This Lecture

E. Gamma, R. Helm, R. Johnson and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995.


Some Patterns

• Presentation Layer
  • *Model View Controller*
  • Controller
    • *Page Controller*
    • *Front Controller*
  • View
    • *Template View*
    • *Transform View*
Revenue Recognition

- Revenue Recognition Problem

<table>
<thead>
<tr>
<th>Product</th>
<th>Contract</th>
<th>Rev Recog</th>
</tr>
</thead>
<tbody>
<tr>
<td>type (W,D,S)</td>
<td>date signed revenue</td>
<td>amount date</td>
</tr>
</tbody>
</table>

Rules for different product types:

W: all right away
S: 1/3 now, 1/3 60 days, 1/3 90 days
D: 1/3 now, 1/3 30 days, 1/3 60 days
An object model of the domain that incorporates both behavior and data

- Contract
  - recognizedrevenue
  - computeRecogs
- Product
  - computeRecogs
- Strategy
  - recognizedrevenue
  - computeRecogs
- CompleteStrategy
  - recognizedrevenue
  - computeRecogs
Problem

- The rules are made up by accountants and lawyers ...
- They might change (retroactively) for non-technical reasons
- Computing and inserting recognitions in database when contract is signed won’t work!
New Design ...

- Calculate recognized revenue dynamically
How a Strategy Calculates Recognized Revenue

```
Date effectiveDate;
RecogStrategy prev;
Money recogRev(c) {
    Money amt = 0;
    if( c.date < effectiveDate ) {
        amt = amt + prev.recogRev(c);
    }
    ... 
    return amt;
}
```

- Delegate old revenue to previous strategy
Remarks

• This works b/c strategy is an object *not* a value ... so as law changes the strategy can be updated “in place”

• What if new law changes strategy for some products but not others?

• A more elaborate scheme might cache previously computed recognitions for performance
Caching Design ...

Contract
  recogRevenue

Product
  recogRev(c)
    * 1

Recog Strategy
  recogRev(c)
    * 1

RevRecog
  amount
date

CompleteStrat
  recogRev(c)

3WayStrat
  recogRev(c)
Table Module

A single instance that handles the business logic for all rows in a database table

```
Table Module

<table>
<thead>
<tr>
<th>Contract</th>
<th>computeRecogs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>getProductType</td>
</tr>
<tr>
<td>Rev Recog</td>
<td>insert(id, amt, date)</td>
</tr>
<tr>
<td></td>
<td>regdRev(id, date)</td>
</tr>
</tbody>
</table>

DB1
```

• Works well only for simple hierarchical schemas

• Same problem (early binding) as theDomain Model solution above
  • similar solution
  • but “caching” interpretation of recognition table is problematic
Presentation

• Model-View-Controller

• Dates back to SmallTalk in 1970s

Model-View-Controller

- Separate the presentation from the model
- Separate the controller from the view
Model-View-Controller

• Separate the presentation from the model
  • presentation depends on model, but model is independent of presentation
  • application may have multiple presentations for the same model
  • or even multiple presentations on screen simultaneously
  • Observer pattern!
Model-View-Controller

• Separate the controller from the view
  • clean ...

• may share a view among controllers
  • editable vs readonly

• the Servlet vs JSP distinction ...
Model-View-Controller
Model-View-Controller in J2EE

- Controller is a servlet
  - may immediately invoke a session bean
- Model is (session and entity) EJBs
- Data for view is entity beans
  - stored in accessible place by controller
- View is a JSP
  - controller forwards to view
Implementation in J2EE

- Servlet handles request
- Servlet invokes business logic (EJBs) to obtain results (entity EJBs)
- Put results in view-accessible location (request, session or servlet context)
- Forward request to view (a JSP page)
- Extract data from beans and display
Example - Request Based Data

MyServlet(request, response) {
    // invoke business logic to get data values
    ValueObject val = ...
    request.setAttribute("key", val);
    // determine view page
    String viewPageName = "theViewPage.jsp";
    // forward data to the page
    RequestDispatcher disp = request.getRequestDispatcher(
        "/WEB-INF/" + viewPageName);
    disp.forward(request, response);
}

<jsp:useBean id="key" type="package.ValueObject"
    scope="request"/>
...
<jsp:getProperty name="key" property="someProperty"/>
Example - Session Based Data

```java
MyServlet(request, response) {
    // invoke business logic to get data values
    ValueObject val = ...
    HttpSession session = request.getSession();
    session.setAttribute("key", val);
    // determine view page
    String viewPageName = "theViewPage.jsp";
    // forward data to the page
    RequestDispatcher disp =
        request.getRequestDispatcher(
            "/WEB-INF/" + viewPageName);
    disp.forward(request, response);
}
```

```xml
<jsp:useBean id="key" type="package.ValueObject"
    scope="session"/>
...
<jsp:getProperty name="key" property="someProperty"/>
```
Example - Application Based Data

MyServlet(request, response) {
    // invoke business logic to get data values
    ValueObject val = ...
    ServletContext ctx = getServletContext();
    ctx.setAttribute("key", val);
    // determine view page
    String viewPageName = "theViewPage.jsp";
    // forward data to the page
    RequestDispatcher disp =
        request.getRequestDispatcher(
            "/WEB-INF/" + viewPageName);
    disp.forward(request, response);
}

<jsp:useBean id="key" type="package.ValueObject"
    scope="application"/>
...
<jsp:getProperty name="key" property="someProperty"/>
- An object that handles a request for a specific page or action on a Web site
- Client URL directly reaches page controller
Front Controller

- A single controller handles all requests for a Web site
- Passes request to appropriate command controller
Application Controller

- A centralized point for handling screen navigation and the flow of an application
- “Wizard” style of interaction
- Treat application UI as a state machine
Template View

- Render information into HTML by embedding syntactic markers in an HTML page
- Reminds you of JSP, doesn’t it!
Transform View

- A view that processes domain data element-by-element and transforms it into HTML

- Contrast:
  - A template view is organized around the output page
  - A transform view is organized around the data elements

- Dominant example:
  - data is XML (or a data transfer object that can serialize itself into XML)
  - transformations are XSLT