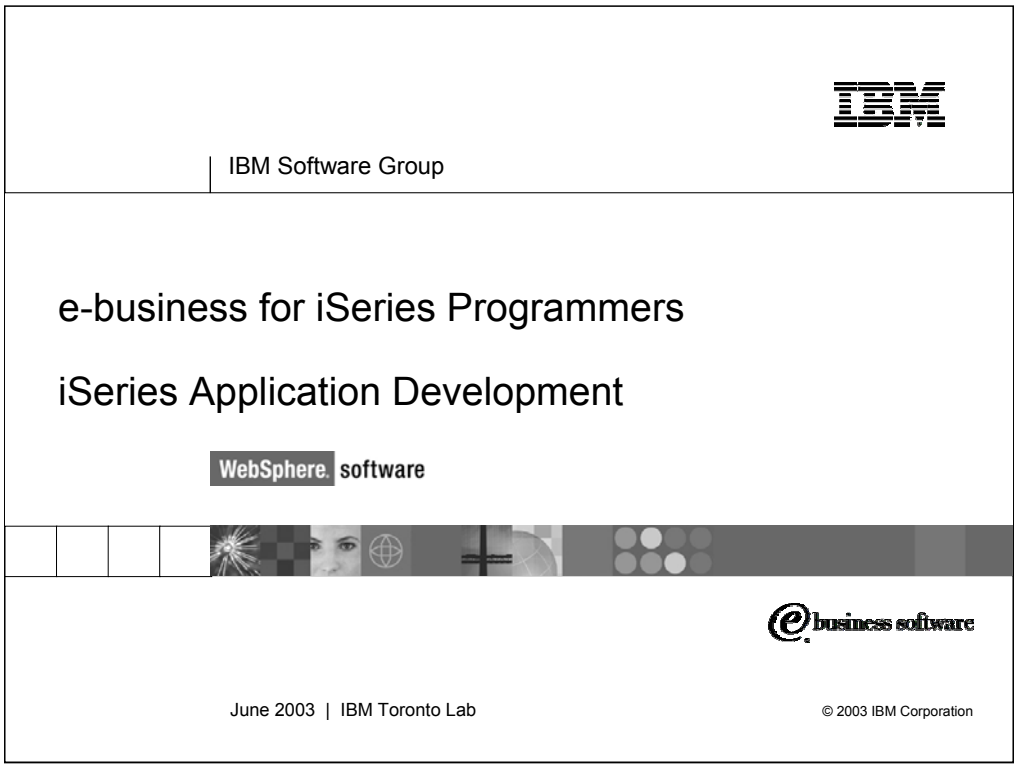


This presentation reviews how to move your iSeries applications to the Web.

Table of contents

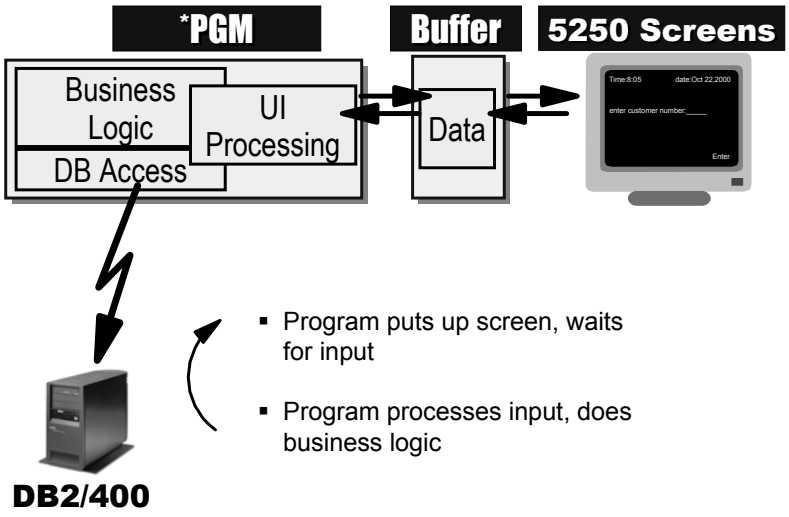
This presentation first gives you an overview of the architecture of an e-business application. It then provides more details on the technology: DHTML, Java Beans, JSPs, and Servlets. Next we cover how to code an e-business application. An example is shown to illustrate. The last topic covers tips to easily create an e-business application.

Architecture

- Today's Model
- eBusiness Architecture
- Model View Controller
- eBusiness Tiers
- Accessing Data
- Introducing Struts
- Tiers with Struts

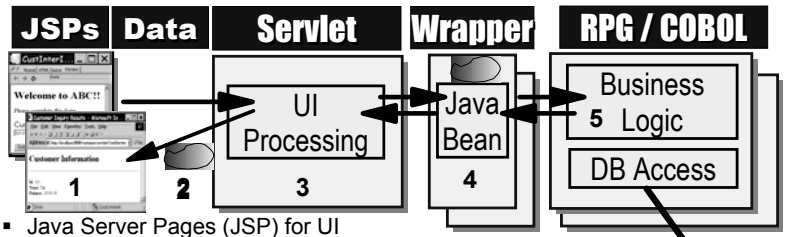
This section covers the e-business architecture topics.

Today's Model



- Program puts up screen, waits for input
- Program processes input, does business logic

eBusiness Architecture



DB2/400

1 ▪ Java Server Pages (JSP) for UI
- HTML
  - Usual tags for constant parts, FORM tag containing input fields
  - JavaScript for client-side event processing
  - Cascading Style Sheet (CSS) usage for attributes like font, color
- JSP tags for dynamically-substituted data
  - Data is extracted from a Java Bean passed from the servlet

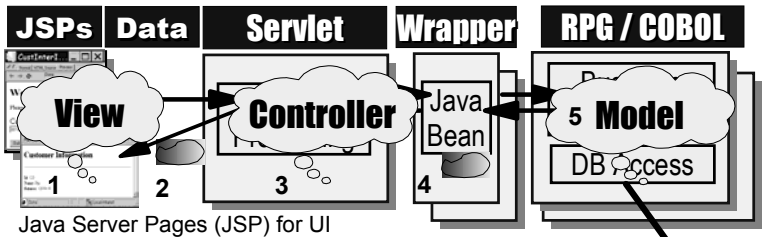
2 ▪ Java Bean containing application data to a JSP

3 ▪ Single Servlet for UI processing, flow control

4 ▪ Java Bean encapsulation of business logic

5 ▪ Business Logic: *PGM or ILE procedures

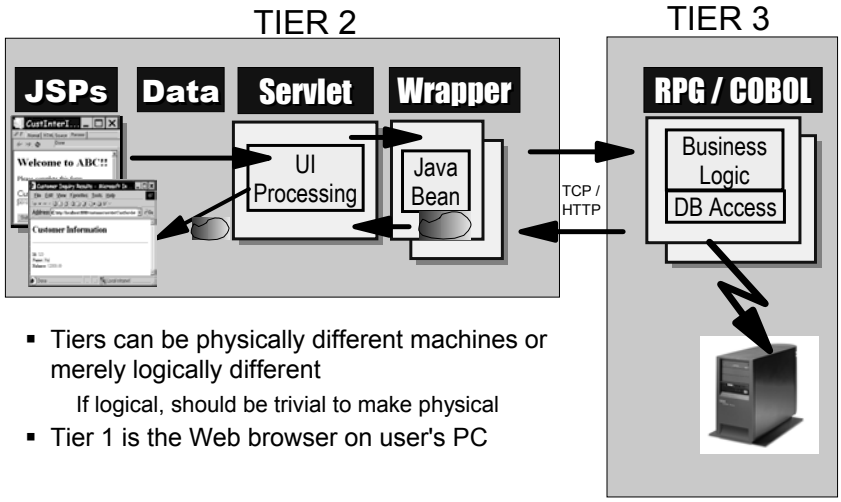
Model View Controller



1. Java Server Pages (JSP) for UI
   - HTML
     - Usual tags for constant parts, FORM tag containing input fields
     - JavaScript for client-side event processing
     - Cascading Style Sheet (CSS) usage for attributes like font, color
   - JSP tags for dynamically-substituted data
     - Data is extracted from a Java Bean passed from the servlet
2. Java Bean containing application data to a JSP
3. Single Servlet for UI processing, flow control
4. Java Bean encapsulation of business logic
5. Business Logic: *PGM or ILE procedures

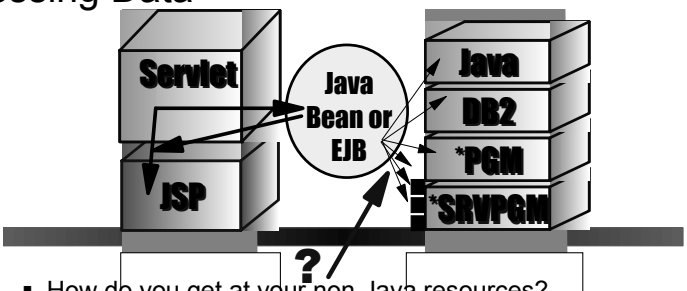
DB2/400

eBusiness Tiers



- Tiers can be physically different machines or merely logically different
  - If logical, should be trivial to make physical
- Tier 1 is the Web browser on user's PC

Accessing Data



- How do you get at your non-Java resources?
  - iSeries Toolbox for Java!
    - Call programs or service programs
      - Using ProgramCallMarkupLanguage (PCML)
    - Run OS/400 commands
    - Read/write data queues / areas
    - Access DB2/400 via JDBC or direct access
    - Much more!

Introducing Struts

- Struts is an open-source standard
  - http://jakarta.apache.org/struts/index.html
  - A framework for Web apps
  - Enforces MVC structure
  - A starting point versus blank slate

Tiers with Struts

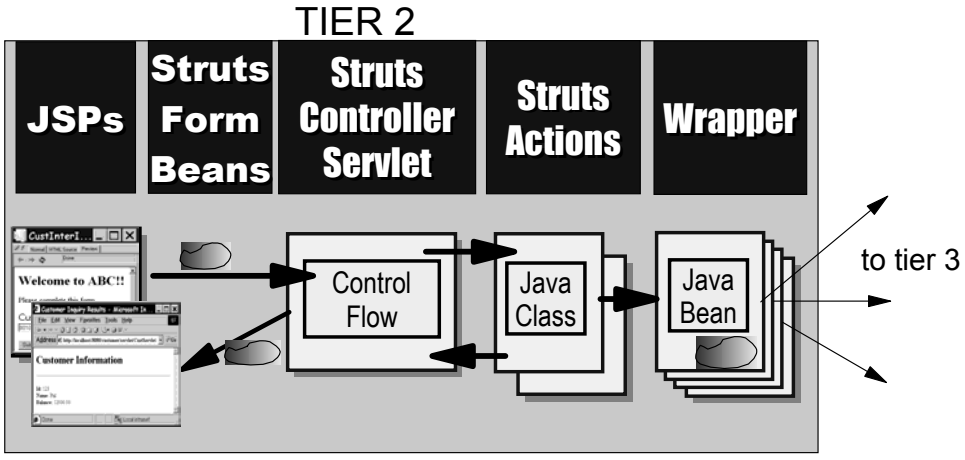


Table of contents

Now we know the e-business architecture, let’s look at the technology of an e-business application.

Details

- DHTML Details
  - HTML
  - CSS
  - JavaScript
  - DOM
- JavaBeans
- Servlets
- JSPs

Details

DHTML Details

First let’s look at DHTML.

What is DHTML?

- DHTML is "Dynamic HTML", which refers to:
  - HTML 4.0 or higher
  - Cascading Style Sheets
    - for defining visual attributes for HTML tags in a re-usable and cascading way
  - JavaScript
    - for client-side event handling
    - also known as ECMA-Script
  - Document Object Model (DOM)
    - for access to all Web page components via JavaScript
- http://w3.org

Details

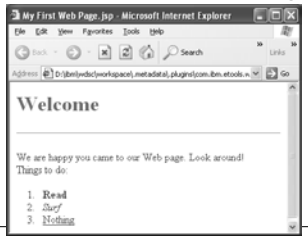
DHTML Details
HTML

Next lets look at HTML

Typical HTML / JSP file template

Example

```
<!doctype HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
    <title>ACME Home Page</title>
  </head>
  <body>
    <h1>Welcome</h1>                          heading one
    <hr>                                      horizontal rule
    <p>We are happy you came to our Web page. Look around!   paragraph
    <br>Things to do:                         line-break
      <ol>                                    ordered list
        <li><b>Read</b></li>                  list item  bold
        <li><i>Surf</i></li>                  list item  italic
        <li><u>Nothing</u></li>               list item  underline
      </ol>
  </body>
</html>
```



Text tags

Tag	Description
	Emphasis
	Stronger emphasis
<dfn></dfn>	Defining instance of term
<code></code>	Sample of computer programming code
<samp></samp>	Sample output
<kbd></kbd>	Text to be entered by user
<var></var>	Instance of a programming variable
<pre></pre>	Pre-formatted text. White space is honored
<cite></cite>	Citation
<abbr></abbr>	Abbreviation
<acronym></acronym>	Acronym (we at IBM need this one!)
<blockquote></blockquote>	Quotation, long
<q></q>	Quotation, short
	Subscript
	Superscript
<ins></ins>	Inserted text in a review markup
<del></del>	Deleted text in a review markup

Table tags

```
<table border=1.0>          start of table
  <caption>Item Inventory</caption>          table caption
  <tr>          table row
      <th>Item</th>          table heading
      <th>Description</th>
      <th>In stock</th>
  </tr>          end of table row
  <tr>
      <td>0001</td>          table data
      <td>Blue widget</td>
      <td>120</td> </tr>
  <tr> <td>0002</td> <td>Red widget</td> <td>1000</td> </tr>
  <tr> <td>0003</td> <td>White widget</td> <td>0</td> </tr>
</table>          end of table
```

Link tags

where to go

what to show

anchor tag

```
<a href="http://www.ibm.com">Click here to visit IBM<a>
```

open new window

```
<a href="http://www.ibm.com" target="awindow">Click here<a>
```

```
<h1 id="target1">This is target1</h1>
... or ...
<a name="target1">This is target1<a>
. . .
<a href="#target1">This is target1<a>
```

how to tag a "target"

how to link to a target

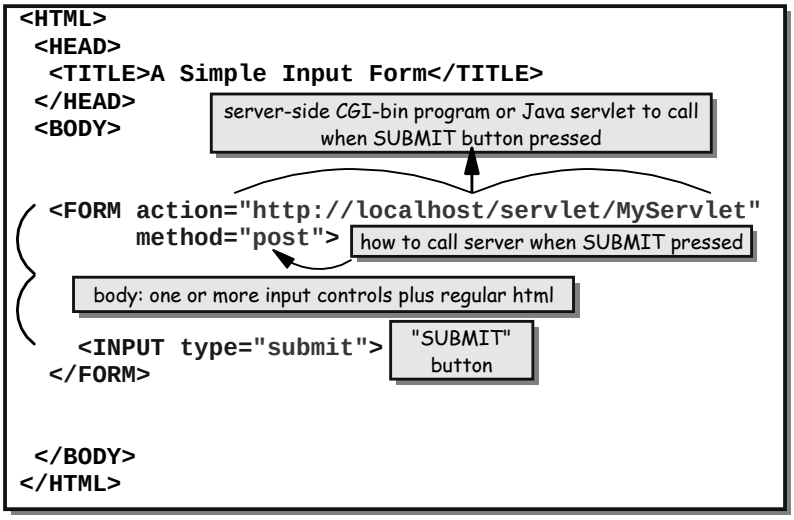
Frame tags

```
<frameset rows="35%, 65%">
   <frame src="logo.gif">
   <frameset cols="20%, 80%">
      <frame src="undercon.gif">
      <frame src="acme.html">
   </frameset>
   <noframes>
      <a href="acme.html">No-frames page</a>
   </noframes>
</frameset>
```

HTML Forms

- Allows you to solicit user input
- Can build HTML document that contains:
  - checkboxes
  - radio buttons
  - push buttons
  - entry fields
  - selectable lists
  - ... others
- Collect data from user
  - send to server program
    - The FORM tag specifies the name of the program
      - Historically a CGI-bin program
      - New option is a Java Servlet

Anatomy of a FORM



Example of a FORM

```
<FORM action="http://localhost/servlet/MyServlet"
      method="post">
  Name <INPUT type="text" name="name"><BR>
  Age <INPUT type="text" name="age" size="3" maxlength="3"><BR>
  Country <SELECT name="country">
            <OPTION selected>Canada</OPTION>
            <OPTION>Mexico</OPTION>
            <OPTION>United States</OPTION>
          </SELECT> <BR>
  e-mail <INPUT size="30" type="text" name="email"><BR>
  <BR>
  <INPUT type="radio" name="sex" value="M" checked>Male
  <INPUT type="radio" name="sex" value="F">Female<BR><BR>
  <INPUT type="checkbox" name="mail" checked>e-mail me<BR><BR>

  <INPUT type="submit" value="Register">
  <INPUT type="reset"  value="Reset">
</FORM>
```

entry field

selectable list

entry field

radio buttons

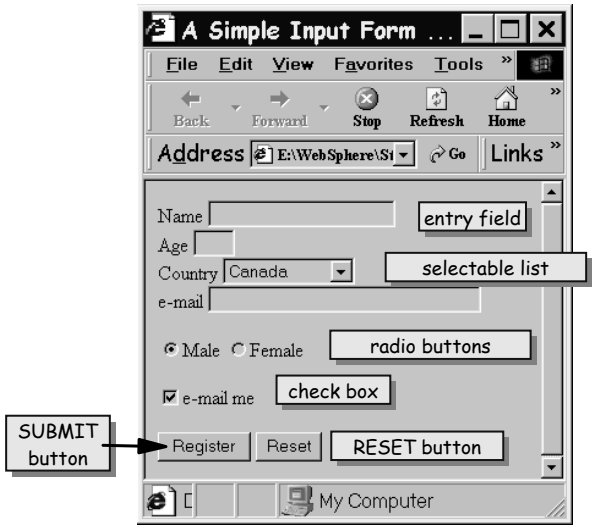
check box

SUBMIT button

RESET button

http://www.w3.org/TR/REC-html40/

Example of a FORM (continued)



Details

DHTML Details
CSS

Now we look at CSS

Cascading Style Sheets

```
<html>
 <head>
    <title>ACME Home Page</title>
    <style type="text/css">
      li { background-color: yellow }
    </style>
 </head>
 <body>
     <h1>Welcome</h1>
      <p>We are happy you came to our Web page. Look around!
      <br>Things to do:
         <ol>
           <li><b>Read</b></li>
           <li><i>Surf</i></li>
           <li><u>Nothing</u></li>
         </ol>
 </body>
</html>
```

makes *all* tags have a yellow background

Cascading Style Sheets (continued)

```
<html>
 <head>
    <title>ACME Home Page</title>
    <style type="text/css">
      .xxx { background-color: yellow }
    </style>
 </head>
 <body>
     <h1>Welcome</h1>
      <p>We are happy you came to our Web page. Look around!
      <br>Things to do:
          <ol>
            <li class="xxx"><b>Read</b></li>
            <li><i>Surf</i></li>
            <li><u>Nothing</u></li>
          </ol>
 </body>
</html>
```

makes only *some* tags have a yellow background

Property – Value pairs

Property	Value
Background-color	Aqua. black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, transparent, white, yellow or RGB color in the form #rgb, or #rrggbb, or rgb(r%,g%,b%)
Background-image	Url("uri") or none
Background-repeat	Repeat, repeat-x, repeat-y or no-repeat
Border-width	Thin, medium, thick, +-number.decimal
Border-style	None, hidden, dotted, dashed, solid, double, groove, ridge, inset, outset
Border-spacing	+-number.decimal<um>
Border-color	Aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, transparent, white, yellow or RGB color in the form #rgb, or #rrggbb, or rgb(r%,g%,b%)
Color	Foreground text color. Same as background-color
Margin-top	+-number.decimal<um> or +-number%
Margin-right	+-number.decimal<um> or +-number%
Margin-bottom	+-number.decimal<um>or +-number%
Margin-left	+-number.decimal<um> or +-number%
Font-family	Explicit as in: "courier" or "helvatica" or generic as in: "serif", "sans-serif", "cursive", "fantasy" and "monospace"
Font-size	Small, medium, large, smaller, larger or an actual number or percentage
Font-stye	Normal, italic, oblique
List-style-image	Ulr("Uri")
List-style-position	Inside or outside
List-style-type	Disc, circle, square, decimal, decimal-leading-zero, lower-roman, upper-roman, lower-greek, lower-alpha, lower-latin, upper-alpha, upper-latin, Hebrew, Armenian, Georgian, cjk-ideograhic, hiragana, katakana, hiragana-iroha, katakana-iroha or none
Text-align	Left, right, center, justify
Text-transform	Capitalize, uppercase, lowercase, none
Text-decoration	None underline overline line-through blink (multiple values allowed for this property, space delimited)
Text-shadow	<color> <length length length>

Not a complete list

Example with embedded style sheet

```
<html>
 <head>
    <title>ACME Home Page</title>
    <style type="text/css">
      h1 { background-color: yellow }
      h3 { background-color: blue; color: white; border-style: groove }
      em { color: red; font-style: italic }
      strong { font-size: larger; font-style: oblique }
      li { list-style-type: disc }
    </style>
 </head>
 <body>
     <h1>Welcome</h1>
      <hr>
      <p>We are <em>happy</em> you came
          to our Web page.<br>
      <strong>Look around!</strong>
      <h3>Things to do:</h3>
          <ol>
            <li><b>Read</b></li>
            <li><i>Surf</i></li>
            <li><u>Nothing</u></li>
          </ol>
 </body>
</html>
```

Example with external style sheet

```
<html>
 <head>
    <title>ACME Home Page</title>
    <link type="text/css" rel="stylesheet" href="acmecss.css">
 </head>
 <body>
    <h1>Welco
     <hr>
     <p>We are
          to our
     <strong>L
     <h3>Thing
        <ol>
          <li><
          <li><
          <li><u>Nothing</u></li>
        </ol>
 </body>
</html>
```

file "acme.css"

```
<style type="text/css">
   h1 { background-color: yellow }
   h3 { background-color: blue; color: white; border-style: groove }
   em { color: red; font-style: italic }
   strong { font-size: larger; font-style: oblique }
   li { list-style-type: disc }
</style>
```

Details

DHTML Details
JavaScript

Next we look at JavaScript

JavaScript

- JavaScript is not Java!
  - A loosely typed, interpreted language,
    - Runs in the Browser
    - Variables aren't declared, just used
  - Officially name is ECMAScript
    - Look for ECMA-262 at www.ecma-international.org
    - Look at www.javascriptsource.com for samples
    - Search google for "javascript" and "tutorial"

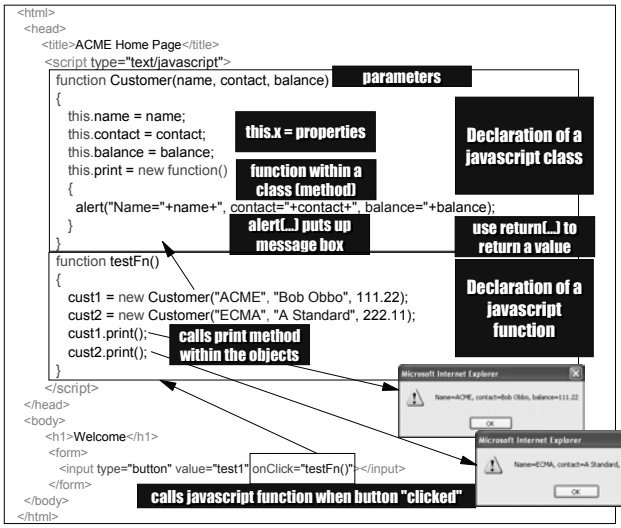
```
<script type="text/javascript">
…
</script>
```

How to imbed javascript logic. Place in <head> section

```
<script type="text/javascript" src="/scripts/common.js">
…
</script>
```

How to include javascript from a javascript file (.js extension)

JavaScript Example



JavaScript Syntax

```
if (day == "sat")
  price = 20.0;
else if (day == "sun")
  price = 5.0;
else
  price = 10.0;
```

```
if (arg1 && arg2)
  if (arg2 > arg1)
    larger = arg2;
```

```
var anArray = new Array(5);
for (var idx = 0; idx < anArray.length; idx++)
{
   anArray[idx] = idx;
   alert("anArray value at index " + idx + ": " + anArray[idx]);
}
var anotherArray = new Array(0,1,2,3,4);
var dayArray(3);
anArray["sat"] = 20.0;
anArray["sun"] = 5.0;
anArray["mon"] = 10.0;
```

```
total = 0;
for (var idx=1; idx <= 10; idx++)
```

```
total = 0;
var idx = 1;
while (idx <= 10)
{
  total = total * idx;
  idx++;
}
```

```
total = 0;
var idx = 1;
do
{
   total = total * idx;
   idx++;
} while (idx <= 10)
```

```
// a line comment
```

```
/* a multi-line
   comment */
```

semi-colons are optional but recommended

"var" is optional keyword on first use of variable

javascript is case sensitive

Reference

DataType	Values
boolean	true and false
numeric	signed/unsigned numbers: integers, decimals, exponentials, hexadecimals octals
string	single or double quoted text. Escaped characters like \n also supported
function	a variable can be assigned to a function. Using the variable calls the function
object	a variable can be assigned to an object created by using new XXX(...) on a class. The properties and methods in the object can be accessed/run by x.yyy or x.zz()

operators	Values
==, !=, &&, ||, >, <, >=, <=, !	comparators
?:	conditional ternary: isMale = (sex == male) ? true : false;
=	assignment
&, |	bitwise masking
++, --	increment, decrement
+, -, /, *, %	binary math
+,-	unary math
new	instantiate an object
instanceof	test datatype of variable or object

Built-in Objects	Description of methods and properties
Array	arrays stuff, such as concat(), join()
Boolean	boolean stuff, such as valueOf()
Date	date stuff such as set/getMonth()
Function	for functions, such as caller property
Global	callable without qualification
Math	abs(), max(), sin(), random(), etc
Number	numeric stuff like MAX_VALUE
Object	for objects, such as constructor
RegExp	for regular expressions
String	concat(), indexOf(), substr(), etc

Built-in Methods	Description
alert(string)	displays a message box showing the string, with an OK button
confirm(string)	displays a message box showing the string, with Yes/No buttons. Returns 0 for yes, 1 for no
prompt(prompt-string, default-input-string)	displays a dialog with a prompt string and an entry field. Returns contents of entry field

Events

Event	Description
onBlur	When focus leaves a document element
onClick	When user clicks left mouse button, anywhere in document
onContextMenu	When user clicks right mouse button to show a context menu, anywhere in document
onDblclick	When user double-clicks anywhere in document
onFocus	When document element gains focus
onHelp	When user presses F1 or Help
onKeydown	When user presses a key (on the way down)
onKeypress	When user presses an alphanumeric key (when fully pressed)
onKeyup	When user releases a key
onLoad	When object is loaded. Typically use on <body> tag
onMousedown	When user clicks the document with either mouse button
onMousemove	When user moves the mouse over the document
onMouseout	When user moves the mouse out of the document
onMouseover	When user moves the mouse into the document
onMouseup	When user releases a mouse button
onUnload	When user exits frame or window.
onStop	When user clicks the stop button or leaves the Web page
onSubmit	When user submits form

not a complete list

Details

DHTML Details
DOM

Now we look at DOM

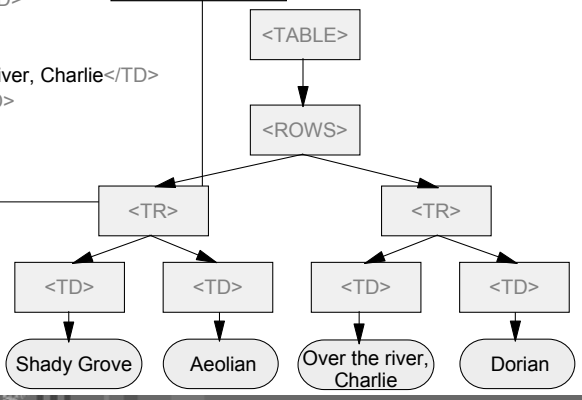
Document Object Model

- To make JavaScript interesting...
  - It needs to be able to access everything in a Web page
    - To be able to read the contents of entry fields and selections of check boxes, for example
    - Indeed, to access anything and everything in the page
    - For both reading of content, and for writing of contents and CSS attributes
- The Document Object Model gives us this
  - The entire contents of the Browser, and the browser itself, is available as objects in a tree
- Web Browsers add their own objects too
  - Like "window" for the current window. Use window.document to access the DOM

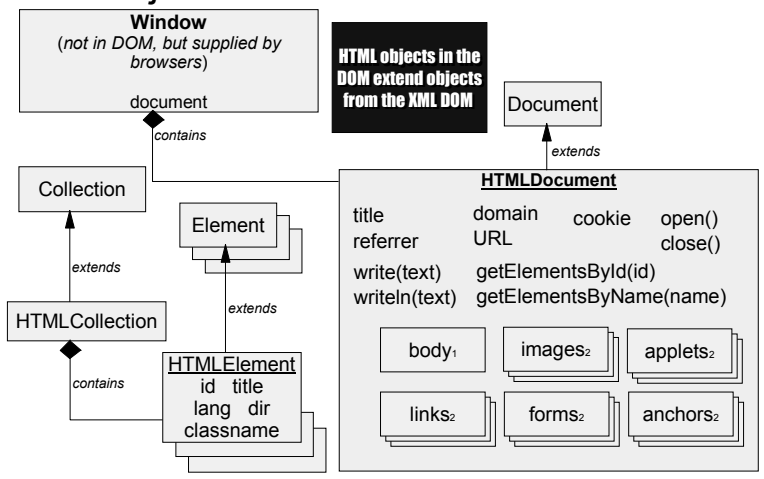
Document Object Model

```
<TABLE>
    <ROWS>
      <TR>
          <TD>Shady Grove</TD>
          <TD>Aeolian</TD>
      </TR>
      <TR>
          <TD>Over the River, Charlie</TD>
          <TD>Dorian</TD>
      </TR>
    </ROWS>
</TABLE>
```

Example: HTML like this becomes accessible in a tree like this:

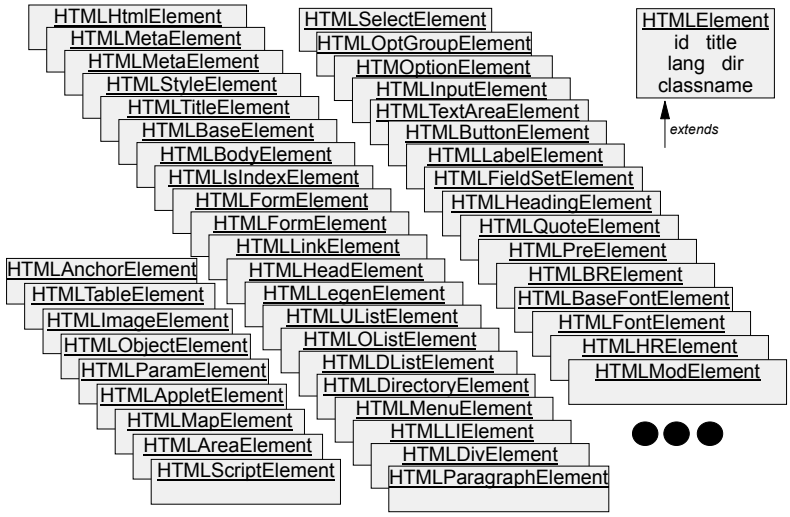


Document Object Model



1 of type HTMLElement
2 of type HTMLCollection, which is a list of HTMLElements

Document Object Model



Document Object Model

```
<html>
  <head>
     <title>ACME Home Page</title>
     <script type="text/javascript">
       function OpenNewWindow()
       {
          contents="<h1>Hello World!</h1>";
          contents+="<form><p><input type='submit' value='Close' ";
          contents+="ONCLICK='window.close(); return false;'></P></FORM>";
          Win=window.open('', 'Window', 'resizable,height=100,width=210');
          Win.document.writeln(contents);
       }
     </script>
  </head>
  <body>
     <h1>Welcome</h1>
     <form>
        <input type="button" value="Open Window" onClick="OpenNewWindow()"></input>
     </form>
  </body>
</html>
```

Common Uses

- Common uses of JavaScript and DOM:
  - Rollover ... change image as mouse passes
  - Client-side error checking of input fields
  - Mapping multiple buttons to "Submit"
  - Formatting text before displaying it
  - Masking text as it is entered
  - Page load and unload events
  - Making images, etc clickable
  - Opening new browser windows
  - Dynamically affecting page content
  - Calculators, date pickers, etc
  - Intercepting back/forward buttons

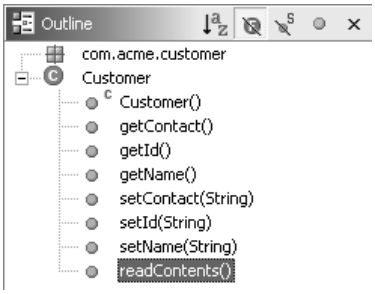
Details

Java Beans

Next we review Java Beans

Java Beans

- Java Beans characteristics:
  - Default constructor (no parameters)
  - Properties. Fields with
    - getXXX() methods for reading,
      setXXX(xx) methods for writing
  - Methods
    - For doing interesting things
  - Events
    - Listeners register their interest in a bean's event
    - The bean "fires" the event by calling the registered listeners
    - Event mechanism not typically used in server-side code
- Java Beans are not EJBs
  - They are really just fancy data structures



Details

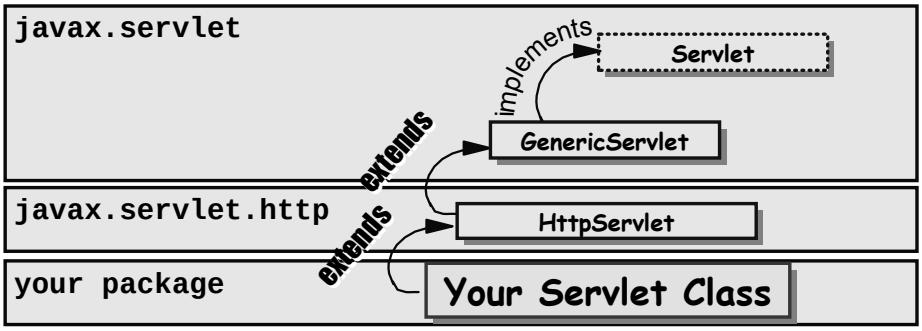
Servlets

Now we review Servlets.

What are Servlets?

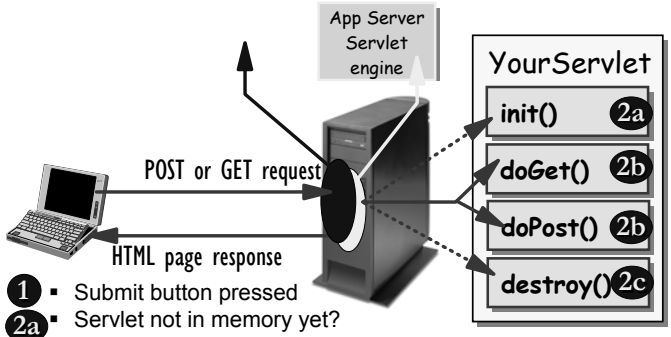
- Servlets are . . .
  - Java classes (programs written in Java)
- Servlets run . . .
  - On the second tier
  - Inside an Application Server
    - Which is a plug-in to an HTTP Server
  - When a user goes to your Web page
- The input to Servlets are . . .
  - User-entered data from a Web page
- The output of a Servlet is . . .
  - Java Bean, passed to a JavaServer Page

Servlet Classes



- javax.servlet package
  - classes to support generic, protocol-independent servlets
- javax.servlet.http package
  - adds HTTP-specific functionality

How Servlets Work



1. Submit button pressed
2a. Servlet not in memory yet?
   init method called (once, until destroy called)
2b. Either doGet or doPost called
3. New HTML page returned to user (usually via JSP)
2c. When Application Server stopped, destroy called

Servlet Methods

- The doGet/doPost methods accept two parameters:
  - Request object: Tells the servlet about the request
  - Response object: It is used to return a response
- Either doPost or doGet will be called for you
  - Just code one, and code the other to call it!
- Optionally code these methods:
  - init
    - Only called once, when servlet brought into memory
  - destroy
    - Only called once, when servlet removed from memory

My First Servlet

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class MyFirstServlet extends HttpServlet
{
  public void init()
  {
  }
  public void doPost(HttpServletRequest  request,
                     HttpServletResponse response)
         throws ServletException, IOException
  {
      // your code
  }
  public void doGet(HttpServletRequest  request,
                    HttpServletResponse response)
         throws ServletException, IOException
  {
      doPost(request, response);
  }
  public void destroy()
  {
  }
}
```

- **Typical Servlet class shell**

just call doPost from doGet

Details

JSPs

Next we review JSPs

What are Java Server Pages

- JSPs are . . .
  - .jsp files containing html tags plus JSP tags
- JSP tags . . .
  - Allow dynamic data to be inserted into the static HTML
    - Data is extracted from Java beans that are passed to the JSP
- JSPs are called . . .
  - By your servlet
- The input to JSPs are . . .
  - Java Beans passed from your Servlet
- The output of a JSP is . . .
  - A full Web page, displayed to user

JSP Tags

- JSPs contain HTML and embedded JSP-tags:
  - Tags for embedding Java directly into the html
    - Eg, processing to determine what HTML to emit
  - Tags for embedding Java beans directly into the html
    - Declare a bean
      - Have JSP engine instantiate it, or pass it in from a servlet
    - Emit HTML to display data from a bean
  - Tags for other things
    - Like including other JSP files

JSP Important Tags

- Most-often used JSP tags:
  - jsp:useBean
    - For declaring a bean passed in from Servlet
      - <jsp:useBean id="custbean" class="cust.CustBean" scope="session"/>
      - id= name used throughout JSP
      - class=fully qualified class name (package.class)
      - scope="session" for beans scoped to this user's session
  - jsp:getProperty
    - For extracting data from a declared bean, by calling its getXXX methods
    - <jsp:getProperty name="custbean" property="balance"/>
      - name=name from id parameter of jsp:useBean tag
      - property=method name, minus the "get" part

My First JSP

```
<jsp:useBean id="abean"
             class="apackage.TheBeanClass"
             scope="session">
</jsp:useBean>

<html>
 <head>
   <title>My Title</title>
 </head>

<body>
 <h1>Customer Information</h1>
 <hr>
 <br><b>Id:</b>
   <jsp:getProperty
     name="abean" property="aProperty"/>
</body>

</html>
```

- **Typical JSP shell**

1. Declare bean(s)
2. Usual HTML tags for static parts
3. Insert jsp:getProperty where dynamic data is to be inserted

Use Beans for Dynamic Data

- To use a bean for servlet-JSP communication:
  - Instantiate bean in servlet
  - Set input parameters via setXXX calls
  - Call method to "run" bean (by convention, method is execute())
  - Pass bean to JSP by placing it in appropriate servlet runtime object using servlet APIs
  - Pass control to the JSP
  - In the JSP, declare the bean and extract out its data to emit into HTML output

Bean Scope

- First decision is bean lifetime or scope

Scope	Description
page	Lives only for this page and this request
request	Lives for all pages involved in this request (ie, if **<jsp:forward>** used)
session	Lives for all pages involved in any request by same client
application	Lives for all pages involved in any request by any client for this Web Application

- This decision affects:
  - How to declare <jsp:useBean> tag
  - How to pass bean from servlet to JSP

Passing Bean From Servlet

- How to pass bean from servlet to JSP:

Scope	Description
page	Not applicable. Can only be instantiated in JSP.
request	request.setAttribute("mydatabean", dataBean);
session	HttpSession session = request.getSession(true); session.putValue("mydatabean", dataBean);
application	ServletContext context = getServletContext(); context.setAttribute("mydatabean", dataBean);

- To actually transfer control to JSP:

```
RequestDispatcher dispatch = getServletContext().
                getRequestDispatcher("/jsp/HelloJSP.jsp");
dispatch.forward(request, response);
```

Declaring Bean in JSP

- How to declare bean in JSP:

Scope	Description
page	<jsp:useBean id="mydatabean" class=pkg.DataBean scope="page"/>
request	<jsp:useBean id="mydatabean" class=pkg.DataBean scope="request"/>
session	<jsp:useBean id="mydatabean" class=pkg.DataBean scope="session"/>
application	<jsp:useBean id="mydatabean" class=pkg.DataBean scope="application"/>

- Tip: Beans must be in a named package or you will get "class not found" error!

Table of contents

Now we know the technology details, we are ready to see how to code an e-business application.

How to Code an eBusiness Application

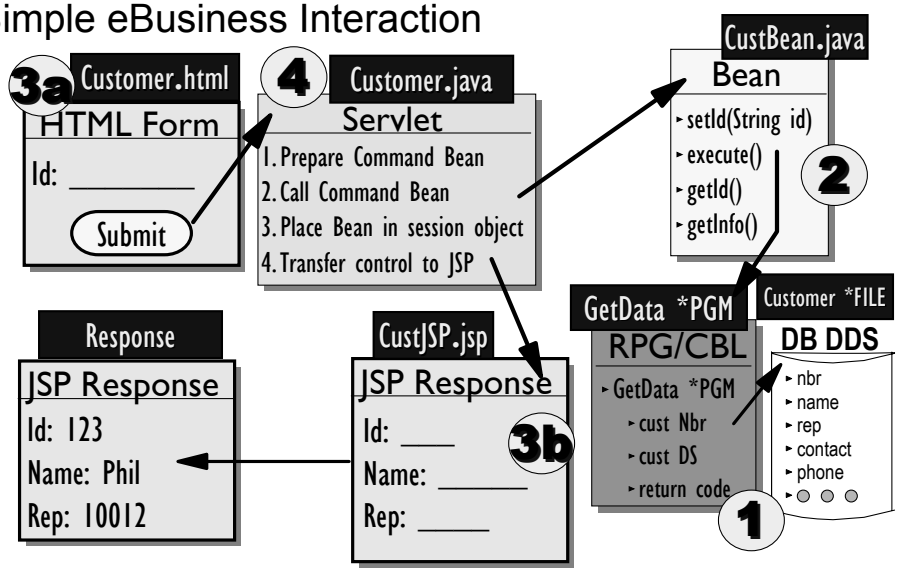
- Overview
- Simple eBusiness Interaction
- Create / Leverage the Business Logic
- Business logic accessible from Java
- Java to RPG
- Create input and output JSPs
- Create the Servlet

These are the topics we will cover.

Overview

1. Create or leverage the business logic
2. Make business logic accessible from Java
   - Encapsulate input/output parameters in Java bean
   - Supply execute method to call business logic
   - Tip: Use PCML from AS/400 Toolbox for Java
3. Create the input and output JavaServer Pages
   - Use HTML tags for constant part
   - Use HTML forms for input fields (.html file ok, vs .jsp)
   - Use JSP tags to subsitute dynamic data from your Bean
4. Create the Servlet
   - Reads user input
     - Sets input data in wrapper bean
   - Calls execute method of wrapper bean
     - Results in output variables being set
   - Decides on JSP to display next, displays it

Simple eBusiness Interaction



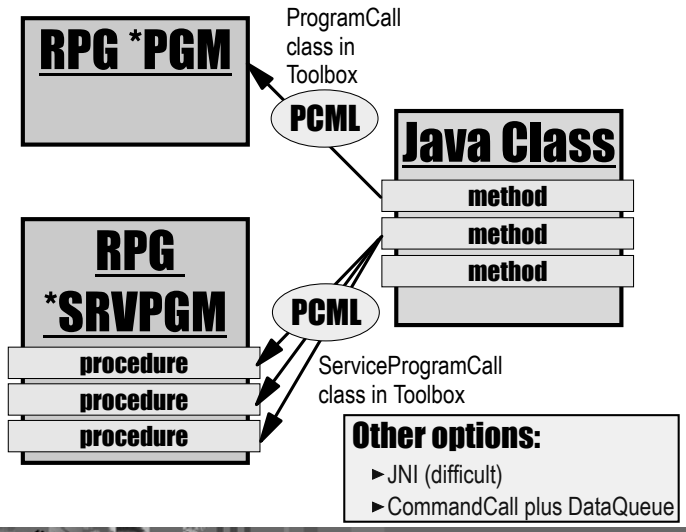
1. Create / Leverage the Business Logic

- Requires callable non-interactive RPG or COBOL logic
- Either *PGM object, or *SRVPGM object with ILE Procedures inside it
- No tool exists to help extract business logic from interactive applications
  - Roll up your sleeves, dig in
  - Use procedures in RPG IV
  - Use ILE COBOL
  - Suggestion: hire consultant whose done this before
  - Sometimes: easier to start from scratch, copy and paste snippets of logic
    - Remote System Explorer in WDSC makes effective tool for this

2. Make Business Logic Accessible from Java

- Requires use of iSeries Toolbox for Java
  - Use ProgramCall to call *PGM objects
  - Use ServiceProgramCall to call *SRVPGM objects
- Use Program Call Markup Language (PCML) to make it easier
  - Simple tag based language to describe pgm/srvpgm call
  - Compiles into ProgramCall / ServiceProgramCall code

Java to RPG



3. Create the input and output Java Server Pages

- Create .html or .jsp file containing html FORM prompting for user input
- Create .jsp file to display output
  - Declare bean passed from Servlet using <jsp:useBean>
  - Extract data from bean using <jsp:getProperty>

4. Create the Servlet

- Read user input
  - Use request.getParameter to read user input
- Instantiate bean that calls RPG / COBOL
- Set input data
- Call execute method of wrapper bean
  - Results in output variables being set
- Decide on JSP to display next, call it
  - Use appropriate servlet API

Table of contents

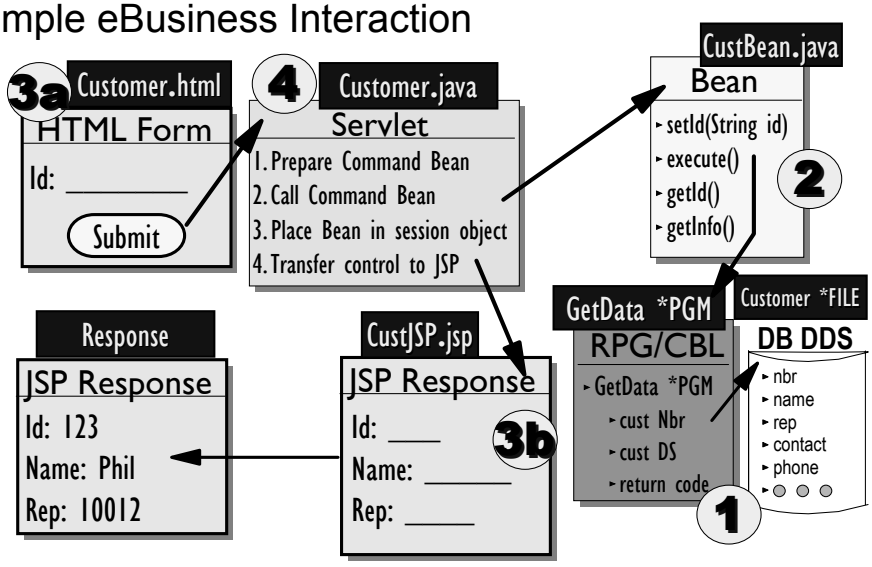
Now lets look at an example e-business application.

Example of an eBusiness Application

- Simple eBusiness interaction
- RPG IV Program
- PCML For *PGM Call
- Input Page
- Output JSP
- Servlet
- Running the Example

Here are the topics we will cover.

Simple eBusiness Interaction



RPG IV Program

```
FCUSTOML3  IF   E           K Disk
D Custnoi         S                   like(CUSTNO)
D CSTRUC        E DS                  extname(customl3:custom01)
D return          S              20
D*------------------------------------------------------------------
C     *entry        plist
C                   parm                    custnoi
C                   parm                    cstruc
C                   parm                    return
C                   eval      return=*blank
C     custnoi       chain     customl3                           5050
C                   if        *in50
C                   eval      return='CUS0001 ' + CUSTNOI
C                   else
C                   eval      return='0'
C                   endif
C                   return
```

input

output

Program: GetData

RPG IV Program – DB Files Used

File: Customl3

```
A* Logical file description
A          R CUSTOM01                  PFILE(CUSTOMER)
A          K CUSTNO
```

```
A* Physical file description - CUSTOMER FILE
A      R CUSTOM01
A        CUSTNO         7            COLHDG('Customer number')
A        CUSTNA        40            COLHDG('Company name')
A        REPNO          5            COLHDG('Rep identifier')
A        CONTAC        30            COLHDG('Name')
A        CPHONE        17            COLHDG('Telephone')
A        CFAX          17            COLHDG('Fax')
A        CADDR         40            COLHDG('Address')
A        CCITY         30            COLHDG('City')
A        CCOUNT        20            COLHDG('Country')
A        CZIP          10            COLHDG('Postal Code')
A        CZIPLO         1            COLHDG('PC location')
A                                    VALUES('1' '2' '3')
A      K CUSTNO
```

File: Customer

PCM for *PGM Call

```
<pcml version="2.0">
 <struct name="custStruct" usage="inputoutput">
  <data name="CUSTNO" length="7" usage="inputoutput" type="char"/>
  <data name="CUSTNA" length="40" usage="inputoutput" type="char"/>
  <data name="REPNO" length="5" usage="inputoutput" type="char"/>
  <data name="CONTAC" length="30" usage="inputoutput" type="char"/>
  <data name="CPHONE" length="17" usage="inputoutput" type="char"/>
  <data name="CFAX" length="17" usage="inputoutput" type="char"/>
  <data name="CADDR" length="40" usage="inputoutput" type="char"/>
  <data name="CCITY" length="30" usage="inputoutput" type="char"/>
  <data name="CCOUNT" length="20" usage="inputoutput" type="char"/>
  <data name="CZIP" length="10" usage="inputoutput" type="char"/>
  <data name="CZIPLO" length="1" usage="inputoutput" type="char"/>
 </struct>

 <program name="CustInterPGM"
          path="/QSYS.LIB/coulthar.LIB/getdata.PGM">
  <data name="CUSTNO" length="7" usage="inputoutput" type="char"/>
  <data name="CustInfo" usage="output" type="struct"
        struct="custStruct"/>
  <data name="ReturnValue" length="20" usage="output"
        type="char"/>
 </program>
</pcml>
```

File: GetCust.pcml

Java Bean for *PGM

```
package cust;
import java.math.*;

public class CustBean
{
    private ProgramCallDocument pcml = null;
    public CustBean()
    {
      pcml = new ProgramCallDocument(as400System, "GETCUST");
    }
    public boolean execute()
    {
      pcml.callProgram("CustInterPGM");
    }
    public String getCustNumber()
    {
      return (String)pcml.getValue("CustInterPGM.CUSTNO");
    }
    public void setCustNumber( String CUSTNO )
    {
      pcml.setValue( "CustInterPGM.CUSTNO", CUSTNO );
    }
    public String getCustName()
    {
       return (String)pcml.getValue("CustInterPGM.CustInfo.CUSTNA");
    }
    public String getRepNo()
    {
       return (String)getValue("CustInterPGM.CustInfo.REPNO");
    }
     ○ ○ ○
    public String getReturnValue()
    {
      return (String)pcml.getValue("CustInterPGM.ReturnValue");
    }
}
```

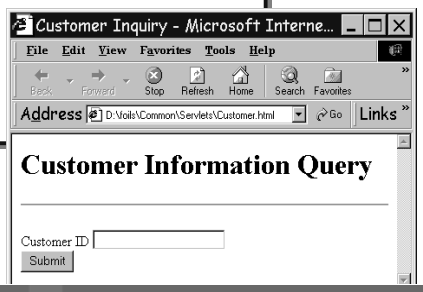
Class: CustBean.java
Package: cust

Input Page

Customer.html

```
<HTML>
<TITLE>Customer Inquiry</TITLE>
<BODY>
<h1>Customer Information Query</h1>
<hr>
<FORM METHOD=POST ACTION=
  "http://localhost:8080/customer/servlet/Customer">
  <p>Customer ID
  <INPUT TYPE="text" NAME="id">
  <br>
  <INPUT TYPE="submit"
         VALUE="Submit">
</FORM>
</BODY>
</HTML>
```

Remember this, it is used in servlet's **getRequest** call to retrieve contents of this field



Output JSP

CustJSP.jsp

```
<jsp:useBean id="custbean"
             class="cust.CustBean"
             scope="session">
</jsp:useBean>
<html>
 <head>
   <title>Customer Inquiry Results</title>
 </head>
<body>
 <h1>Customer Information</h1>
 <hr>
 <br><b>Id: </b>
   <jsp:getProperty
     name="custbean" property="custNumber"/>
 <br><b>Name: </b>
   <jsp:getProperty
     name="custbean" property="custName"/>
 <br><b>Rep: </b>
   <jsp:getProperty
     name="custbean" property="repNo"/>
○ ○ ○
</body>
</html>
```

Call getCustNumber

Call getCustName

Call getRepNo

Servlet

Customer.java

```
public class Customer extends HttpServlet
{
   public void doPost(HttpServletRequest  request,
                      HttpServletResponse response)
          throws ServletException, IOException
   {
      cust.CustBean custBean = new cust.CustBean();
      custBean.setCustNumber( request.getParameter("id") );
      custBean.execute();
      // 1) Put Bean in current session object
      HttpSession session =
         request.getSession(true);
      session.putValue("custbean", custBean);
      // 2) Invoke JSP file
      RequestDispatcher dispatch =
        getServletContext().
        getRequestDispatcher("/jsp/CustJSP.jsp");
      dispatch.forward(request, response);
   }
   public void doGet(HttpServletRequest  request,
                     HttpServletResponse response)
          throws ServletException, IOException
   {
       doPost(request, response);
   }
}
```

Use our bean

Put bean in session

Call JSP

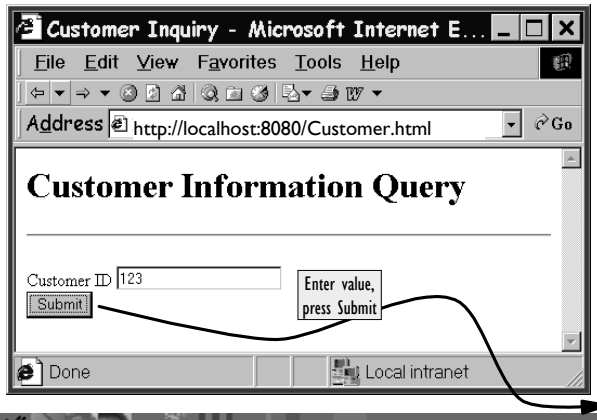
Retrieve contents of HTML form input field with **name="id"**

JSP in /jsp subdir

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
```

Running the Example

- In browser enter URL of input page:
  http://localhost:8080/Customer.html



Running the Example (continued)

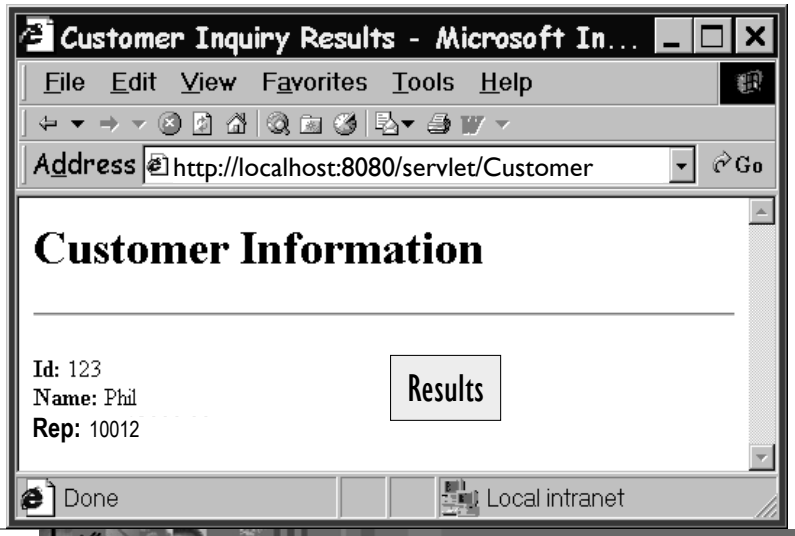


Table of contents

Finally lets look at some tips to make creating an e-business application easy.

Tips

- Tools to Help
- WebSphere
- Links

Here are the topics we will cover.

Tips

Tools to Help

First lets look at tools to help.

Tools to Help

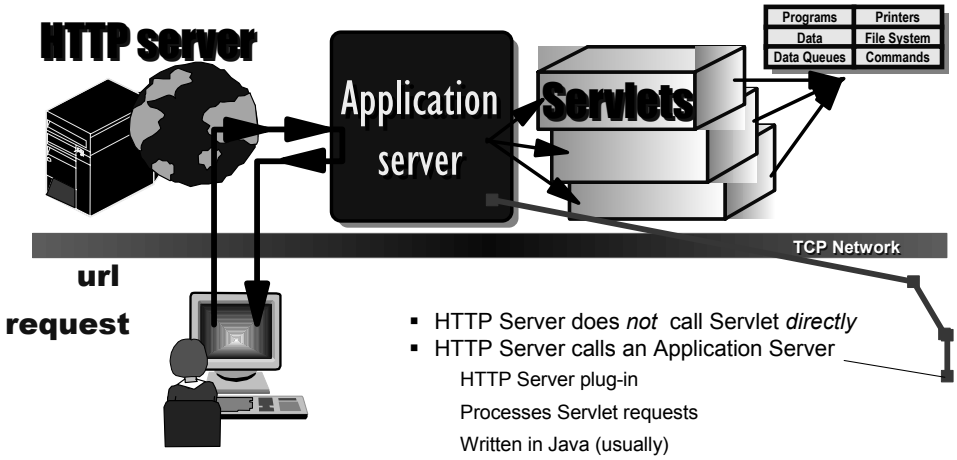
- WebSphere Development Studio Client
  - iSeries Web Tooling
    - For generating all plumbing, given a batch *PGM / proc
    - For WYSIWYG editing of HTML and JSPs
      - Just like SDA for DSPFs!
    - For easy publishing of files into WAS / 400
    - For locally testing servlets/JSPs before deploying
  - WebFacing
    - For converting display file source to JSPs and beans
    - For using converted JSPs/beans from *PGMs as is
  - iSeries Java Tooling
    - Awesome tools for writing the Java code
    - Program Call wizard to wrap *PGM/proc as a JavaBean

Tips

WebSphere

Now lets look at WebSphere Application Server.

Application Servers

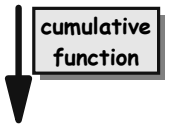


- HTTP Server does *not* call Servlet *directly*
- HTTP Server calls an Application Server
  - HTTP Server plug-in
  - Processes Servlet requests
  - Written in Java (usually)
  - Uses same JVM for Servlet

IBM's Application Server

- WebSphere Application Server
  - Available on many platforms
    - OS/400, OS/390, NT, AIX, . . .
  - WAS 5.0 Express
    - Cheap. No EJB support, others
  - WAS 5.0
    - Full function. Includes EJB support, full J2EE.
  - WAS Network Deployment
    - Highly scalable.

www.ibm.com/websphere



Tips

Links

Now we will review some links for additional reference information.

URL's

- www.ibm.com/software/awdtools/wds400
  WebSphere Development Studio for iSeries
- www.ibm.com/websphere/developer
  WebSphere Developer Domain
- www.ibm.com/alphaworks
  Lots of free Java stuff! Eg, Java Beans, XML tools
- www.mc-store.com/bookfromibmp.html
  Books on Java, Web, e-business
- www.w3.org
  Standards for HTML, CSS and JavaScript
- http://jakarta.apache.org
  Cool open-source stuff, like Struts

Summary

- **Architecture**
- Details
- How to code an eBusiness Application
- Example of an eBusiness Application
- Tips for creating eBusiness Applications Easily

This presentation first gave you an overview of the architecture of an e-business application. It then provided more details on the technology: DHTML, Java Beans, JSPs, and Servlets. Next we covered how to code an e-business application. An example was shown to illustrate. The last topic covered tips to easily create an e-business application.

Trademarks & Disclaimers

References in this document to IBM products or services do not imply that IBM intends to make them available in every country.
The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

AS/400	IBM(logo)
AS/400e	iSeries
e (logo) business	OS/400
IBM	

Lotus, Freelance Graphics, and Word Pro are registered trademarks of Lotus Development Corporation and/or IBM Corporation.
Domino is a trademark of Lotus Development Corporation and/or IBM Corporation.

C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.
Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.
UNIX is a registered trademark of The Open Group in the United States and other countries.
SET and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.
Other company, product and service names may be trademarks or service marks of others.

Information is provided "AS IS" without warranty of any kind.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

Information in this presentation concerning non-IBM products was obtained from a supplier of these products, published announcement material, or other publicly available sources and does not constitute an endorsement of such products by IBM. Sources for non-IBM list prices and performance numbers are taken from publicly available information, including vendor announcements and vendor worldwide homepages. IBM has not tested these products and cannot confirm the accuracy of performance, capability, or any other claims related to non-IBM products. Questions on the capability of non-IBM products should be addressed to the supplier of those products.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Contact your local IBM office or IBM authorized reseller for the full text of the specific Statement of Direction.

Some information in this presentation addresses anticipated future capabilities. Such information is not intended as a definitive statement of a commitment to specific levels of performance, function or delivery schedules with respect to any future products. Such commitments are only made in IBM product announcements. The information is presented here to communicate IBM's current investment and development activities as a good faith effort to help with our customers' future planning.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Photographs shown are of engineering prototypes. Changes may be incorporated in production models.

Disclaimer

- Acknowledgement:

  This presentation is a collaborative effort of the IBM Toronto iSeries Applicaiton Development presentation team, including work done by:

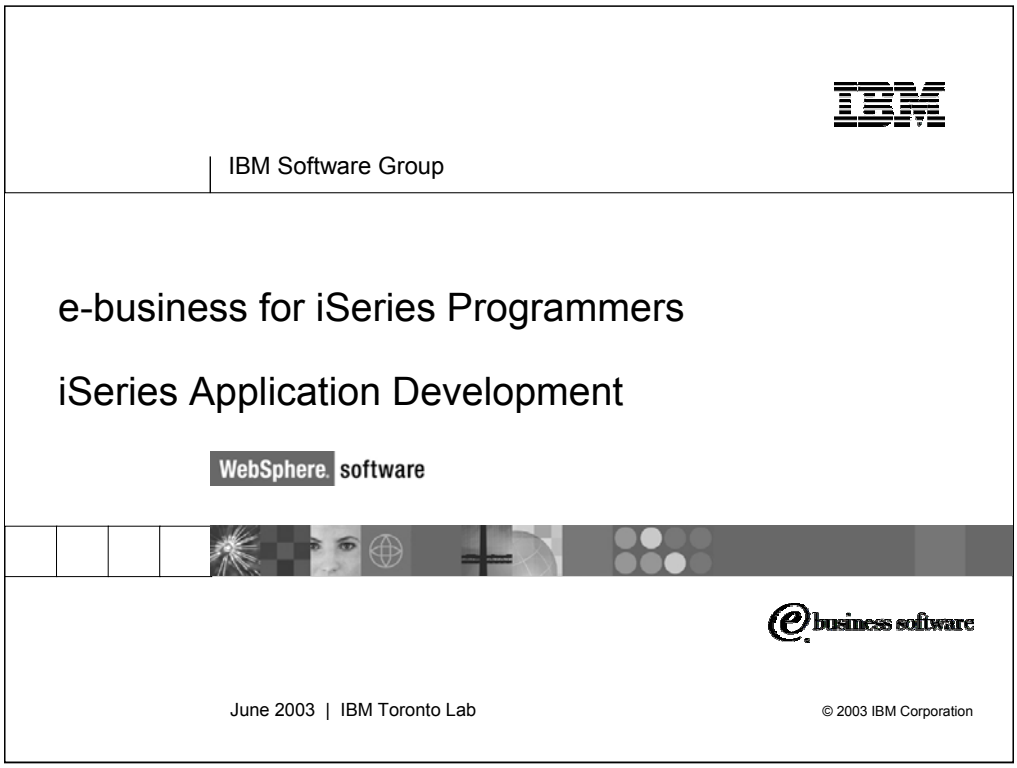
  Phil Coulthard, George Farr, Claus Weiss, Don Yantzi

- Disclaimer:

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

- Reproduction:

  The base presentation is the property of IBM Corporation. Permission must be obtained PRIOR to making copies of this material for any reason.



This presentation reviews how to move your iSeries applications to the Web.