WebE - Design

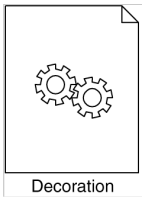
Source:
Building Web Applications with UML Second Edition By Jim Conallen, 2002

IF3037
STEI ITB - 2008

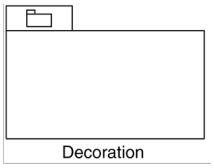
WAE – Component View

- Static Page
- Dynamic Page
- Physical Root

Static Page



Dynamic Page



Physical Root

Static Page

- Metamodel: Component
- A resource that can be directly requested by a client browser
- Performs no server-side execute and is delivered directly from the file system to the client intact
- Constraint:

  Cannot realize logical components that execute on the server, that is, server pages.

  Static pages can realize only client pages

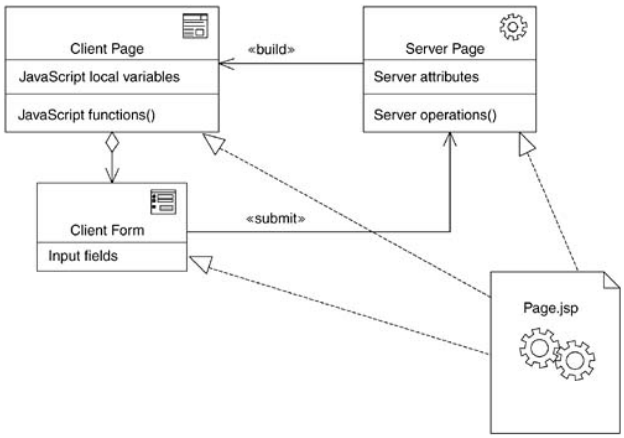
Dynamic Page

- Metamodel: Component
- A resource that can be requested by a client browser
- When requested or delegated to via a «forward» relationship, server-side processing takes place
- The results of this processing can change the state of the server and be used to construct some of the HTML that is streamed out to the requesting client
- Can accept user input submitted by forms
- Constraints: Must realize a single server page

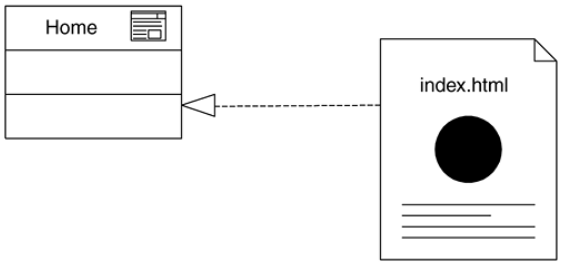
Physical Root

- Metamodel: component package
- An abstraction of a file hierarchy that contains requestable resources
  - Clients request static or dynamic files directly from this hierarchy
  - Maps directly to a Web server file system directory
- Tagged values:
  - Host name, the name of the host of the Web server, such as www.mycompany.com.
  - Context, the application context. The context appears as a top-level directory, such as www.myco.com/appcontext.

Logical-view classes realized by dynamic page component



Client pages realized by static page components



Designing Web App

- Most of the activities are the same as for any client/server system:
  - partitioning the objects into the system's tiers and developing the necessary infrastructure and helper classes to add to the analysis model
- In Web-centric systems, Web pages are first-class objects, and the WAE gives us a notation for including them in our design models
- Proper partitioning of the business objects in a Web application is critical and depends on the architecture
  - Objects may reside exclusively on the server, the client, or both
  - Thin Web client applications place all objects behind the server, running either on the Web server or on another tier associated with the server.
  - Thick Web client applications allow some objects to execute on the client.
  - Web delivery applications have the most freedom in the placement of objects, being essentially distributed object systems that happen to use a browser.

Thick Web client Web applications

- When designing thick Web client Web applications, a large number of the objects discovered during analysis can be easily partitioned in the first pass
- For the most part, persistent objects, container objects, shared objects, and complex objects all belong on the server
  - Objects with associations to server resources, such as databases and legacy systems, also belong in the server tier
  - Objects that maintain static associations or dependencies with any of these objects must also exist on the server
- An object can exist on the client if it has no associations or dependencies with objects on the server and has associations and dependencies only with other client resources, such as browsers and Java applets
  - Candidate objects for the partitioning on the client are field validation objects, user interface controls, and navigation assisting controls
  - Client objects can be implemented with JavaScript, JavaBeans, applets, ActiveX (COM), or even plug-ins

Web Delivery Web Applications

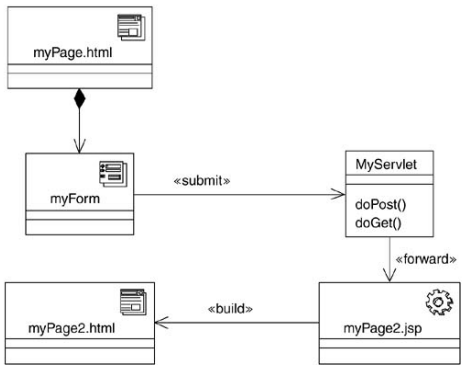
- One of the primary reasons for distributing objects to the client is to take some of the load off the server.
- It is also natural to place objects in the part of the system where they will be most effective
- As a general rule, place objects where they have the easiest access to the data and the collaborations they require to perform their responsibilities
- If an object can exist on the client and if most, if not all, its associations are on client objects, that object is a likely candidate for placement on the client

Identifying web pages

- While the objects are being partitioned, Web pages are also being defined
  - Involves the discovery of Web pages and their relationships with one another and with the objects of the system
- Web page design elements—client and server pages—are discovered by first looking at the UX model and understanding the software architecture document
- In the early generations of Web applications, Web pages mapped one to one to what we now refer to as UX model screens.
  - Each page was responsible for preparing its output by interacting with server-side objects
- Today's development environments and frameworks enable us to build more robust, sophisticated Web applications with the same effort we used to create simple ones just four years ago
  - The two predominant Web architecture frameworks available today are J2EE and .NET.

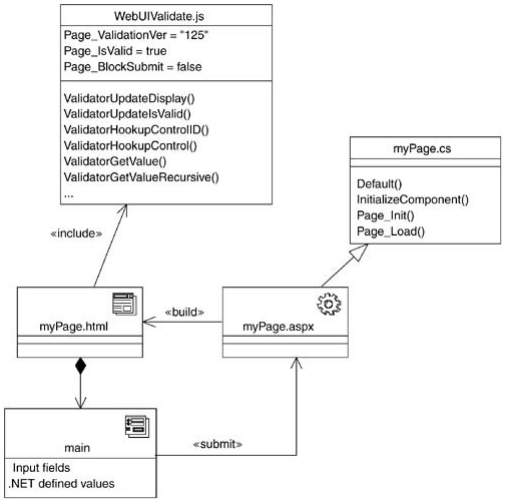
JavaServer Page Model 2 Architecture

- JavaServer Pages Model 2 Architecture describes the general philosophy of separating the two mechanisms for accepting and processing input and building output.
- The general strategy is to use servlets for accepting all user-supplied input.
- A servlet is a completely Java-written component.
- With the submitted data accepted and processed, the servlet delegates the building of the response page to a JSP.
- JavaServer pages are more appropriate for building HTML output since the majority of the code in the component is often HTML.



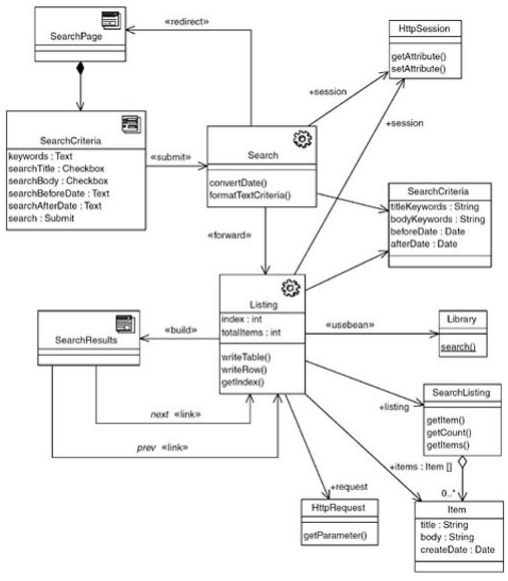
.NET paradigm for handling user input

- The ASPX file myPage.aspx is modeled with a «server page» stereotyped class.
- The C# code behind class myPage.cs is a superclass to the ASPX class and contains the majority of the event-handling code.
- The general strategy is:
  to leave the ASPX file to focus on building the output page.

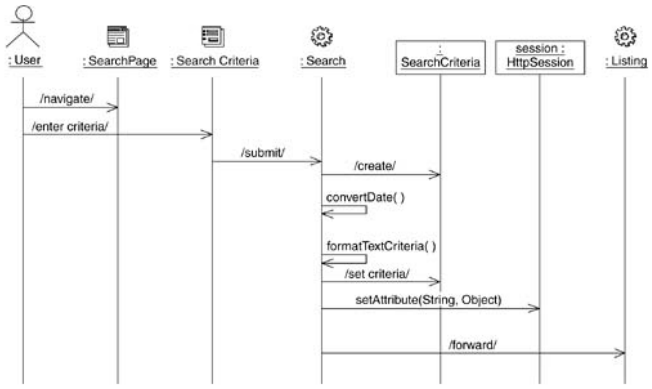


Design model fragment of catalog search functionality

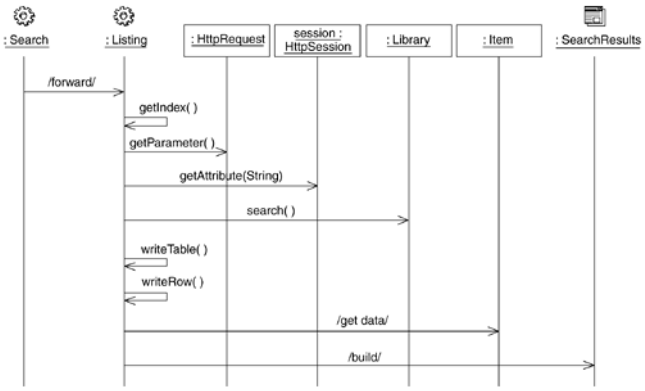
- The Web page with the search form is not built by a server page, so there is the option of using a «static page» component for its implementation.
- The search form submits its data to the Search «server page», which is responsible for accepting the search request and processing
- Building the response page is the job of the Listing «server page». When the Search page has completed the search, control is forwarded to the Listing page, which uses the built-in session management mechanism for J2EE applications (HttpSession).
- The SearchResults client page implements a page-scrolling mechanism—Paged Dynamic List, Page-by-Page Iterator—and has two «link» relationships to itself.
- The parameter tag value for each of these links has a value to indicate which direction to scroll: { parameters="scroll=next" } and { parameters="scroll=prev" }.



Search catalog sequence diagram using stereotyped elements



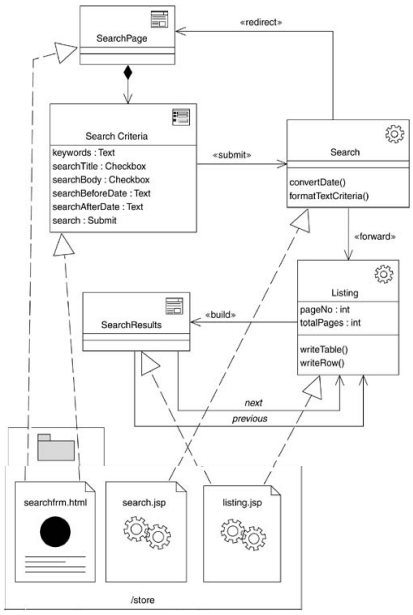
Build Listing sequence diagram



Component view realizations of logical-view classes

- The components are all under the physical root at the same level.
- The physical root is named /store, which also happens to be the context.
- A client browser would request the search page by using the URL:

  http://localhost/store/searchform.html.

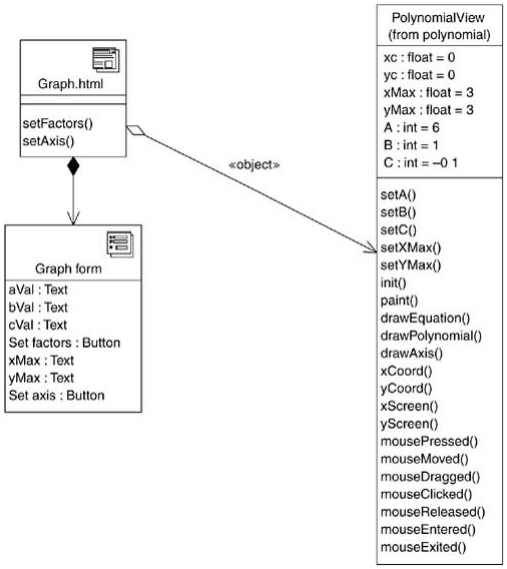


Client-Side Scripting

- Designing Web applications that have dynamic client pages requires careful attention to the partitioning of the objects
- Thick Web client applications can have all sorts of objects and activity on the client

Modeling applets and other embedded controls

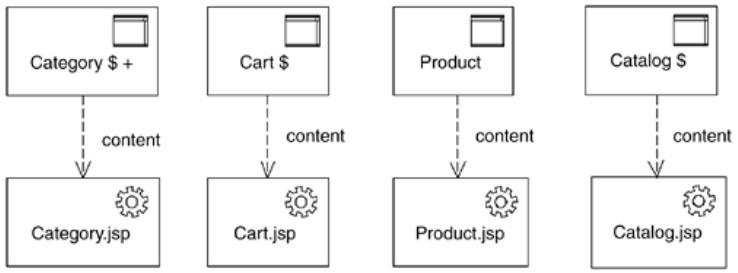
- A simple class diagram with a «client page» that has two defined JavaScript functions.
- Shows an «object» stereotyped association to the PolynomialView applet class.
- The client page also has an HTML form included



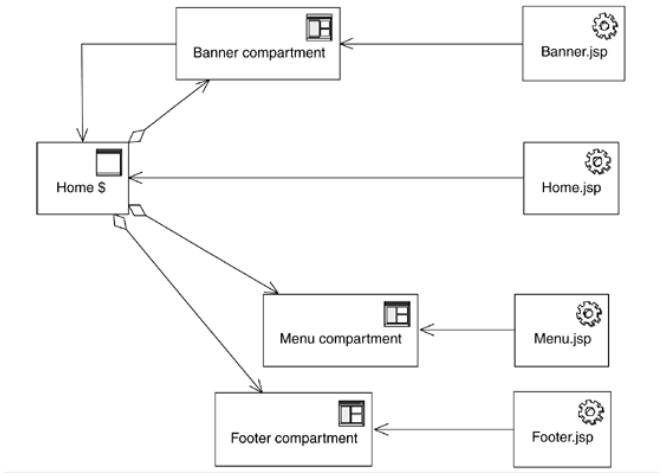
Mapping to the UX Model

- To establish and to maintain mappings directly between use case and design models to the UX model
- The mappings are captured in class diagrams that contain UX screens and design model classes with dependency relationships connecting them
  - show which Web page classes realize which UX model screens
  - simple architectures have a one-to-one mapping between a Web page class—client or server page—and the screen
  - in more complex architectures, Web page classes are responsible for delivering multiple screens

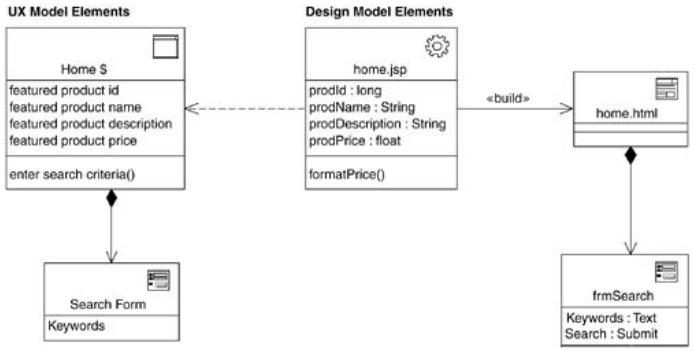
Design model mapping with UX model



Design model mappings with UX model, including «screen compartment» mappings



Design model implementing UX model contract



Guidelines for Web Application Design

- Be wary of using the latest capabilities of browsers
  - The browser wars continue, and it is difficult to predict which features will eventually become standard
- Think about how the page is going to be tested
  - Don't use visual cues to let the actor know when the page is safe to interact with unless these same cues are accessible by an automated testing tool
- Avoid the temptation to use multiple browser windows simultaneously
  - Although a useful feature for some applications, designing and maintaining two client interfaces requires more than double the effort
- Keep the focus of server pages on the construction of the user interface
  - Avoid placing business logic code in the server page. Use external objects to encapsulate this type of logic