## Web Application Architecture (n-Tier)

<table>
<thead>
<tr>
<th>Client Domain</th>
<th>Flow Control &amp; Presentation Domain</th>
<th>Business Domain</th>
<th>Backend Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Contains copyrighted material of DaimlerChrysler AG, 2000. Used by permission.
**Responsibilities of the Domains**

**Client Domain**
- User Interface
- Data Entry Forms
- Minimum control data

**Flow Control Domain**
- Server side entry point for clients
- Application workflow
- UI production
- HTTP parameter conversion

**Business Domain**
- Application services
- Business logic
- No knowledge about
  - User Interface
  - Application workflow
  - Transport protocol
  - Persistence

**Backend Domain**
- Persistence
  - All access to database(s)
  - Encapsulates type of database
  - Encapsulates OR-Mapping
- Application Integration
  - ERP-Systems
  - Legacy-Systems

---

**Web Application Architecture (Basic)**

<table>
<thead>
<tr>
<th>Client Domain</th>
<th>Flow Control &amp; Presentation Domain</th>
<th>Business Domain</th>
<th>Backend Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>JSP</td>
<td>Business Object</td>
<td>Persistence Services, RDBMS, Legacy Syst., ERP</td>
</tr>
<tr>
<td>Client Browser</td>
<td>Servlet</td>
<td>Business Command</td>
<td></td>
</tr>
</tbody>
</table>

---

- **knows object**
- **process flow**
- **process steps**
Interaction Steps (Basic)

1. A web browser calls a server via HTTP with a specific URL. On server-side a ControllerServlet gets the request.
2. The ControllerServlet determines which service is required and calls a BusinessCommand to do the work.
3. The BusinessCommand uses services from backend systems like database access and host procedure calls.
4. The BusinessCommand receives business objects and executes further business logic on or with them.
5. The BusinessCommand returns all necessary objects to the ControllerServlet.
6. The ControllerServlet prepares ViewBeans for the output and forwards control to a JavaServer Page.
7. The JSP creates the HTML output from static layout data and the received ViewBeans.

Prerequisites & Implications

- There is only one URL for the whole application.
- To determine which service is required a special parameter must be passed from the client to the server.
- To keep all parameter hidden it's necessary to use POST requests rather than GET requests.
  - Only FORMS
- If more than one action should be possible on the same data of a HTML page, then scripting is mandatory.
HTML Example

```html
...<FORM name="main" action="/MyApp/Service" method="POST">
    <INPUT type="HIDDEN" name="PageID" value="23">
    <INPUT type="HIDDEN" name="ServiceID" value="DEFAULT">
    ...
    <INPUT type="button" value="Edit" onclick="invokeService('EDIT')">
    <INPUT type="button" value="Delete" onclick="invokeService('DELETE')">
</FORM>
...```

Javascript Example

The following script can be used in HTML pages to submit a form for different required services:

```javascript
...<SCRIPT language="Javascript">
    function invokeService( serviceName )
    {
        document.forms["main"].elements["ServiceID"].value = serviceName;
        document.forms["main"].submit();
    }
</SCRIPT>
...```
A web browser calls a server via HTTP with a specific URL. There an AuthenticationServlet checks whether the client is correctly logged in to the server.

If the client is not yet logged in, the AuthenticationServlet returns a Login-Dialog to the client.

If the client is logged in, the AuthenticationServlet forwards the request to the ControllerServlet.

The ControllerServlet „asks“ the AuthorizationHandler for all authorization data for the logged in user.

The AuthorizationHandler checks and returns all necessary data about the user’s access rights.

The ControllerServlet „asks“ the ScreenFlowManager, if the required service is valid.

The ScreenFlowManager checks the current state of the application and returns a JSP in case of any detected error. Otherwise it returns null.
### Interaction Steps (Extended)

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>The ServletController detects the correct DisplayCommand to handle the request and calls its execution method</td>
</tr>
<tr>
<td>8</td>
<td>The DisplayCommand validates and converts the request parameters to its corresponding business command parameter and invokes the necessary method(s) of the command</td>
</tr>
<tr>
<td>9</td>
<td>The BusinessCommand uses services from backend systems like database access and host procedure calls.</td>
</tr>
<tr>
<td>10</td>
<td>The BusinessCommand receives business objects and executes further business logic on or with them.</td>
</tr>
<tr>
<td>11</td>
<td>The BusinessCommand returns all necessary BusinessObjects to the DisplayCommand</td>
</tr>
<tr>
<td>12</td>
<td>The DisplayCommand wraps the BusinessObjects into ViewBeans and returns those to the ControllerServlet.</td>
</tr>
<tr>
<td>13</td>
<td>The ControllerServlet forwards the ViewBeans to the ScreenFlowManager and asks for information about the next response.</td>
</tr>
<tr>
<td>14</td>
<td>The ScreenFlowManager selects the appropriate JSP file (or XSL file) and returns it to the ControllerServlet</td>
</tr>
<tr>
<td>15</td>
<td>The ControllerServlet forwards the ViewBeans and the response information to the PresentationManager</td>
</tr>
<tr>
<td>16</td>
<td>The PresentationManager prepares for output processing and forwards control to a JSP.</td>
</tr>
<tr>
<td>17</td>
<td>The JSP output, generated from static HTML and the passed through ViewBeans goes back over HTTP to the client’s web browser</td>
</tr>
</tbody>
</table>
An instance of a ScreenFlowManager exists for each session
- It determines and observes the workflow of one client
- It could be build as a ready to use component that allows specialization through subclassing
- It can be based on configuration data (e.g. from XML files)
- That configuration data might be generated from a workflow modelling tool

Responsibilities of the ScreenFlowManager

1) Validation, if a request is correct (Step 3)
   Depends on:
   - Preconditions (previous actions/data -> session data)
   - Access rights (user data -> session data)
   - Defined application flow (configuration data -> session data)

2) Determination which page has to be shown next (Step 14)
   Depends on:
   - Executed service (request data)
   - Request parameters (request data)
   - Previous actions/Data (session data)
   - Result of business service execution (service result data)
   - Defined JSPs for the various alternatives (configuration data)
Advantages of that Architecture

- Having only one servlet reduces configuration effort on the application server
- Single entry point simplifies authorization check
- Clear separation of responsibilities
- Parts like ControllerServlet and ScreenflowManager can be implemented in a generic way (Frameworks)
- Presentation layer is completely decoupled from business layer
- Workflow can easily be modified or replaced. Even user/role specific workflow is possible

Disadvantages of that Architecture

- Too complex for smaller applications
- Much conversion of parameters and other data
- Lots of data has to be handed over from one object to another ScreenflowManager<->ControllerServlet<->DisplayCommand
- It's hard to decouple business logic and workflow
SUN J2EE Blueprints
Designing Enterprise Applications
http://java.sun.com/j2ee/download.html#blueprints

IBM Redbooks
Servlet and JSP Programming with IBM WebSphere Studio and VisualAge for Java
http://www.redbooks.ibm.com/abstracts/sg245755.html

IBM Redbooks
Design and Implement Servlets, JSPs, and EJBs for IBM WebSphere Application Server
http://www.redbooks.ibm.com/abstracts/sg245754.html

IBM Library Papers
IBM Application Framework for e-business: Web Application Programming Model

Article about JSP Model 2 from Govind Seshadri

Presented by
Manfred Duchrow
Daedalos Consulting GmbH
Ruhrtal 5
58456 Witten

manfred.duchrow@daedalos.com
daedalos.duchrow@daimlerchrysler.com