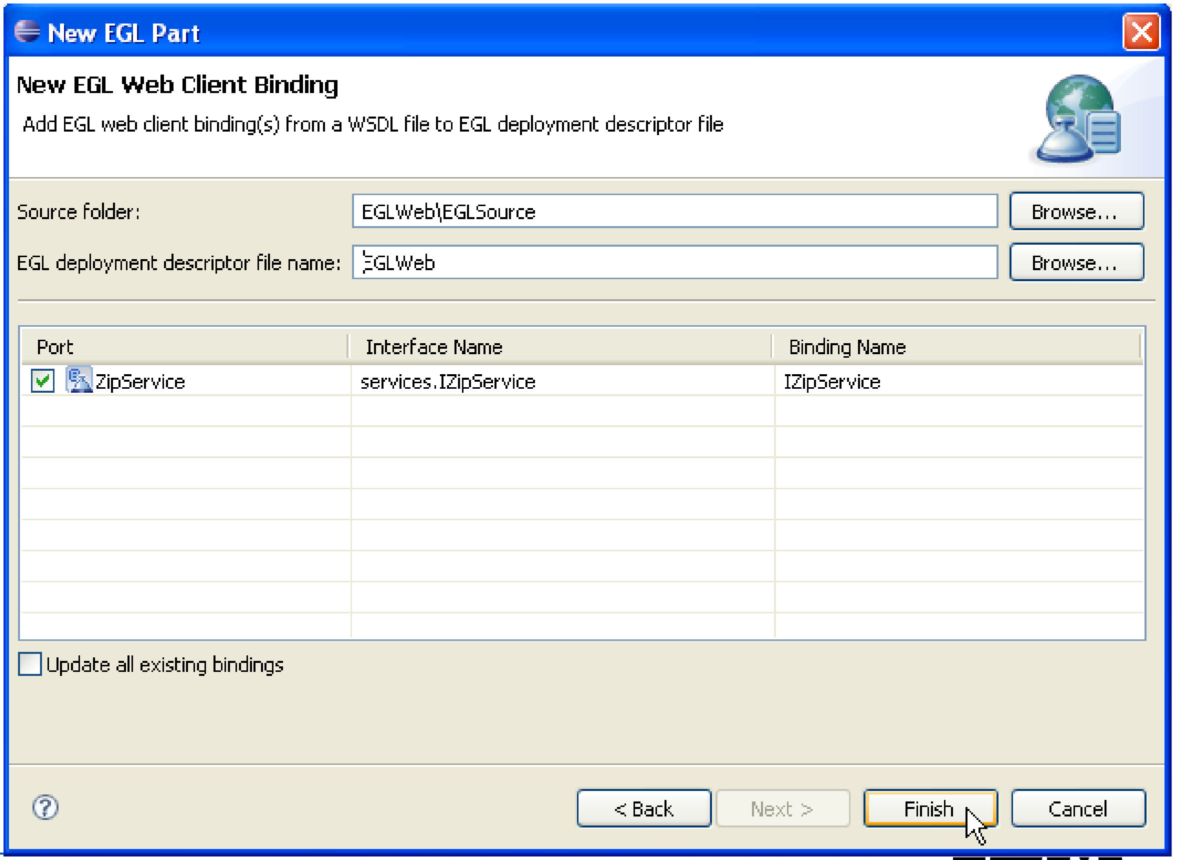
New EGL Interface

Create EGL Interfaces from a WSDL file

Source folder:	EGLWeb\EGLSource
Package:	services
EGL source file name:	IZipService

ZipService

Interface Name	IZipService
Functions	✓ caTestbk



New EGL Part

New EGL Web Client Binding

Add EGL web client binding(s) from a WSDL file to EGL deployment descriptor file

Source folder:	EGLWeb\EGLSource
EGL deployment descriptor file name:	EGLWeb

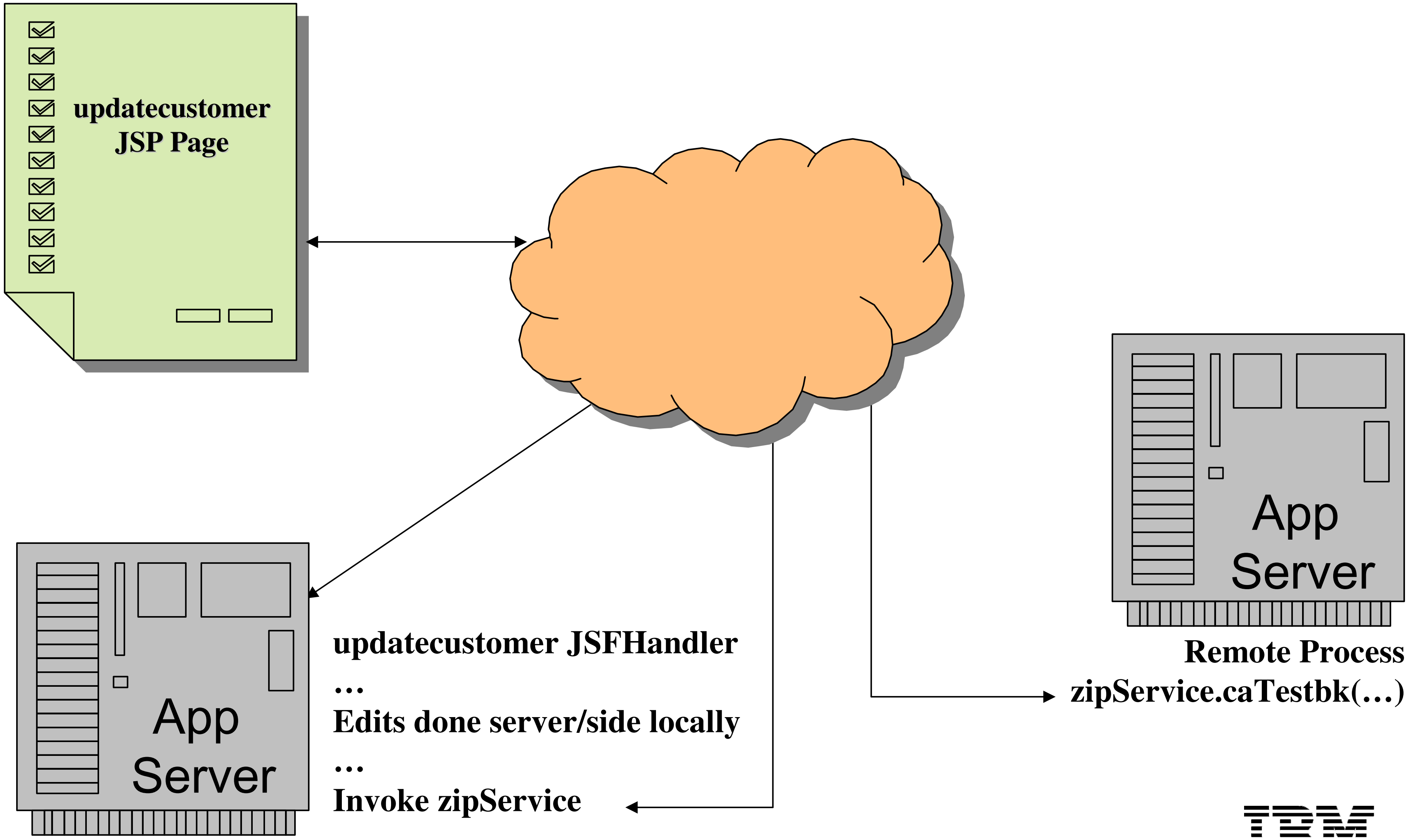
Port	Interface Name	Binding Name
✓ ZipService	services.IZipService	IZipService

☐ Update all existing bindings

- **Click Next** and then **Finish**

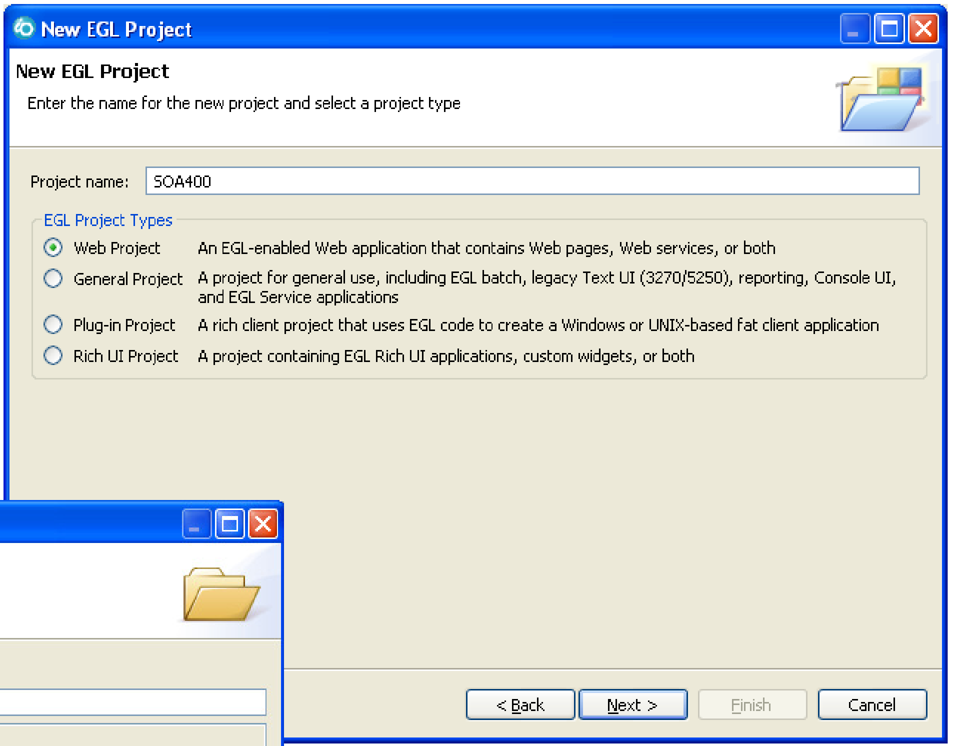
Optional Workshop – Call the Web Service From a Web Page

- Recall that you have an updatecustomer page that allows customers to update their state and zip code. We can invoke the ZipService as a Web Service, and test for California/Zips



Create and Consume a Database Service

- Create a new project: file → New → Project
- Name it SOA400
- Create as an EGL Web Project
- Create a folder name wsdl under WebContent by right clicking “WebContent” → New → Folder



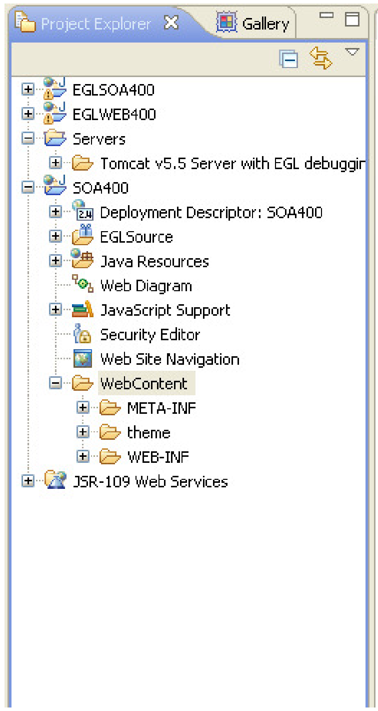
New EGL Project

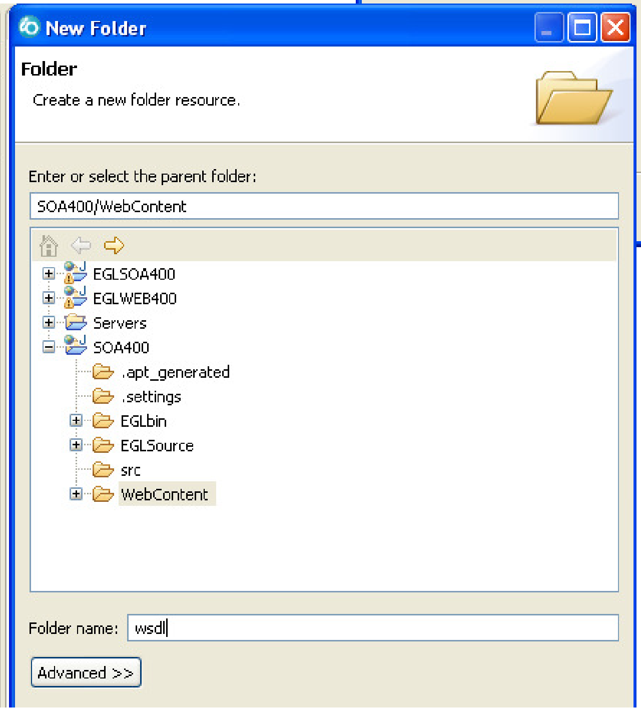
Enter the name for the new project and select a project type

Project name: SOA400

EGL Project Types

[✓ Web Project]	An EGL-enabled Web application that contains Web pages, Web services, or both
General Project	A project for general use, including EGL batch, legacy Text UI (3270/5250), reporting, Console UI, and EGL Service applications
Plug-in Project	A rich client project that uses EGL code to create a Windows or UNIX-based fat client application
Rich UI Project	A project containing EGL Rich UI applications, custom widgets, or both





Folder

Create a new folder resource.

Enter or select the parent folder:

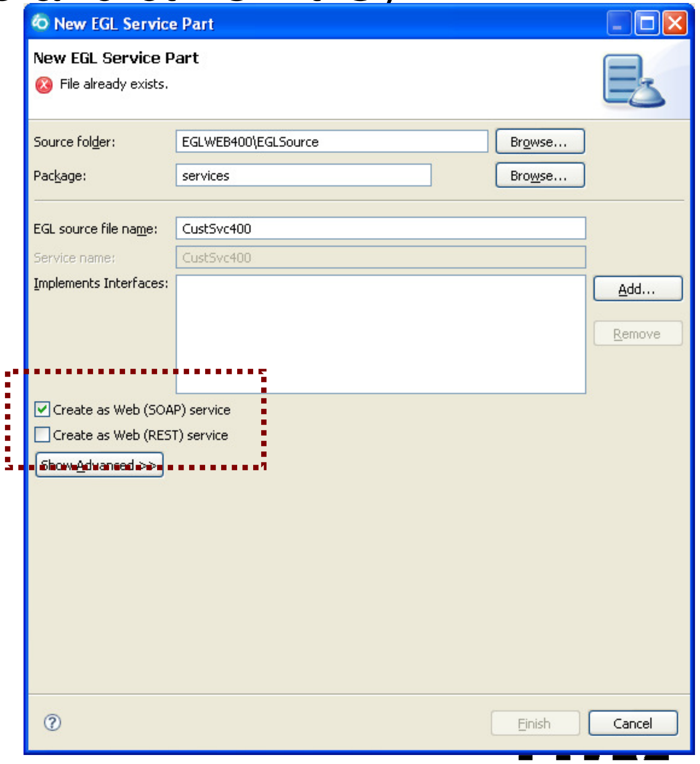
SOA400/WebContent

Folder name: wsdl

⏭ Create and Consume a Database Service

- Create a AS400 Web Service to access database records
- Create a new service. If you do not have a service package set up, right click EGLSource → New → Package and name it "Services"
- Create a new service by Right click Service Package and →New → Service, name it "CustSvc400".
- **Check "Create as Web (SOAP) service check box. (Very Important)**
- Enter the code below: (you can use copy/paste and ctrl-shift-O)

```
package services;

import iseriesddemosibmcom.StatusRec;
import iseriesddemosibmcom.access.CustomerLib;
import iseriesddemosibmcom.data.Customer;

// service
service CustSvc400

// Variable Declarations
status400 StatusRec;

// Function Declarations
function GetCustomer(searchRecord Customer inout, status400 StatusRec)
CustomerLib.GetCustomer(searchRecord, status400);
end

function GetCustomerListAll(customerArray Customer[]  out, status400 StatusRec);
CustomerLib.GetCustomerListAll(customerArray, status400);
end

end
```


New EGL Service Part

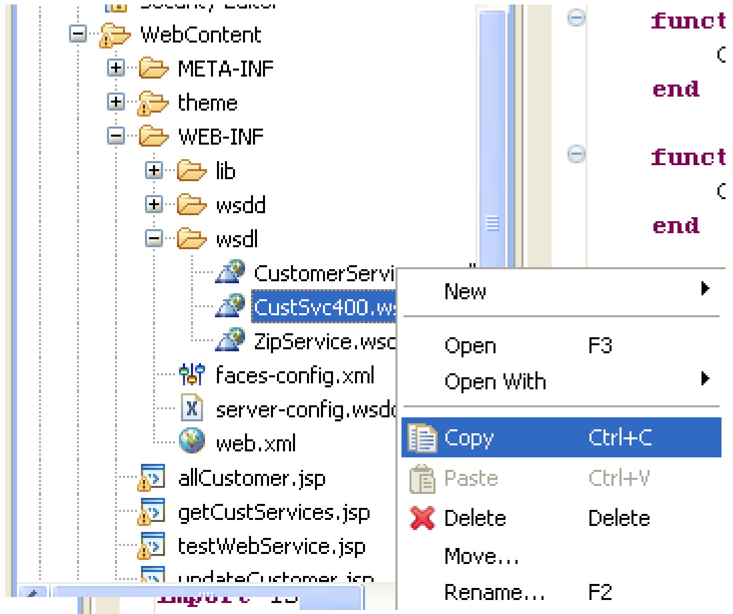
New EGL Service Part
File already exists.

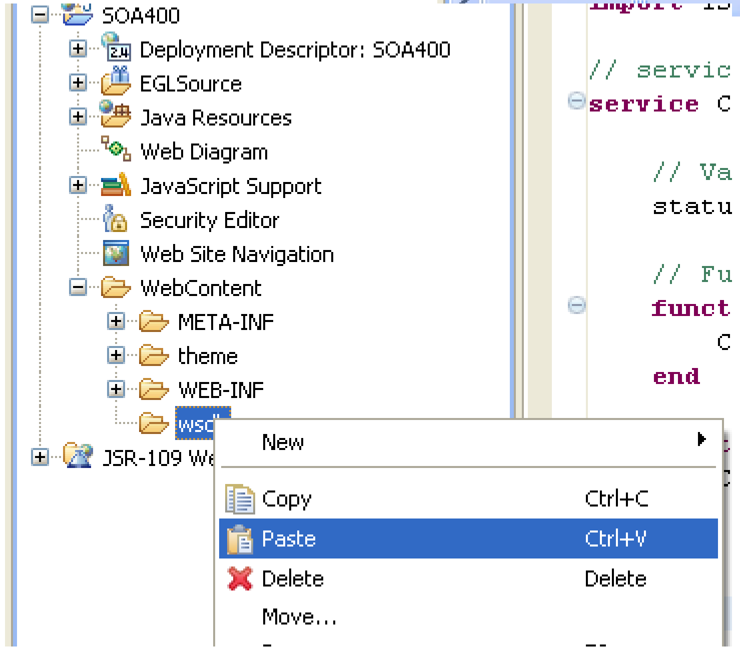
Source folder: EGLWEB400\EGLSource
Package: services
EGL source file name: CustSvc400
Service name: CustSvc400
Implements Interfaces:

[✓] Create as Web (SOAP) service
[ ] Create as Web (REST) service

Create and Consume a Database Service

- Generate your project by focus on the project name, right click and select “Generate”
- Find the generated CustSvc400.wsdl file in WebContent/wsdl folder, focus on it, right click and select “Copy”
- Paste the wsdl into SOA400/ WebContent/wsdl folder

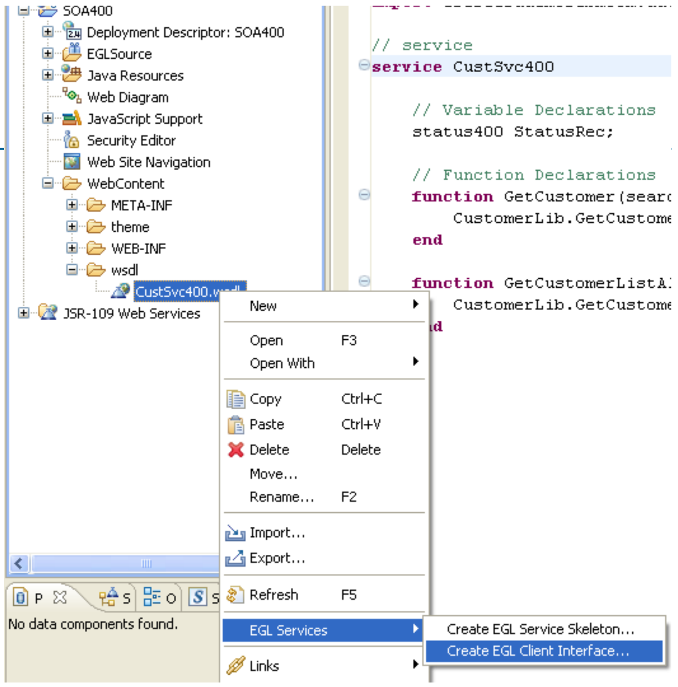




⏭ Create and Consume a Database Service

Create a client interface for the wsdl in the new project

- **Go to the new project SOA400**, right click on CustSvc400.wsdl
- Select → EGL Services → Create EGL Client Interface
- Click "Next"

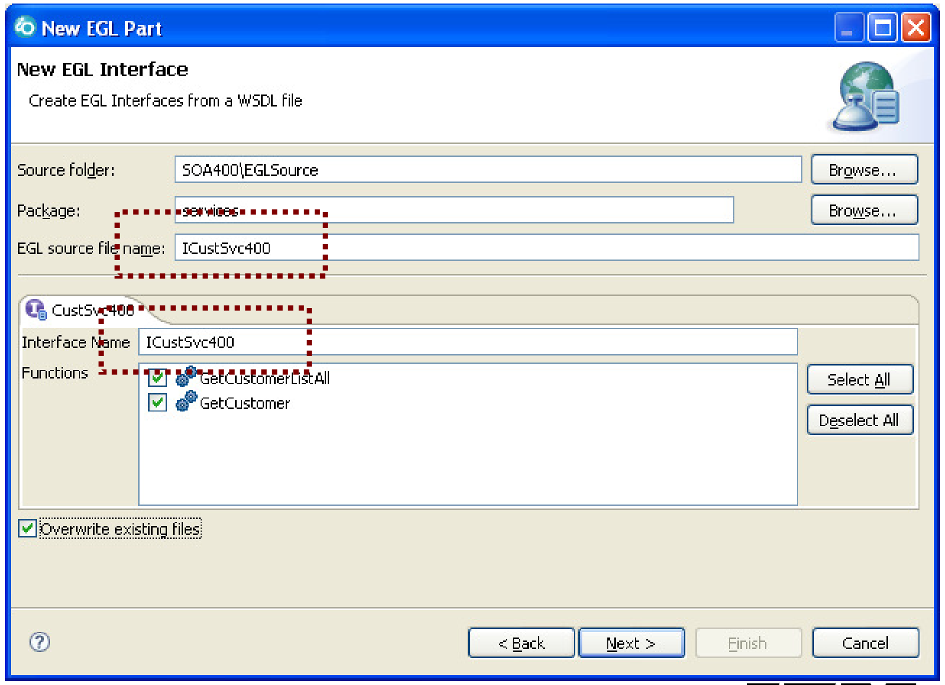


```
// service
service CustSvc400

    // Variable Declarations
    status400 StatusRec;

    // Function Declarations
    function GetCustomer(searc
        CustomerLib.GetCustome
    end

    function GetCustomerListA
        CustomerLib.GetCustome
    d
```

- Name both EGL Source File Name and Interface name as "ICustSvc400"

  It is good practice to name the web interface as "I" + the Web Service name.

- Click Next
- Click Finish
- Generate the SOA400 Project


New EGL Part

New EGL Interface

Create EGL Interfaces from a WSDL file

Source folder: SOA400\EGLSource

Package: services

EGL source file name: ICustSvc400

CustSvc400

Interface Name: ICustSvc400

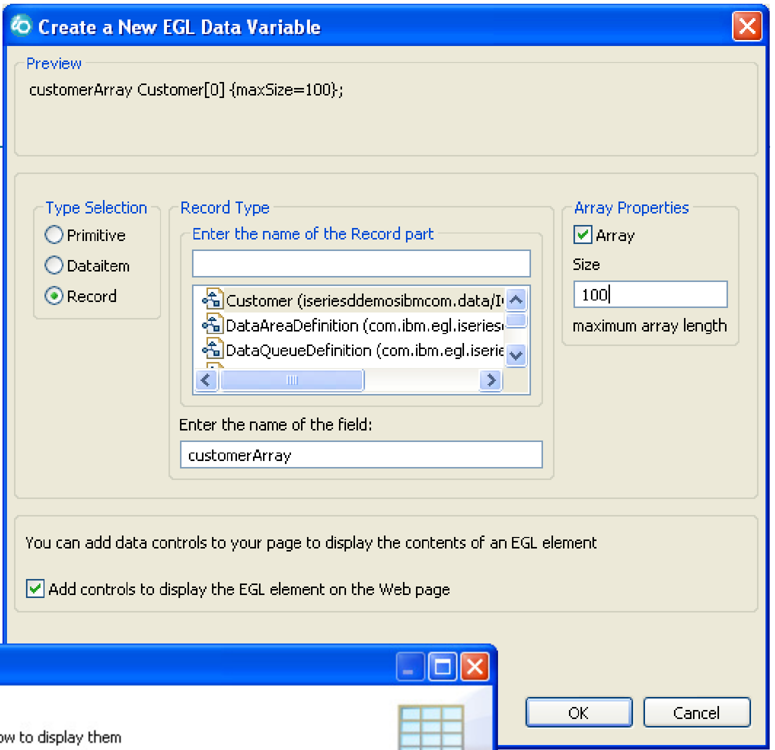
Functions:
- ✓ GetCustomerListAll
- ✓ GetCustomer

✓ Overwrite existing files

Create and Consume a Database Service

Let's consume the SOAP web service you just created

- In SOA400 project, create a new web page:
- SOA400/WebContent → right click → New → Web Page
- Name it "svcAllCust.jsp", select a template and click OK,
- Drag "New Variable" from Pallet to the page and select "Customer"
- Make it an Array.
- Check "Display Existing Record"
- Click "None"
- Select CustomerId, FirstName, LastName and Phone.
- Click Finish



Create a New EGL Data Variable

Preview

customerArray Customer[0] {maxSize=100};

Type Selection: Primitive / Dataitem / [✓ Record]

Record Type

Enter the name of the Record part

Customer (iseriesddemosibmcom.data/I
DataAreaDefinition (com.ibm.egl.iseries
DataQueueDefinition (com.ibm.egl.iserie

Enter the name of the field: customerArray

Array Properties

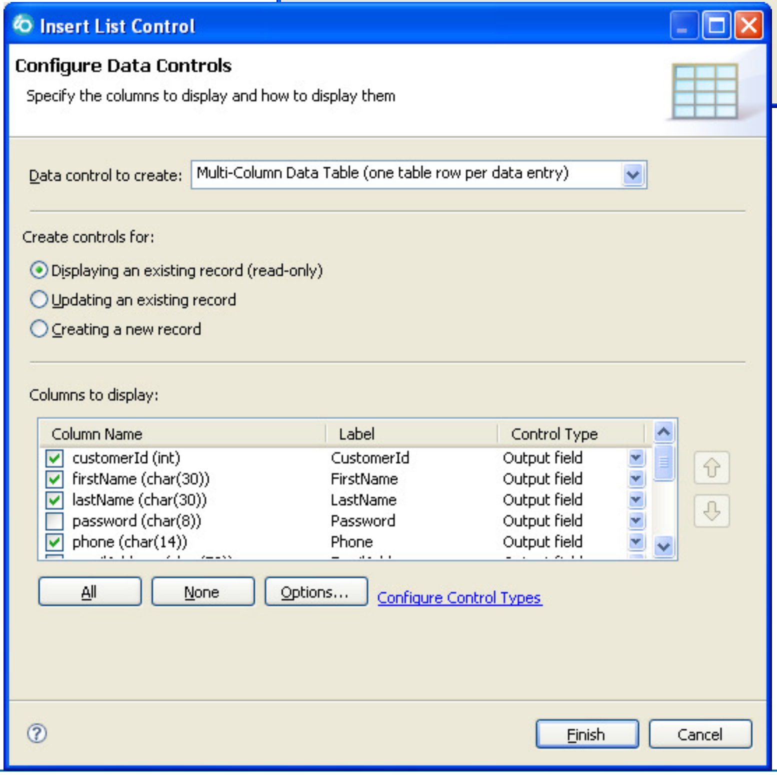
☑ Array

Size: 100

maximum array length

You can add data controls to your page to display the contents of an EGL element

☑ Add controls to display the EGL element on the Web page



Insert List Control

Configure Data Controls

Specify the columns to display and how to display them

Data control to create: Multi-Column Data Table (one table row per data entry)

Create controls for:

[✓ Displaying an existing record (read-only)]
Updating an existing record
Creating a new record

Columns to display:

	Column Name	Label	Control Type
✓	customerId (int)	CustomerId	Output field
✓	firstName (char(30))	FirstName	Output field
✓	lastName (char(30))	LastName	Output field
	password (char(8))	Password	Output field
✓	phone (char(14))	Phone	Output field

⏭ Create and Consume a Database Service

- Edit the Page Code

- Add the following lines of code

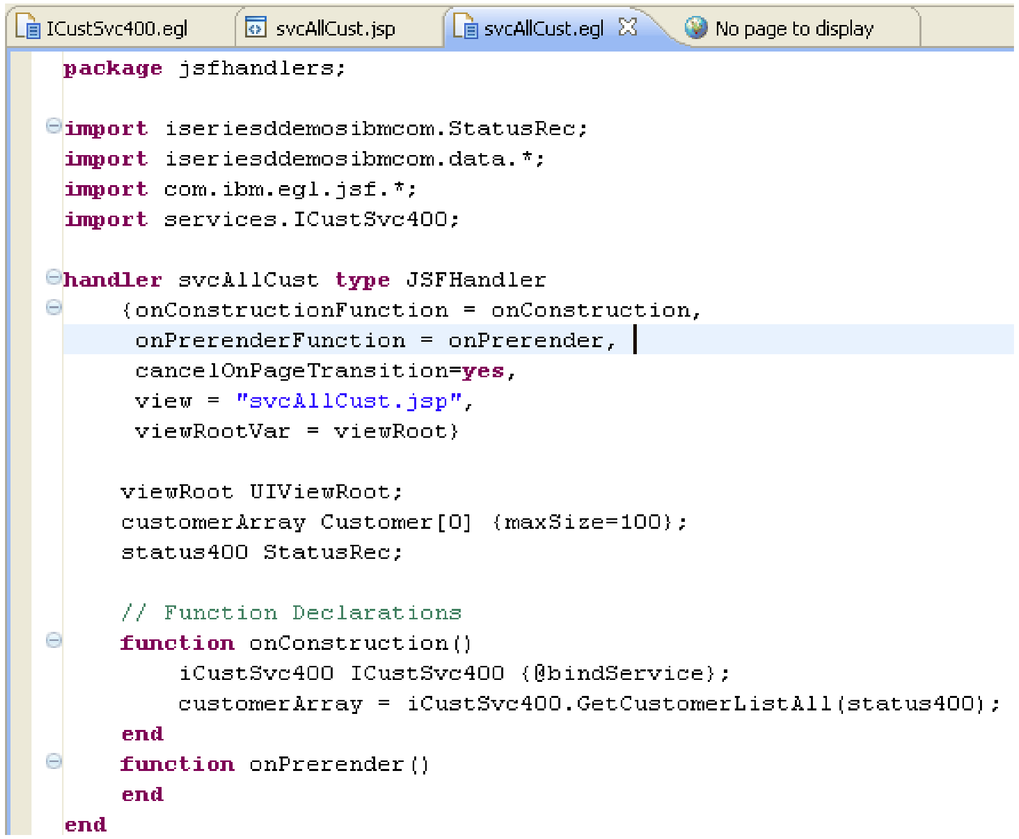
  - USING CONTENT ASSIST! (please ☺)

```
status400 StatusRec;

// Function Declarations
function onConstruction()
    iCustSvc400 ICustSvc400 {@bindService};
    customerArray = iCustSvc400.GetCustomerListAll(status400);
end
```

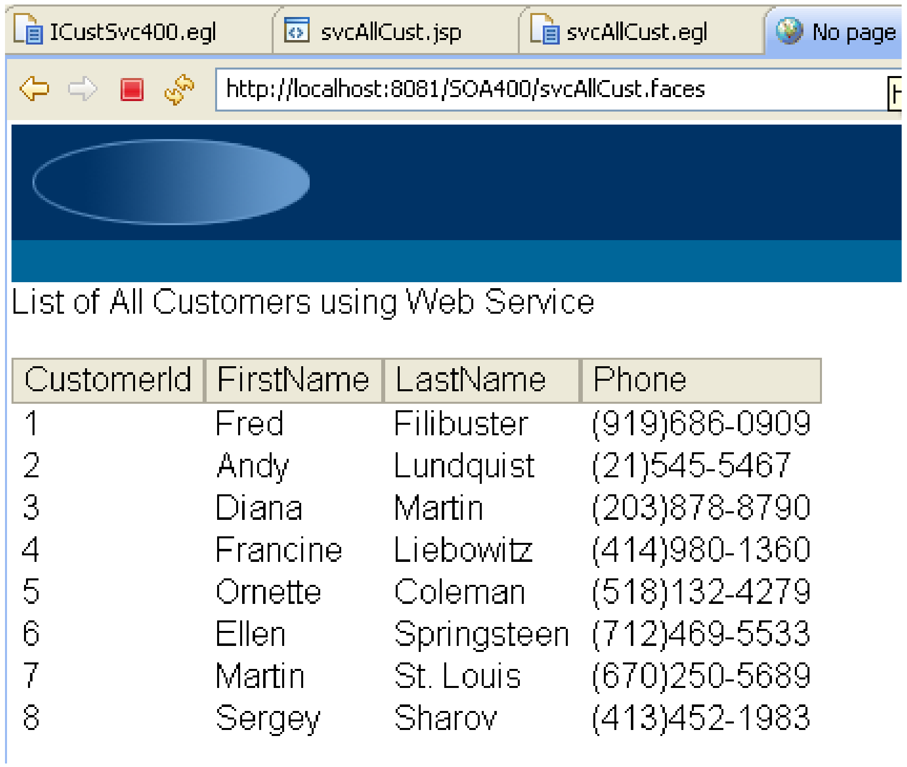
- Run the page

⏭ Create and Consume a Database Service

- You just successfully created and consumed a database web service in minutes.
- This would normally take a skilled SOA developer at least two days to complete.



```
package jsfhandlers;

import iseriesddemosibmcom.StatusRec;
import iseriesddemosibmcom.data.*;
import com.ibm.egl.jsf.*;
import services.ICustSvc400;

handler svcAllCust type JSFHandler
    {onConstructionFunction = onConstruction,
     onPrerenderFunction = onPrerender,
     cancelOnPageTransition=yes,
     view = "svcAllCust.jsp",
     viewRootVar = viewRoot}

    viewRoot UIViewRoot;
    customerArray Customer[0] {maxSize=100};
    status400 StatusRec;

    // Function Declarations
    function onConstruction()
        iCustSvc400 ICustSvc400 {@bindService};
        customerArray = iCustSvc400.GetCustomerListAll(status400);
    end
    function onPrerender()
    end
end
```



List of All Customers using Web Service

CustomerId	FirstName	LastName	Phone
1	Fred	Filibuster	(919)686-0909
2	Andy	Lundquist	(21)545-5467
3	Diana	Martin	(203)878-8790
4	Francine	Liebowitz	(414)980-1360
5	Ornette	Coleman	(518)132-4279
6	Ellen	Springsteen	(712)469-5533
7	Martin	St. Louis	(670)250-5689
8	Sergey	Sharov	(413)452-1983

Optional Topic – Consuming a 3rd Party Web Service

- EGL/RBD also allows you to call external (3rd Party-Provided) Web Services. If time remains in your lab, try the following short workshop to see for yourself!

Here are the steps you follow (from 10,000 feet):

Using a 3rd Party WSDL Steps:

1. Identify/Locate the **WSDL** file from which the interface will be created
2. Download this **WSDL** to your project
3. Create an **EGL Client Interface**
4. Create a calling (client) JSFHandler page

It's really that easy! But a note of caution first:

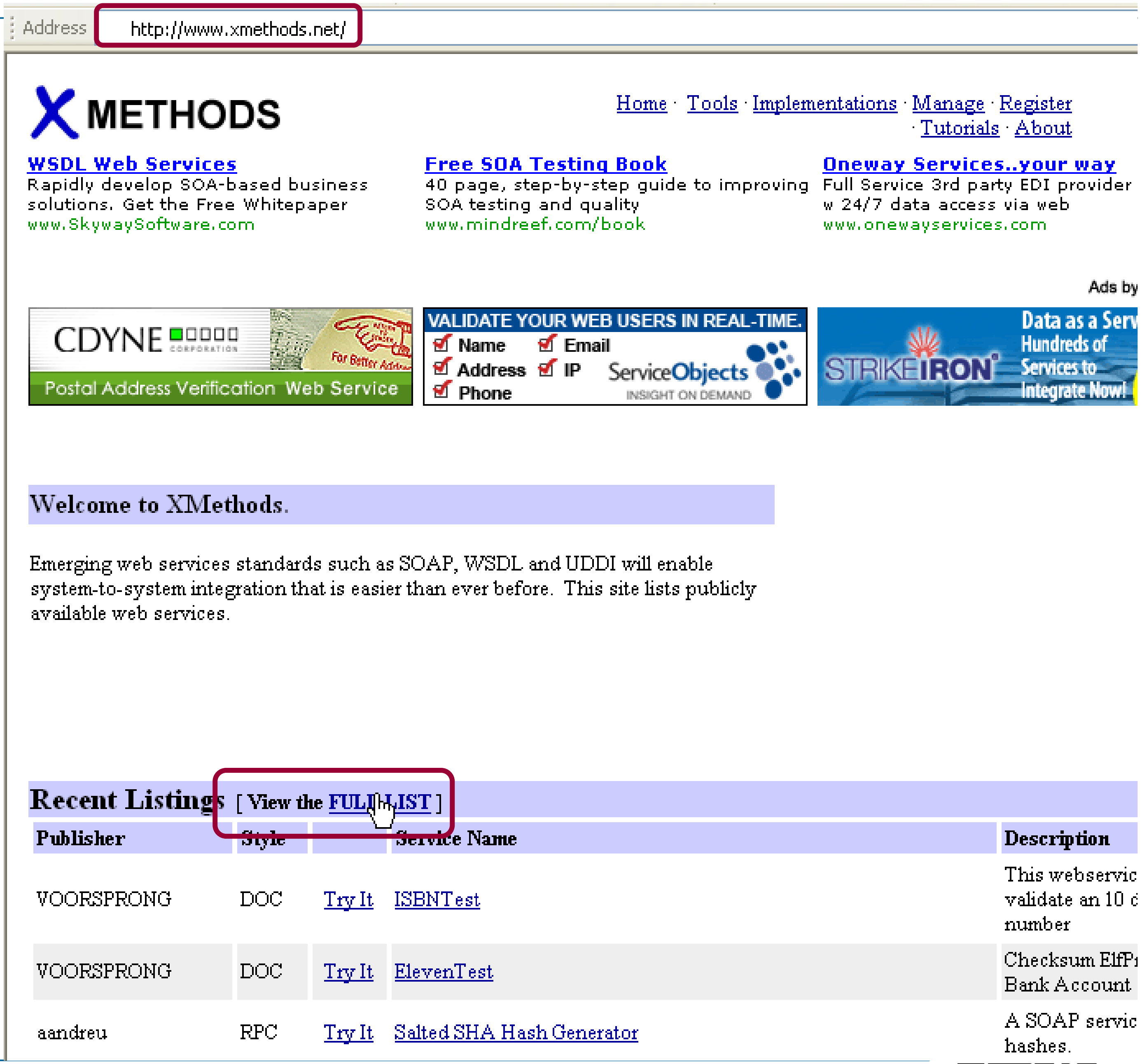
- This next lab requires that you be running a live connection to the internet – free and clear of any corporate fire-walls, or it most likely will not execute. **(See ***Notes)** Not withstanding, if you will be using 3rd Party services it's probably worth understanding the work-flow steps to implement with EGL.
- The web service used in the lab is a true 3rd Party service. As such it may not be running when you test it – as this is not under our control.

⏭ Consuming a 3rd Party Web Service – Find the Service

1. **Identify the WSDL file from which the interface will be created – and download this WSDL to your project**

- ⏭ **From the Xmethods.net site**
- ⏭ **Click:**

View the <u>FULL LIST</u>


Address http://www.xmethods.net/

XMETHODS

Home · Tools · Implementations · Manage · Register · Tutorials · About

WSDL Web Services
Rapidly develop SOA-based business solutions. Get the Free Whitepaper
www.SkywaySoftware.com

Free SOA Testing Book
40 page, step-by-step guide to improving SOA testing and quality
www.mindreef.com/book

Oneway Services..your way
Full Service 3rd party EDI provider w 24/7 data access via web
www.onewayservices.com

Ads by

CDYNE CORPORATION
Postal Address Verification Web Service
VALIDATE YOUR WEB USERS IN REAL-TIME.
Name
Email
Address
IP
Phone
ServiceObjects
INSIGHT ON DEMAND
STRIKEIRON
Data as a Serv
Hundreds of Services to Integrate Now!

Welcome to XMethods.

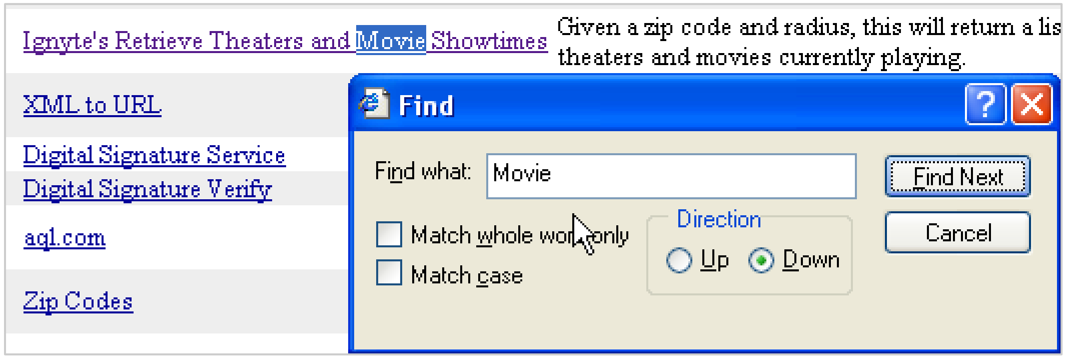
Emerging web services standards such as SOAP, WSDL and UDDI will enable system-to-system integration that is easier than ever before. This site lists publicly available web services.

Recent Listings [View the FULL LIST]

Publisher	Style		Service Name	Description
VOORSPRONG	DOC	Try It	ISBNTest	This webservic validate an 10 d number
VOORSPRONG	DOC	Try It	ElevenTest	Checksum ElfPr Bank Account
aandreu	RPC	Try It	Salted SHA Hash Generator	A SOAP servic hashes.

⏭ Consuming a 3rd Party Web Service – Select the WSDL

- ⏭ **From the Xmethods site full list, then scroll (or Ctrl/F** find) the **Ignyte Retrieve Theaters and Movie Showtimes** service



Ignyte's Retrieve Theaters and Movie Showtimes — Given a zip code and radius, this will return a lis theaters and movies currently playing.

XML to URL

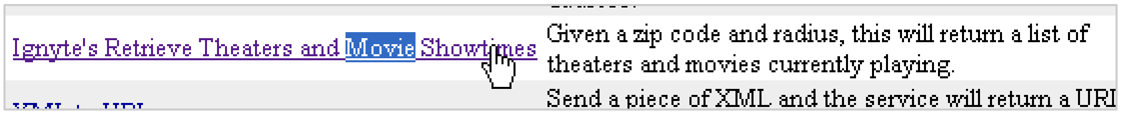
Digital Signature Service

Digital Signature Verify

aql.com

Zip Codes

- ⏭ **Click the link for the Web Service you wish to copy from the Xmethods site:**

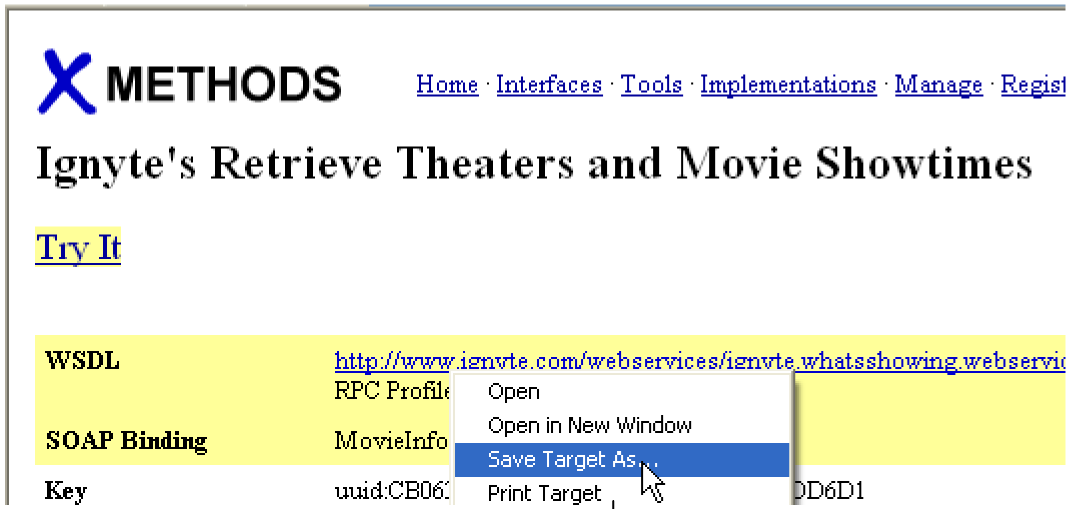


Ignyte's Retrieve Theaters and Movie Showtimes — Given a zip code and radius, this will return a list of theaters and movies currently playing.

Send a piece of XML and the service will return a URI

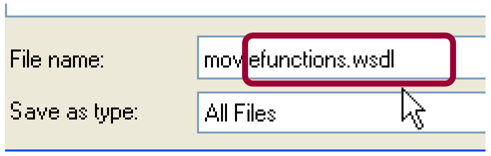
⏭ Consuming a 3rd Party Web Service – Save the WSDL

⏭ Save the WSDL to your project's **\WebContent\WEB-INF\wsdl\** folder as: **moviefunctions.wsdl** (note you will need to rename the extension in the project and probably Refresh (F5) the project directory you save into)



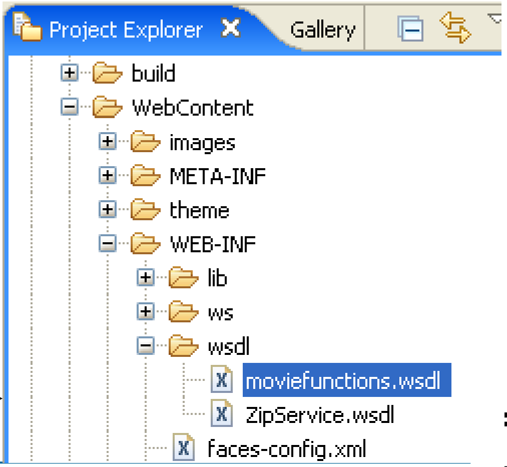
Ignyte's Retrieve Theaters and Movie Showtimes

WSDL	http://www.ignyte.com/webservices/ignyte.whatsshowing.webservic RPC Profile
SOAP Binding	MovieInfo
Key	uuid:CB06 DD6D1



File name: moviefunctions.wsdl

Save as type: All Files



⏭ Consuming a 3rd Party Web Service – Create the EGL Interface

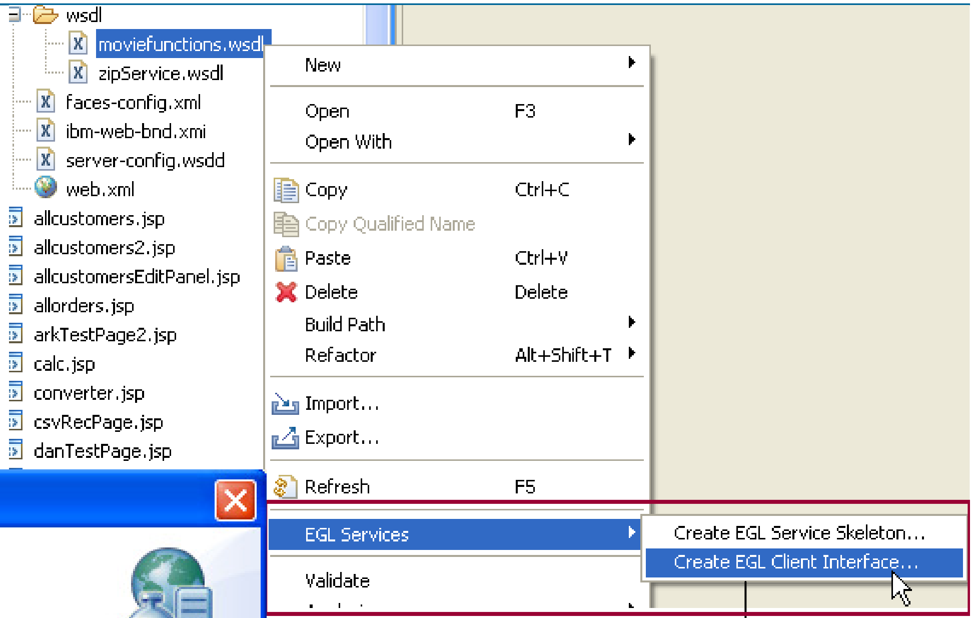
⏭ **Right click over moviefunctions.wsdl – select:**

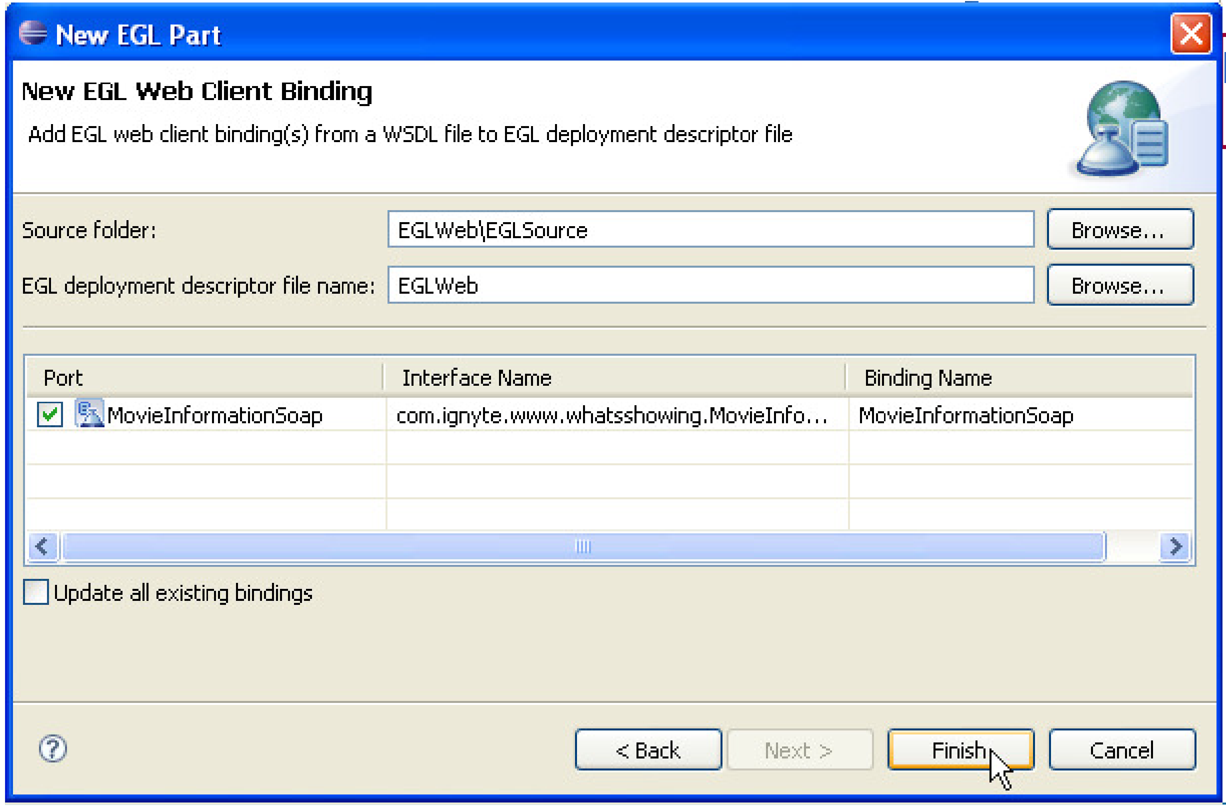
- **EGL Services > Create EGL Client Interface…**

- **Click Next > two times**

From New EGL Web Client Binding

- Click **Finish**





New EGL Part

New EGL Web Client Binding

Add EGL web client binding(s) from a WSDL file to EGL deployment descriptor file

Source folder: EGLWeb\EGLSource

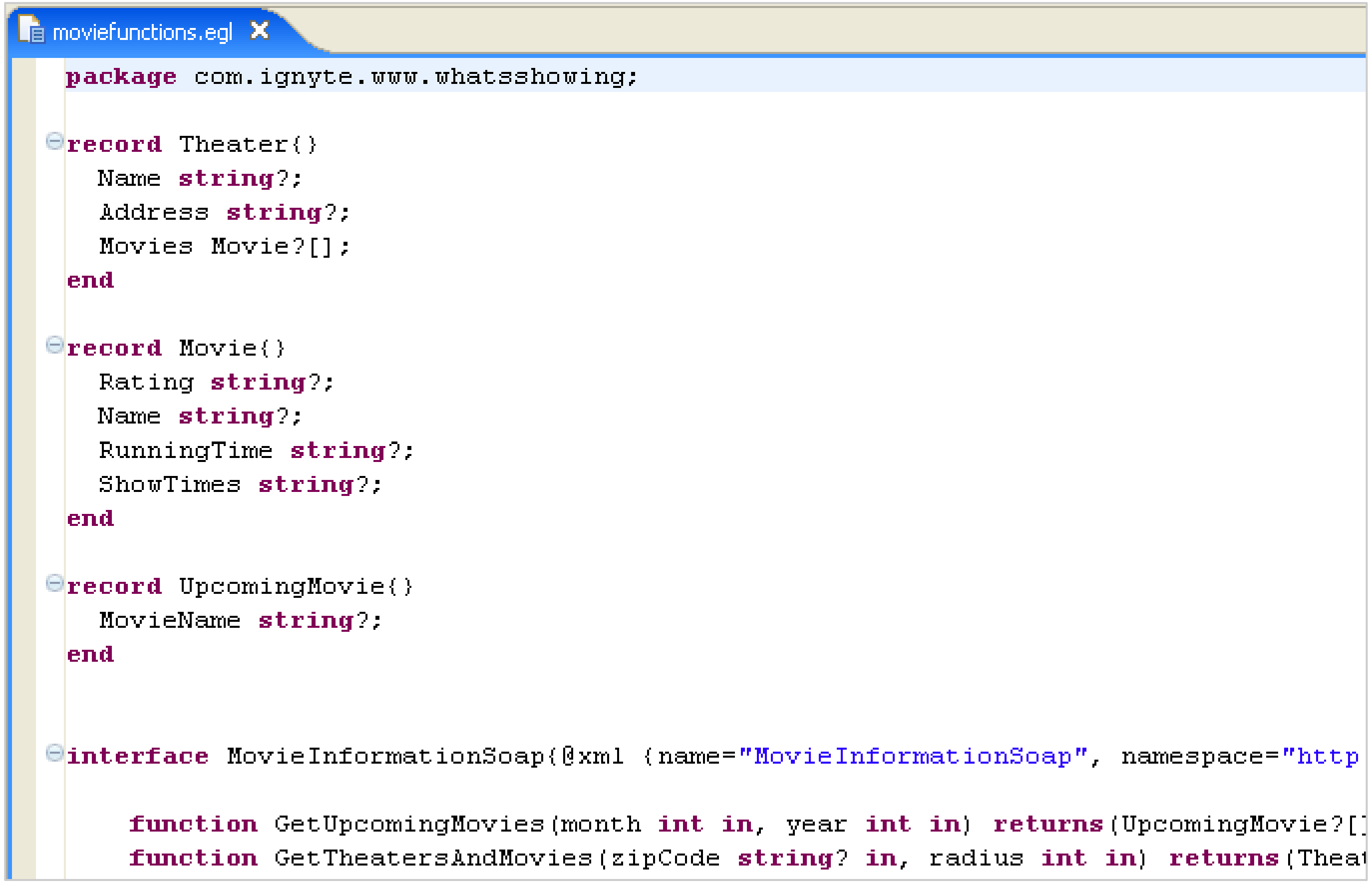
EGL deployment descriptor file name: EGLWeb

Port	Interface Name	Binding Name
✓ MovieInformationSoap	com.ignyte.www.whatsshowing.MovieInfo…	MovieInformationSoap

☐ Update all existing bindings

Consuming a 3rd Party Web Service – The EGL Interface

- This will create an EGL Interface to the 3rd Party Web Service – that you can now reference in (call from) client business logic.



moviefunctions.egl

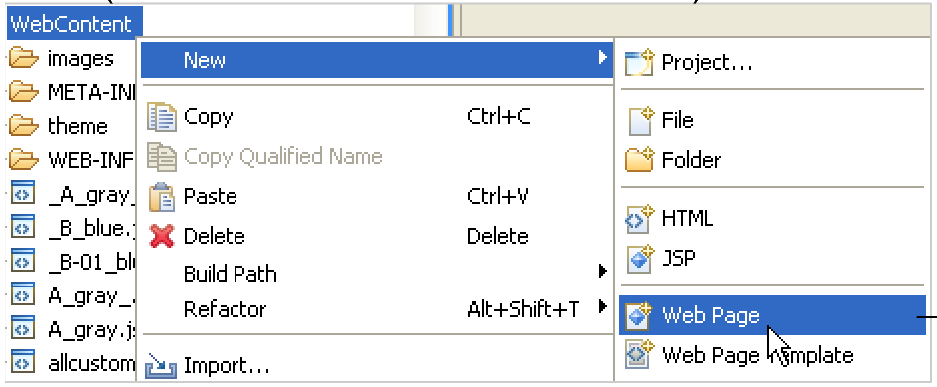
```
package com.ignyte.www.whatsshowing;

record Theater{}
  Name string?;
  Address string?;
  Movies Movie?[];
end

record Movie{}
  Rating string?;
  Name string?;
  RunningTime string?;
  ShowTimes string?;
end

record UpcomingMovie{}
  MovieName string?;
end


interface MovieInformationSoap{@xml {name="MovieInformationSoap", namespace="http

    function GetUpcomingMovies(month int in, year int in) returns(UpcomingMovie?[]
    function GetTheatersAndMovies(zipCode string? in, radius int in) returns(Theat
```

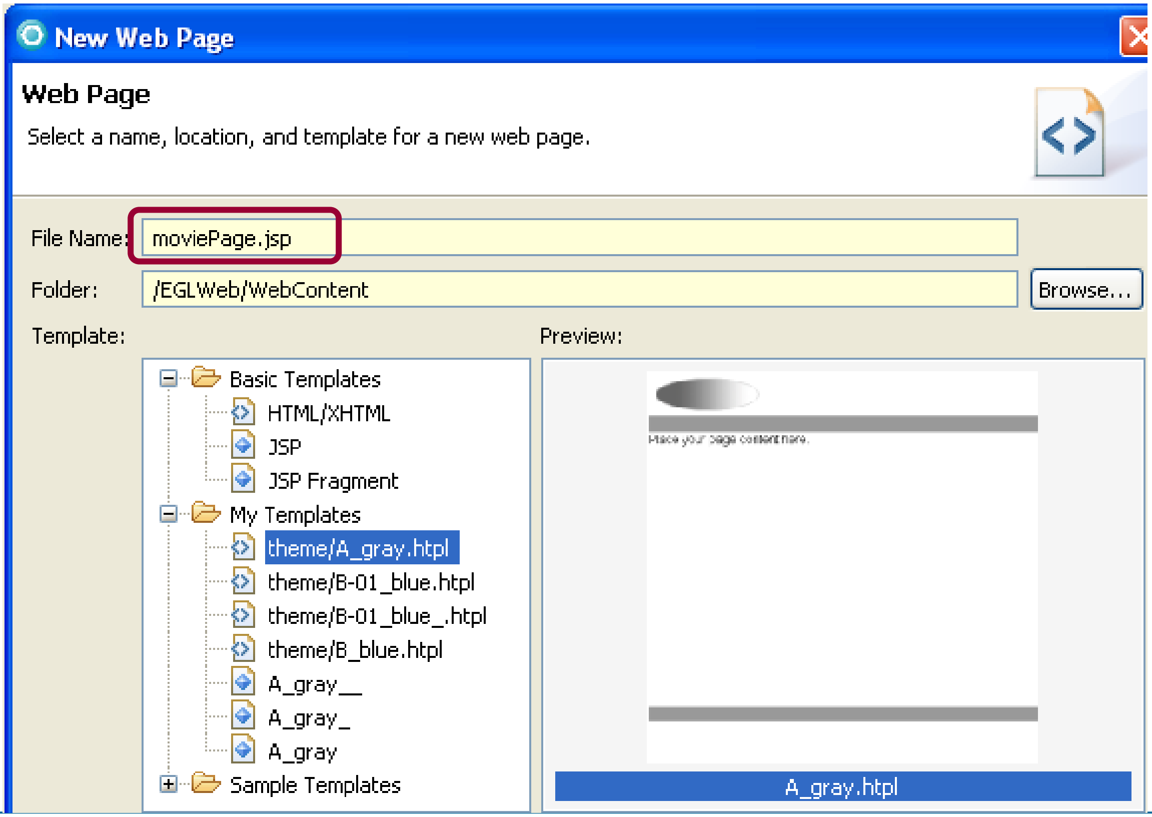
But first we need to generate all the new Java Classes for all these new project resources

- (From Project Explorer) **Generate your project** – note that you may receive Java Generation errors, which will clean themselves up when you create the page (next part of lab)

⏭ Consuming a 3rd Party Web Service – Create a New Web Page

⏭ (From the **WebContent**\ folder) Create a new Web Page named: **moviePage.jsp**




New Web Page

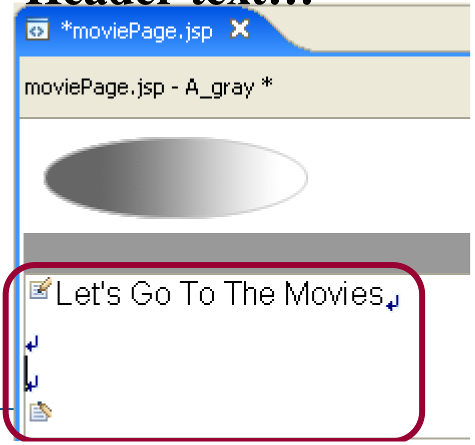
Web Page

Select a name, location, and template for a new web page.

File Name: moviePage.jsp

Folder: /EGLWeb/WebContent

Change the default Header text…



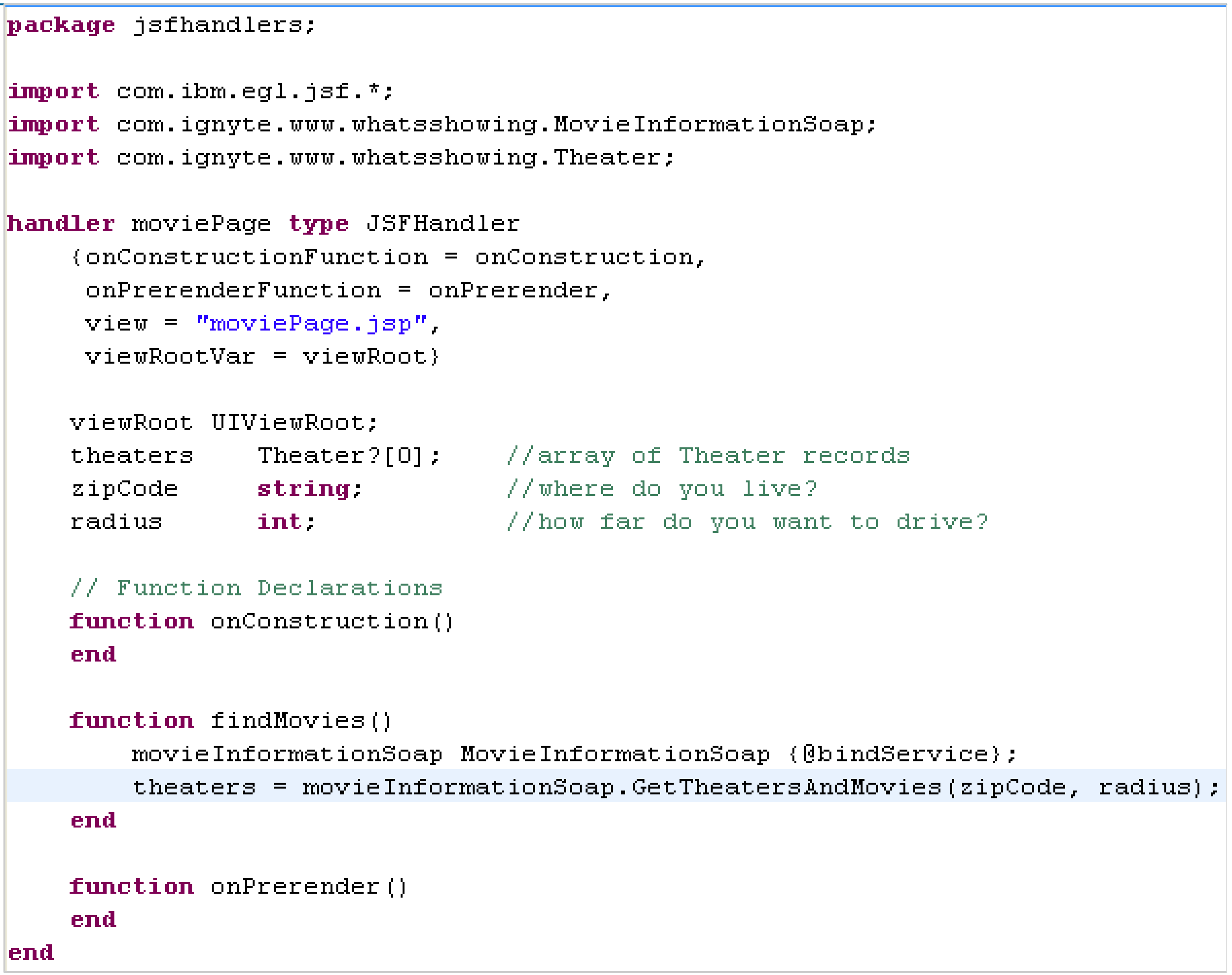
Let's Go To The Movies

⏭ Consuming a 3rd Party Web Service – Edit the JSFHandler Code

- Edit the Page Code, (**moviePage.egl**) and using Content Assist create the following JSFHandler logic

Note – please either use **Content Assist** to create this logic, or copy/paste the completed code from the Notes section of this slide.

- **Press Ctrl/S** – and clean up any typos or syntax errors ☺



```
package jsfhandlers;

import com.ibm.egl.jsf.*;
import com.ignyte.www.whatsshowing.MovieInformationSoap;
import com.ignyte.www.whatsshowing.Theater;

handler moviePage type JSFHandler
    {onConstructionFunction = onConstruction,
     onPrerenderFunction = onPrerender,
     view = "moviePage.jsp",
     viewRootVar = viewRoot}

    viewRoot UIViewRoot;
    theaters    Theater?[0];     //array of Theater records
    zipCode     string;          //where do you live?
    radius      int;             //how far do you want to drive?

    // Function Declarations
    function onConstruction()
    end

    function findMovies()
        movieInformationSoap MovieInformationSoap {@bindService};
        theaters = movieInformationSoap.GetTheatersAndMovies(zipCode, radius);
    end

    function onPrerender()
    end
end
```

⏭ Consuming a 3rd Party Web Service – Edit the JSFHandler Code

From Page Designer

- ⏭ **Press Ctrl/S** – to refresh/synchronize the tools

From Page Data

- ⏭ Hold down the **Ctrl** key, and select:

  `radius – int`

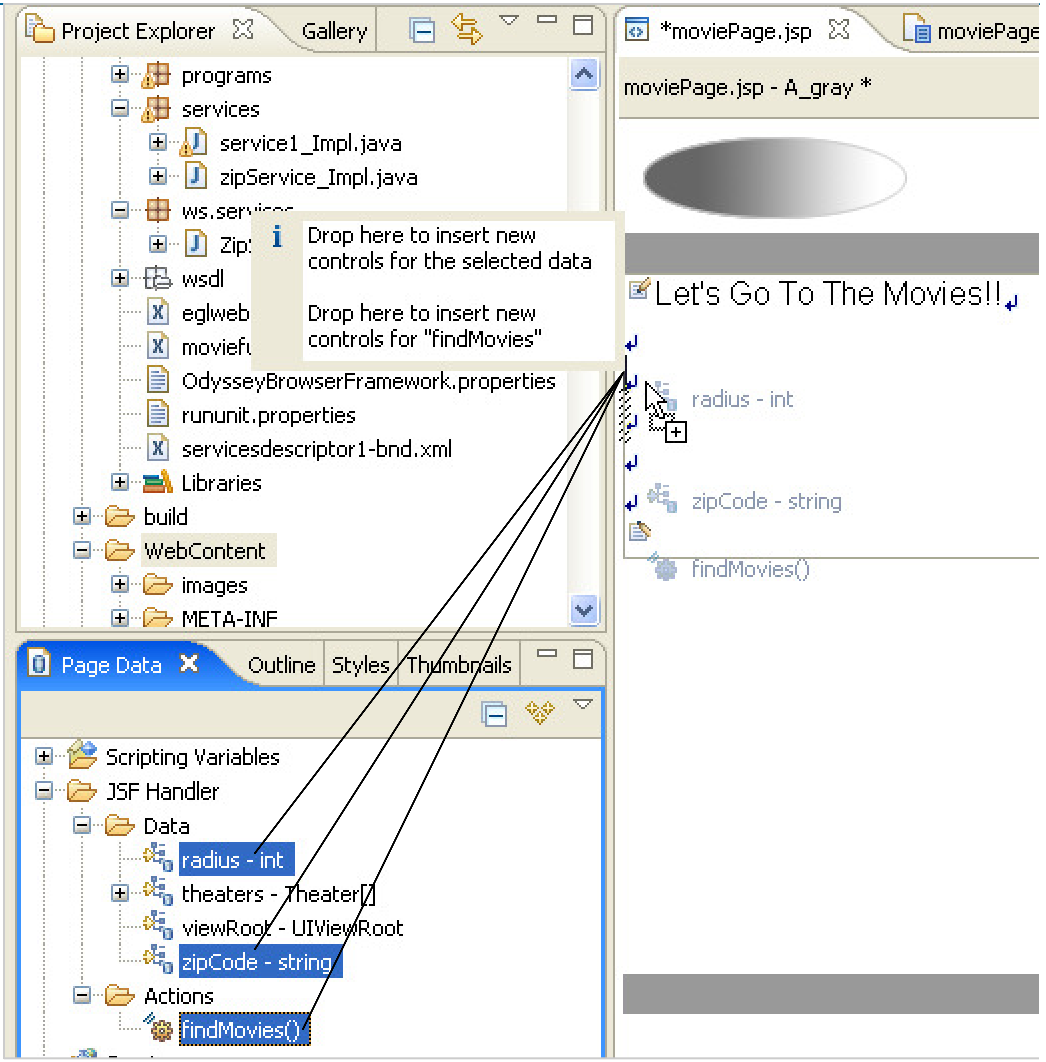
  `zipCode – string`

  `findMovies()`

- ⏭ Drag and drop them onto the page
- ⏭ Make the fields Input fields

  ⦿ **Updating an existing record**

- ⏭ Delete any extraneous Submit Buttons created



⏭ Consuming a 3rd Party Web Service – Edit the JSFHandler Code

From Page Data

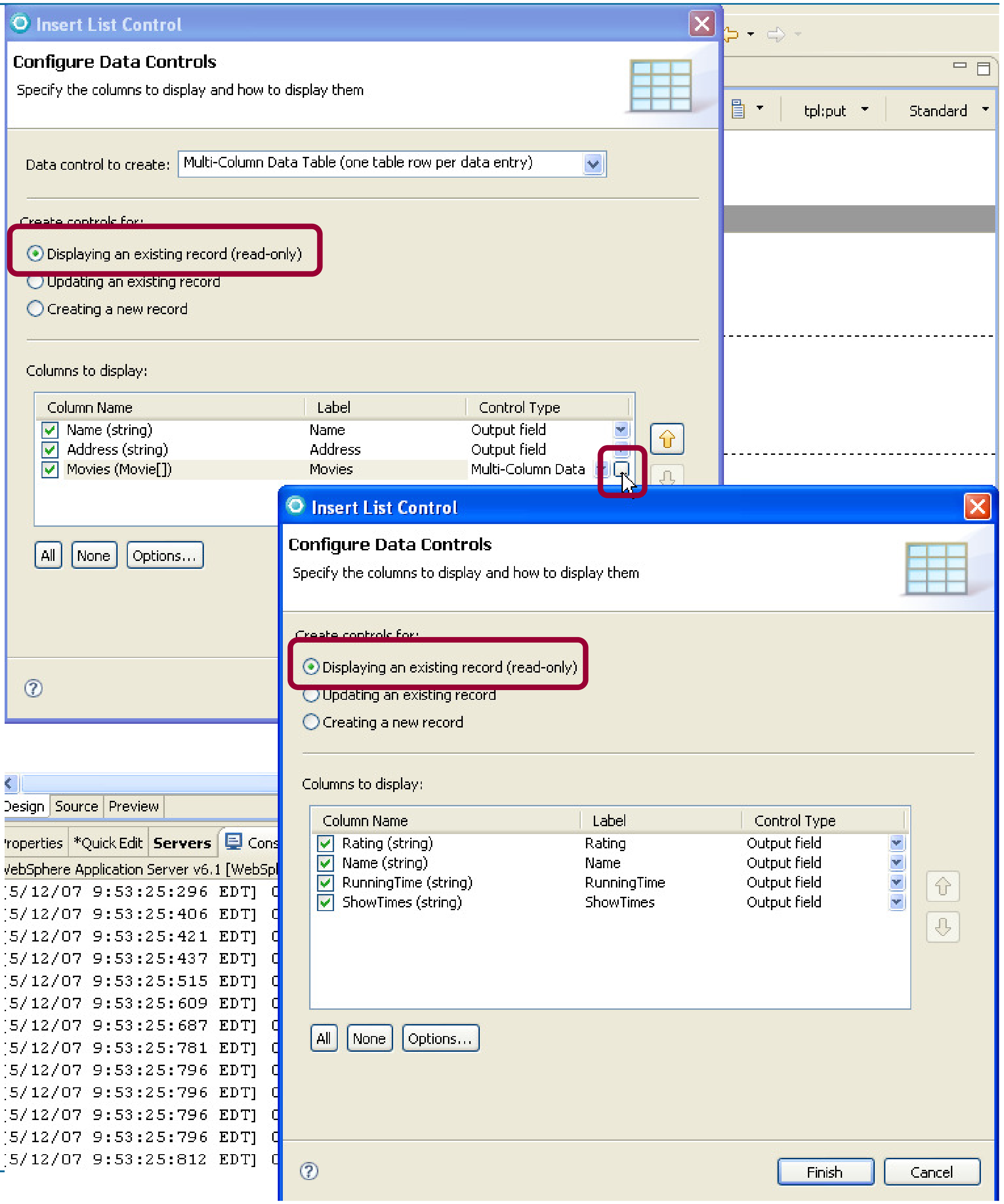
- ⏭ Select:

  `theaters – Theater[]`

- ⏭ Drag and drop the array onto the page
- ⏭ Make all columns read-only
  ⦿Displaying an existing record
- ⏭ Click the ellipsis next to **Multi-Column Data**
- ⏭ Make the nested columns read-only
  ⦿Displaying an existing record
- ⏭ Click Finish twice

OPTIONAL

- ⏭ With the table selected, feel free to change display properties:
  - ⏭ **Outer table:**
    - ⏭ **Border: 1**
  - ⏭ **Inner table:**
    - ⏭ **From Display options: delete columnClass1**



Insert List Control

Configure Data Controls

Specify the columns to display and how to display them

Data control to create: Multi-Column Data Table (one table row per data entry)

Create controls for:

- [✓ Displaying an existing record (read-only)]
- Updating an existing record
- Creating a new record

Columns to display:

Column Name	Label	Control Type
✓ Name (string)	Name	Output field
✓ Address (string)	Address	Output field
✓ Movies (Movie[])	Movies	Multi-Column Data

All | None | Options...

Insert List Control

Configure Data Controls

Specify the columns to display and how to display them

Create controls for:

- [✓ Displaying an existing record (read-only)]
- Updating an existing record
- Creating a new record

Columns to display:

Column Name	Label	Control Type
✓ Rating (string)	Rating	Output field
✓ Name (string)	Name	Output field
✓ RunningTime (string)	RunningTime	Output field
✓ ShowTimes (string)	ShowTimes	Output field

All | None | Options...

Finish | Cancel

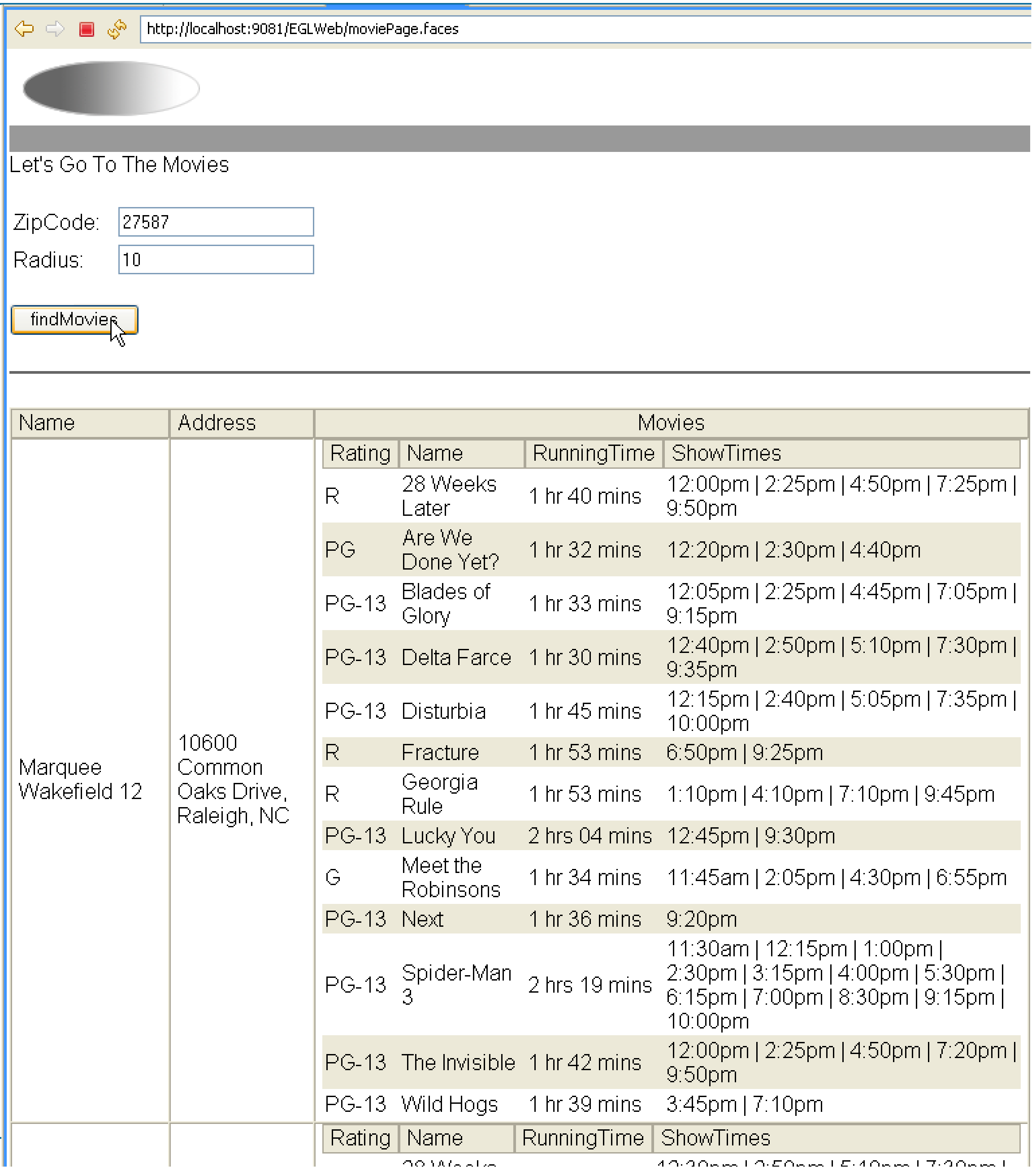
```
WebSphere Application Server v6.1 [WebSp
5/12/07 9:53:25:296 EDT] C
5/12/07 9:53:25:406 EDT] C
5/12/07 9:53:25:421 EDT] C
5/12/07 9:53:25:437 EDT] C
5/12/07 9:53:25:515 EDT] C
5/12/07 9:53:25:609 EDT] C
5/12/07 9:53:25:687 EDT] C
5/12/07 9:53:25:781 EDT] C
5/12/07 9:53:25:796 EDT] C
5/12/07 9:53:25:796 EDT] C
5/12/07 9:53:25:796 EDT] C
5/12/07 9:53:25:796 EDT] C
5/12/07 9:53:25:812 EDT] C
```

⏭ Consuming a 3rd Party Web Service – Edit the JSFHandler Code

- **Run the page**

- Enter a valid U.S. Postal Code (**ZipCode**: **90210**, **54903**, etc), and enter an integer number **(Radius)** which represents how far - in miles, you would be willing to travel to go to the movies.

Note that you must have a live, free and clear internet connection (sans Firewall) **for this page to work.**



http://localhost:9081/EGLWeb/moviePage.faces

Let's Go To The Movies

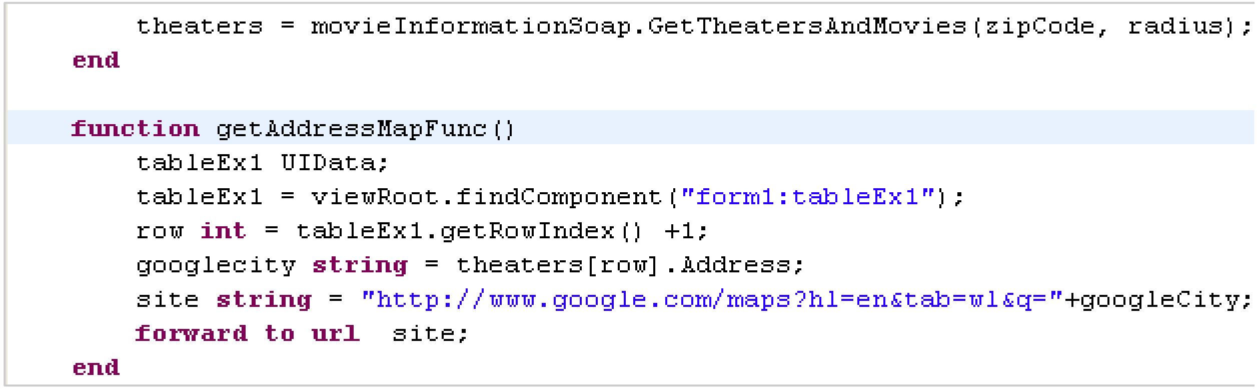
ZipCode:	27587
Radius:	10

Name	Address	Rating	Name	RunningTime	ShowTimes
Marquee Wakefield 12	10600 Common Oaks Drive, Raleigh, NC	R	28 Weeks Later	1 hr 40 mins	12:00pm | 2:25pm | 4:50pm | 7:25pm | 9:50pm
		PG	Are We Done Yet?	1 hr 32 mins	12:20pm | 2:30pm | 4:40pm
		PG-13	Blades of Glory	1 hr 33 mins	12:05pm | 2:25pm | 4:45pm | 7:05pm | 9:15pm
		PG-13	Delta Farce	1 hr 30 mins	12:40pm | 2:50pm | 5:10pm | 7:30pm | 9:35pm
		PG-13	Disturbia	1 hr 45 mins	12:15pm | 2:40pm | 5:05pm | 7:35pm | 10:00pm
		R	Fracture	1 hr 53 mins	6:50pm | 9:25pm
		R	Georgia Rule	1 hr 53 mins	1:10pm | 4:10pm | 7:10pm | 9:45pm
		PG-13	Lucky You	2 hrs 04 mins	12:45pm | 9:30pm
		G	Meet the Robinsons	1 hr 34 mins	11:45am | 2:05pm | 4:30pm | 6:55pm
		PG-13	Next	1 hr 36 mins	9:20pm
		PG-13	Spider-Man 3	2 hrs 19 mins	11:30am | 12:15pm | 1:00pm | 2:30pm | 3:15pm | 4:00pm | 5:30pm | 6:15pm | 7:00pm | 8:30pm | 9:15pm | 10:00pm
		PG-13	The Invisible	1 hr 42 mins	12:00pm | 2:25pm | 4:50pm | 7:20pm | 9:50pm
		PG-13	Wild Hogs	1 hr 39 mins	3:45pm | 7:10pm
		Rating	Name	RunningTime	ShowTimes

⏭ OPTIONAL – Call Google Maps using the Cinema's Address – 1 of 2

- What if you wanted to allow users to click on a cinema address, and have that bring up a GOOGLEMAP of that address? Here are the steps (note that you may need help from your instructor on this).

- ⏭ Copy/paste the code in this slide's Notes, as a new function:
- ⏭ **Ctrl/S** – save your code



```
        theaters = movieInformationSoap.GetTheatersAndMovies(zipCode, radius);
    end

    function getAddressMapFunc()
        tableEx1 UIData;
        tableEx1 = viewRoot.findComponent("form1:tableEx1");
        row int = tableEx1.getRowIndex() +1;
        googlecity string = theaters[row].Address;
        site string = "http://www.google.com/maps?hl=en&tab=wl&q="+googleCity;
        forward to url  site;
    end
```

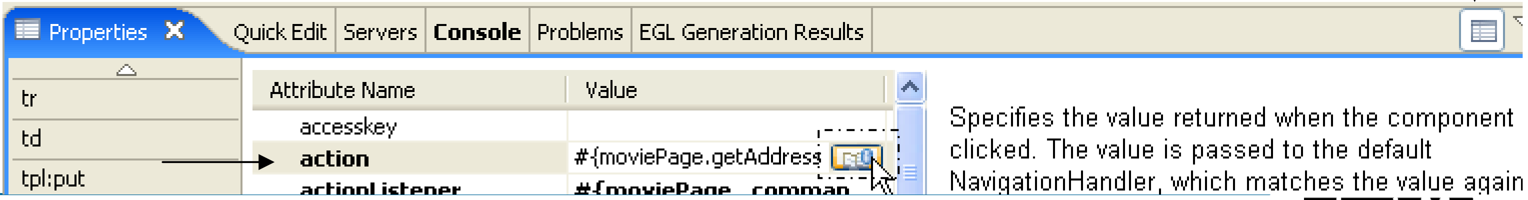
- ⏭ **From Page Designer**
  - ⏭ **From the Palette/Enhanced Faces Components – find: Link – Command**

    If it's not on the Palette you will have to Customize the Palette in order to show this option. Please ask your instructor for assistance on this – or, from the EGL-JSF.PPT, read the slides starting with the slide titled: Customizing the Palette

- ⏭ **Assuming you have the Link-Command in your Palette**
  - ⏭ **Select the Address field in the dataTable**

  - ⏭ **Double-Click the Link Command**
  - ⏭ **From Properties click the All Attributes icon**
  - ⏭ **From action, click the Browse icon and select: getAddressMapFunc()**

All Attributes

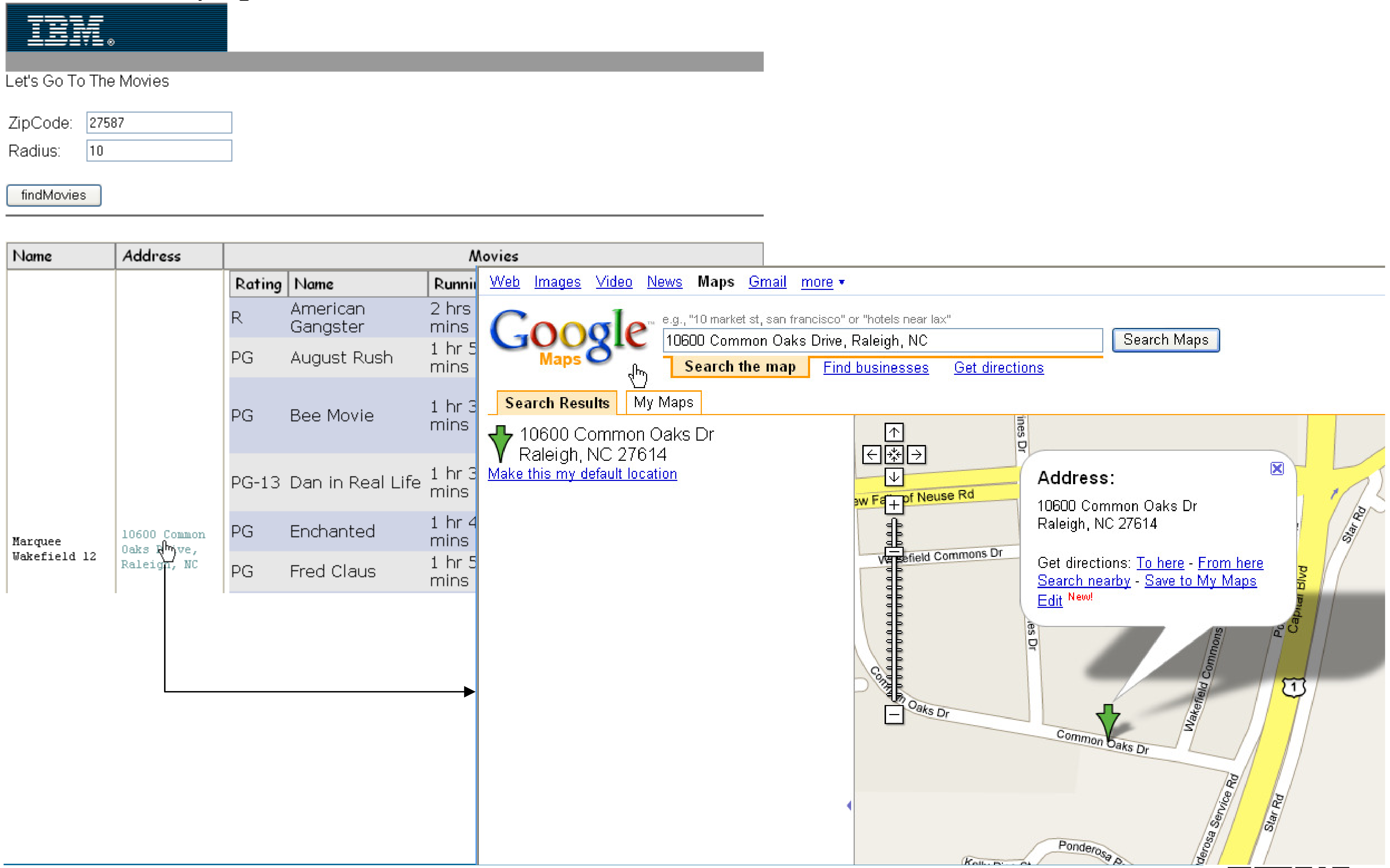


Attribute Name	Value
accesskey	
action	#{moviePage.getAddress
actionListener	#{moviePage._comman

Specifies the value returned when the component clicked. The value is passed to the default NavigationHandler, which matches the value again

⏭ OPTIONAL – Call Google Maps using the Cinema's Address – 2 of 2

⏭ Run the page. Find movies and click an Address



Let's Go To The Movies

ZipCode:	27587
Radius:	10

findMovies

Name	Address	Movies		
		Rating	Name	Runni
Marquee Wakefield 12	10600 Common Oaks Drive, Raleigh, NC	R	American Gangster	2 hrs mins
		PG	August Rush	1 hr 5 mins
		PG	Bee Movie	1 hr 3 mins
		PG-13	Dan in Real Life	1 hr 3 mins
		PG	Enchanted	1 hr 4 mins
		PG	Fred Claus	1 hr 5 mins

Web Images Video News **Maps** Gmail more ▾

e.g., "10 market st, san francisco" or "hotels near lax"

10600 Common Oaks Drive, Raleigh, NC

Search Maps

Search the map | Find businesses | Get directions

Search Results | My Maps

10600 Common Oaks Dr
Raleigh, NC 27614
Make this my default location

Address:

10600 Common Oaks Dr
Raleigh, NC 27614

Get directions: To here - From here
Search nearby - Save to My Maps
Edit New!

Summary

- EGL and RDi-SOA
  - Easy transition for RPG and COBOL developers
  - Best Web 2.0 tooling on the market today
  - Evolve RPG and COBOL assets into Web Applications
  - UI Enrichment opportunities abound
  - Low risk, Quick ROI
  - Flexible deployment targets
  - Highly productive and flexible language and environment
  - Migrate at your own rate and pace: interactive programs only, complete applications, or entire systems

- Next Steps
  - Run a pilot
  - Contact Your IBM Reps

IBM EGL useful links

- Rational Developer for i for SOA Construction
  - http://www-01.ibm.com/software/awdtools/developer/rdisoa/
- RDi SOA – Trails and Installation
  - http://www-949.ibm.com/software/rational/cafe/docs/DOC-3128
- History of EGL
  - http://www-949.ibm.com/software/rational/cafe/docs/DOC-3161
- EGL Trial Download
  - http://www-949.ibm.com/software/rational/cafe/docs/DOC-1384
- RBD (Rational Business Developer)
  - http://www.ibm.com/developerworks/rational/products/rbde/
  - http://www-142.ibm.com/software/products/tw/zh/busdeveloper