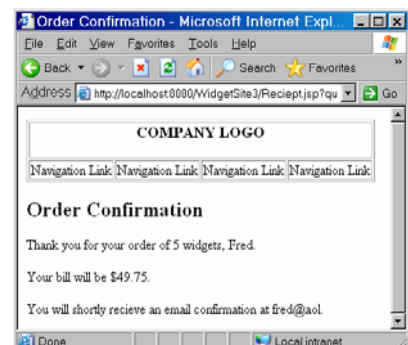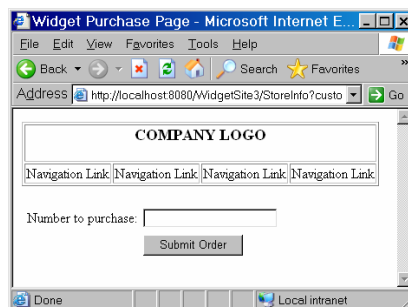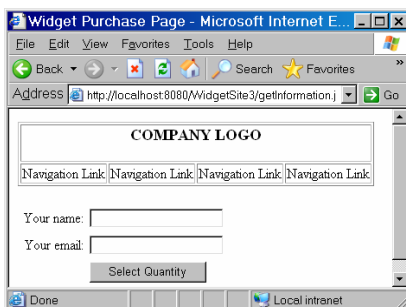# Server-side Web Programming

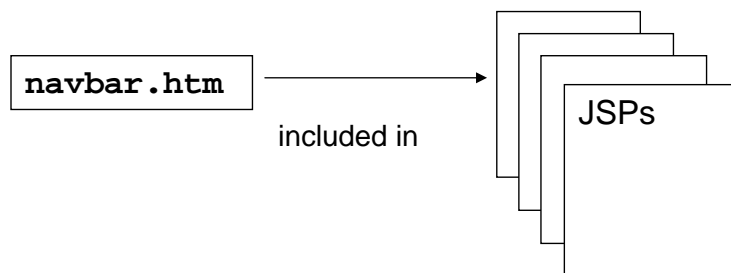## Lecture 11:
Modular Web Programming

Including Static Resources

- Pages on same site should have <u>same layout</u>
  - Site identifiable to user
  - User knows where to look for major components (navbar, etc.)
  - Example: CSS style sheets to unify appearance
- Example: All pages on site have same <u>navigation bar</u>

Including Static Resources

- Bad idea: <u>copy and paste</u> html for navigation bar into each file
  - If change layout, must modify <u>every file</u> on the site
- Better idea: "include" <u>single file</u> containing navigation bar into <u>every page</u> on site
  - Just need to modify <u>single file</u>
  - Done client side with <u>frames</u>

```
navbar.htm
```
included in → JSPs

Server-side Includes

- Can use <u>web container</u> to include files
  - Server page being generated for response
  - Just include files
- Syntax:
  `<%@ include file="`*`relative file location`*`" %>`

  For example:
  `<%@ include file="navbar.html" %>`

Including Dynamic Resources

- Can break <u>dynamically generated</u> JSP into separate files
  - Easier to develop and maintain as <u>group</u>

Configuration.jsp

Information.jsp

CreditCard.jsp

Including Dynamic Resources

- Syntax:
  `<jsp:include page="relative file location" />`

```
16    <body>
17        <%@ include file="/navbar.html" %>
18        <h2>Order Form</h2>
19        <form name="orderform" action="Validate" method="get">
20        <jsp:include page="Configuration.jsp" />
21        <jsp:include page="Information.jsp" />
22        <jsp:include page="CreditCard.jsp" />
23        <input type="submit" value="Configure">
24        </form>
25    </body>
26  </html>
```

```
11   <%
12       String quantity = request.getParameter("quantity");
13       if (quantity == null) quantity = "1";
14       String quantityError = (String)request.getAttribute("quantityError");
15       if (quantityError == null) quantityError = "";
16
17       String name = request.getParameter("customerName");
18       if (name == null) name = "";
19       String nameError = (String)request.getAttribute("nameError");
20       if (nameError == null) nameError = "";
21
22       String phone = request.getParameter("customerPhone");
23       if (phone == null) phone = "";
24       String phoneError = (String)request.getAttribute("phoneError");
25       if (phoneError == null) phoneError = "";
26
27       String email = request.getParameter("customerEmail");
28       if (email == null) email = "";
29   %>
30      <table cellspacing="5">
31       <tr>
32         <td align="right">Number to purchase:
33                        <br/>(at least 1)</td>
34         <td>
35           <input type="text" name="quantity" value="<%= quantity %>">*     
36           <%= quantityError %>
37         </td>
38       </tr>
39       <tr>
40         <td align="right">Your name:</td>
41         <td>
42           <input type="text" name="customerName" value="<%= name %>">*     
43           <%= nameError %>
44         </td>
45       </tr>
46       <tr>
47         <td align="right">Your Phone:
48                        <br/>(in format xxx-xxx-xxxx)</td>
49         <td>
50           <input type="text" name="customerPhone" value="<%= phone %>">*     
51           <%= phoneError %>
52         </td>
53       </tr>
```

Including Dynamic Resources

- Why different syntax?
- Steps in creating response:
  1. JSP translated into an equivalent servlet.
  2. Servlet is compiled. ←
  3. The doGet or doPost method of servlet run to create response page ←

- `<%@ include file="…" %>` executed <u>once</u> when compiled
  – More efficient

- `<jsp:include page="…" />` must be executed <u>every</u> time page called
  – Will generate <u>different</u> html each time based on user parameters