SKILLS MATTER

SiteMesh

2007

GRAILS eXCHANGE

http://www.grails-exchange.com | http://www.grails.org | http://skillsmatter.com

Joe Walnes

- Open Source
  - Author of SiteMesh, XStream, Hamcrest, QDox, Objenesis, Nmock…
  - Contributor to WebWork, Groovy, jMock, ActiveMQ, PicoContainer…
  - Co-founder of OpenSymphony project
  - Wrote some books, articles and stuff, but that's dull
- Day Job
  - Senior software engineer @ Google

SiteMesh helps to separate look and feel from web content

But... what's wrong with includes?

Includes are invasive

signin.jsp

```
<jsp:include page="header.jsp"/>
Please sign in
<jsp:include page="middle.jsp"/>
<form action="signin">
 <p>Username:<br/> ...
 <p>Password:<br/> ...
 <p><input type="submit"/></p>
</form>
<jsp:include page="footer.jsp"/>
```

search.jsp

```
<jsp:include page="header.jsp"/>
Search
<jsp:include page="middle.jsp"/>
<form action="search">
 <p><input type="text"/> ...
</form>
<jsp:include page="footer.jsp"/>
```

results.jsp

```
<jsp:include page="header.jsp"/>
Results
<jsp:include page="middle.jsp"/>
<ul>
  <j:foreach id="result">
    <li> ...
  </j:foreach>
</ul>
<jsp:include page="footer.jsp"/>
```

Includes cause fragmentation of markup

header.jsp

```
<html>
 <head>
  <style>...</style>
  <meta ...>
 </head>
 <body>
  <table class="page">
   <tr>
    <td><img src="logo.gif"></td>
    <td class="header">
```

middle.jsp

```
    </td>
   </tr>
   <tr>
    <td>Nav bar ...</td>
    <td class="content">
```

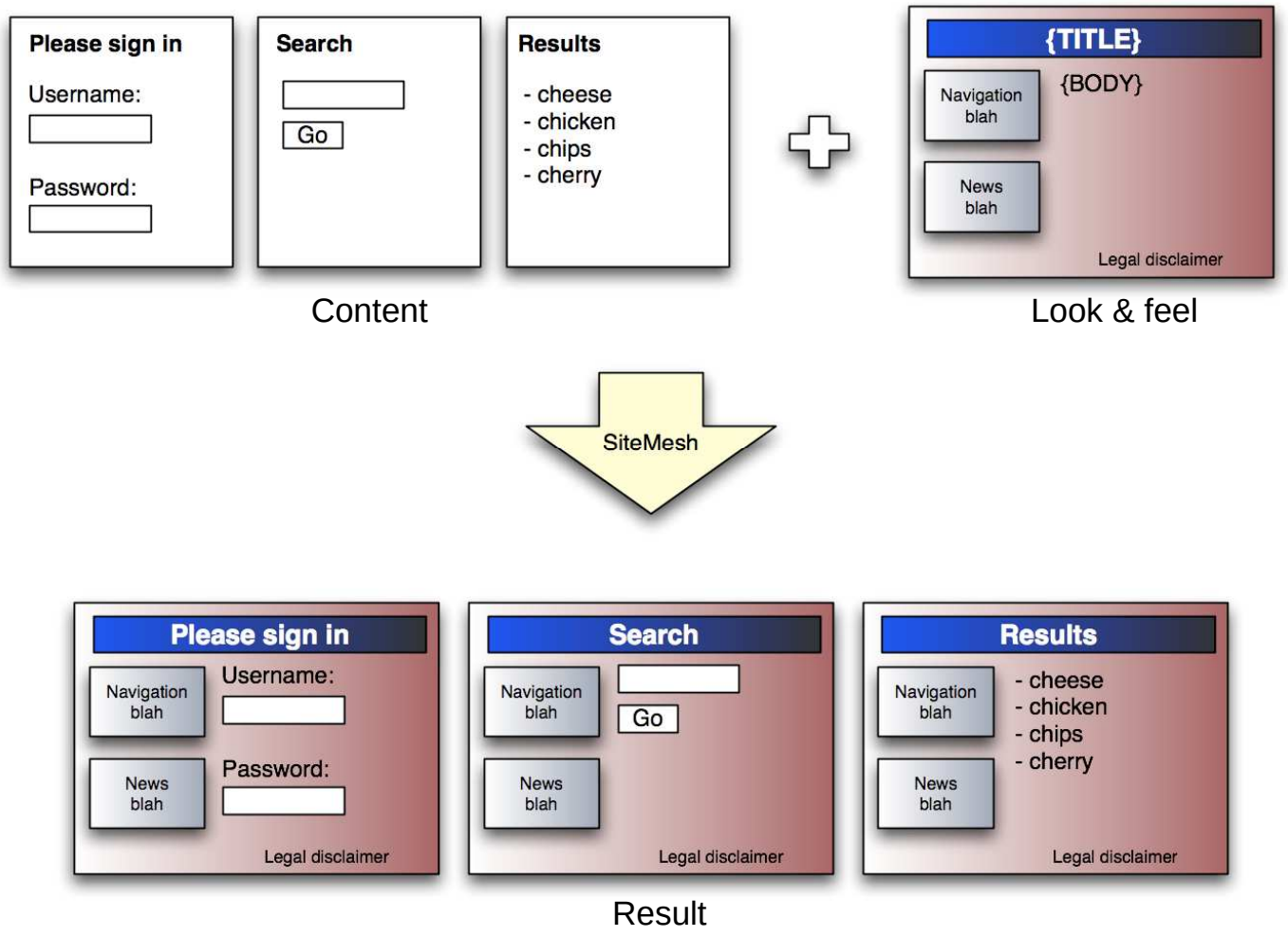
footer.jsp

```
    </td>
   </tr>
  </table>
  Copyright, legal disclaimer ...
 </body>
</html>
```

- Incomplete documents
  - Confuses tools
  - Confuses humans

The SiteMesh approach...

SiteMesh provides a clean separation of content from look and feel

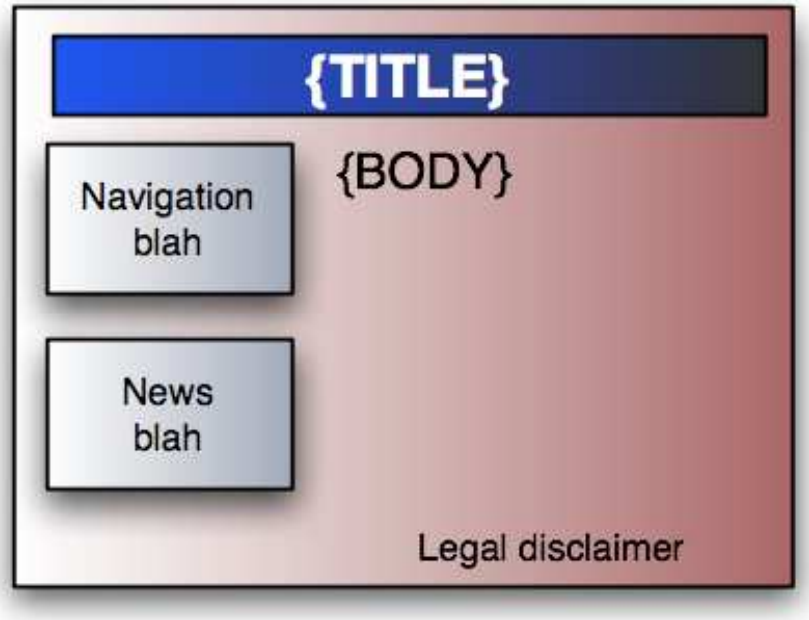


'Content' pages contain vanilla markup

```
<html>
 <head>
  <title>Please sign in</title>
 </head>
 <body>
  <form action="signin">
   <p>
    Username:<br/>
    <input type="text" name="u"/>
   </p>
   <p>
    Password:<br/>
    <input type="password" name="p"/>
   </p>
  </form>
 </body>
</html>
```

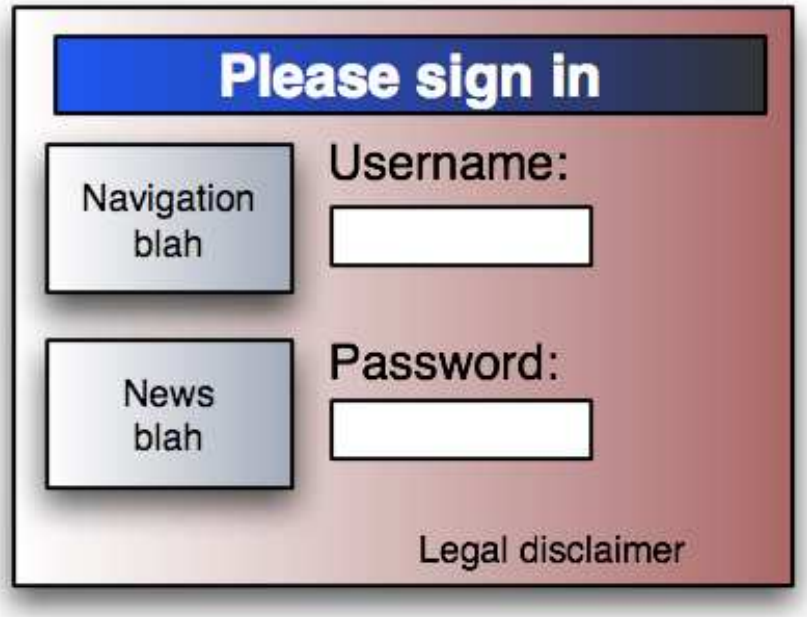
'Decorator' pages contain look and feel

```
<html>
 <head>
  <title><decorator:title/></title>
  <link rel="stylesheet" href="style.css"/>
  <decorator:head/>
 </head>
 <body>
  <table class="page">
   <tr>
    <td><img src="logo.gif"></td>
    <td class="header"><decorator:title/></td>
   </tr>
   <tr>
    <td>Nav bar ...</td>
    <td class="content">
     <decorator:body/>
    </td>
   </tr>
  </table>
  Copyright, legal disclaimer ...
 </body>
</html>
```



SiteMesh merges them together at request time

```
<html>
 <head>
  <title>Please sign in</title>
 </head>
 <body>
  <table class="page">
   <tr>
    ...<td class="header">Please sign in</td>
   </tr>
   <tr>
    ...<td class="content">
    <form action="signin">
     <p>Username:<br/>...</p>
     <p>Password:<br/>...</p>
    </form>
   </td>
   </tr>
  </table>
  Copyright, legal disclaimer ...
 </body>
</html>
```



Benefits of SiteMesh

SiteMesh does not fragment markup

- Both content and decorator pages contain valid HTML documents
  - Plays well with tools
  - Less chance of human error

SiteMesh is not invasive

- Content pages need not even be aware of SiteMesh

SiteMesh does not tie you to a single rendering technology

- Templating systems
  - JSP, GSP, Velocity, FreeMarker, Servlets…
- Supports static .html files
- Technologies can be mix'n'matched in the same page
- Works with existing applications

SiteMesh is a mature project

- Open Source
  - Apache 2 License
- Used extensively on high-traffic sites
  - It’s fast!

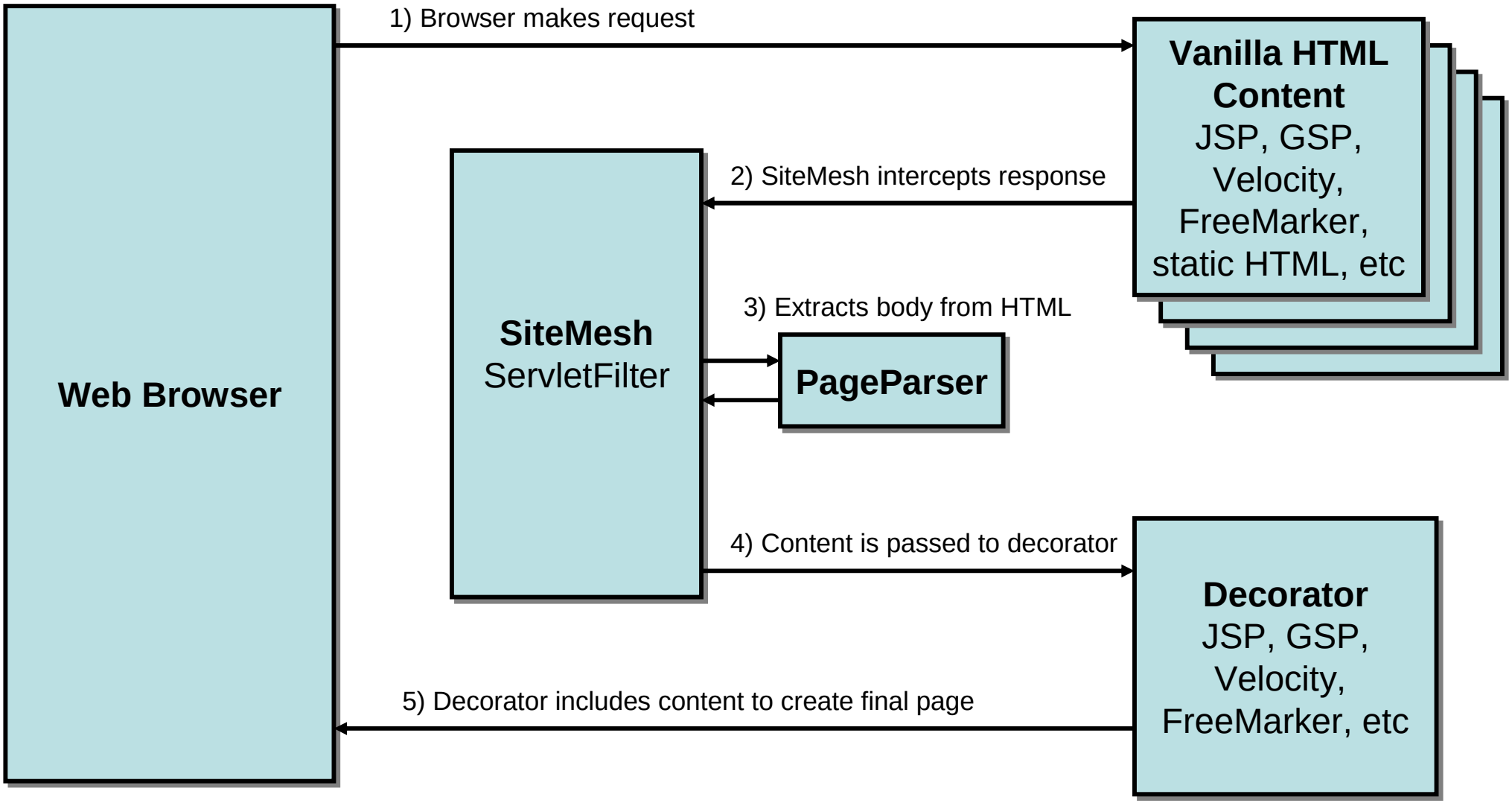
SiteMesh is easy to get started with

- You don’t have to start from scratch
- Drops in to an existing application in minutes

SiteMesh is extensible

- Easy things are easy
- Extension points exist for customization

How it works


Web Browser
1) Browser makes request
Vanilla HTML Content
JSP, GSP, Velocity, FreeMarker, static HTML, etc
2) SiteMesh intercepts response
SiteMesh
ServletFilter
3) Extracts body from HTML
PageParser
4) Content is passed to decorator
Decorator
JSP, GSP, Velocity, FreeMarker, etc
5) Decorator includes content to create final page

Fits in the existing technology stack

- Orthogonal to controller layer
  - Plays nicely with Grails controllers
  - and Struts, WebWork, SpringMVC, Stripes, home grown frameworks, etc
- Orthogonal to view layer
  - SiteMesh splits the view into look/feel and content
  - Leverages existing view technologies

Get started - in 3 steps

1) Activate the filter

- Put sitemesh.jar in WEB-INF/lib
- Add the filter to web.xml

```
<web-app>

  <filter>
    <filter-name>sitemesh</filter-name>
    <filter-class>
      com.opensymphony.module.sitemesh.filter.PageFilter
    </filter-class>
  </filter>

  <filter-mapping>
    <filter-name>sitemesh</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>

  … rest of web.xml…

</web-app>
```

2) Configure decorator mappings

- Create WEB-INF/decorators.xml

```
<decorators>

  <decorator name="main" page="/decorators/main.jsp">
    <pattern>*</pattern>
  </decorator>

  <decorator name="admin" page="/decorators/admin.jsp">
    <pattern>private/*</pattern>
    <pattern>admin/*</pattern>
  </decorator>

</decorators>
```

3) Create a decorator

```
<%@ taglib prefix="decorator"
    uri="http://www.opensymphony.com/sitemesh/decorator" %>
<html>
 <head>
  <title><decorator:title default="Default title"/></title>
  <link rel="stylesheet" href="..."/>
  <decorator:head/>
 </head>
 <body>
  <h1><decorator:title/></h1>
  <div style="content"><decorator:body/></div>
 </body>
</html>
```

Advanced topics

DecoratorMappers select the decorator

- Create WEB-INF/sitemesh.xml config
- Some examples
  - PrintableDecoratorMapper
    - Allows a printer friendly decorator to be used when URL params contain 'printable=true'
  - PageDecoratorMapper
    - Allows content pages to specify which decorator to use
    - <meta name="decorator" content="two-column"/>
  - UserAgentDecoratorMapper
    - Allows decorators to be overridden for specific browsers
    - dec.jsp (default), dec-moz.jsp, dec-ie.jsp, dec-webkit.jsp

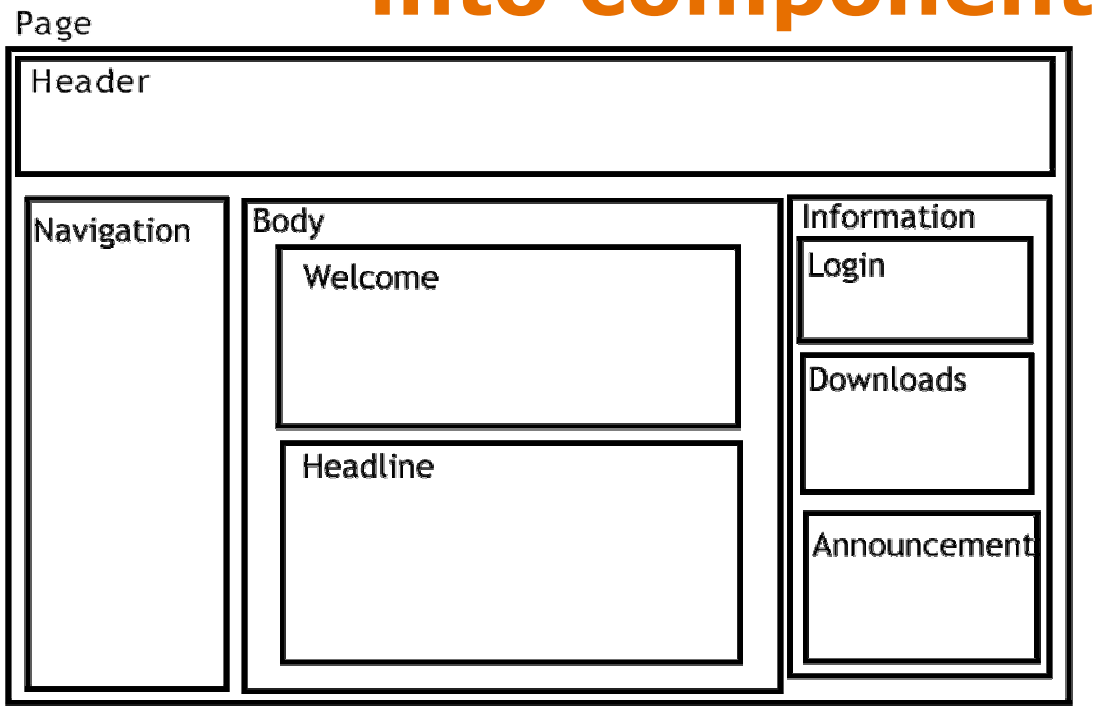
Implementing a custom DecoratorMapper

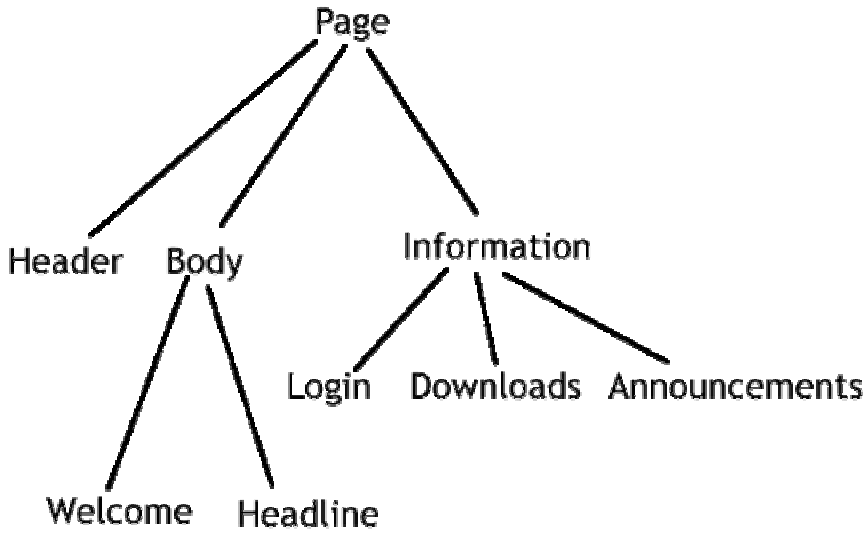
```
public class CustomDecoratorMapper
    extends AbstractDecoratorMapper {

  public Decorator getDecorator(HttpServetRequest request,
                                Page page) {
    if(request.isUserInRole("admin")) {
      // Show special admin decorator to admins.
      return getNamedDecorator("admin");
    } else {
      // Otherwise, use default DecoratorMapper
      return super.getDecorator(request, page);
    }
  }

}
```

Pages can be broken down into component hierarchies





Each component can have a decorator applied

Data can be passed from content page to decorator

- PageParser extracts data and makes available through API
  - Contents of <body>, <head> and <title> tags
  - All <meta> tag values
  - Additional attributes of <html> and <body> tags
  - Special <content> and <param> tags
  - MS word document properties
- Example usages:
  - Allow content page to influence navigation generation code in decorator
  - Allow content page to influence how decorator does layout (e.g. 3 columns)

The HTML parser is extensible

- Features:
  - Performance: outperforms the fastest Java XML parsers
  - Tolerance: deals with very poorly formed HTML
  - Extensibility: users can create custom rules for performing on-the-fly transformations and content-extraction
- Example customizations:
  - Link rewriting
  - Extracting list of headings for navigation

Summary

- Separates look and feel from content
- Allows mix'n'match of technologies
- Plays nicely with other frameworks
- Easy to drop into existing apps
- Doesn't alter your URL structure
- Allows decorators to be applied to smaller components of pages
- Simple, fast and flexible
- http://opensymphony.com/sitemesh

KTHXBYE!