An Introduction to Building Web Application Using Jakarta Struts Framework

Presented to Indianapolis Java User Group

10/30/2002

Wei Li

Challenge from Building Web Applications

- Stateless protocol
- Multiple tiers
- …

Building Web Application with Java - Why it is good

- Build once, run everywhere (platform-independent, vendor-independent)
- Build it in an object-oriented way
- Reusable components
- Flexible and easy to be maintained and extended

Building Web Application with Java - choices

- Pure Java Servlets (control + presentation)
- Java Servlets (control) + template engine (presentation) (Velocity, WebMacro, Tea)
- Java Servlets (control) + XML and XSLT (presentation)
- JSP (presentation + control) + Java Beans (control+model) (Model 1)
- JSP(presentation) + Java Servlets (control)+ Java Beans (model) (Model 2)

Problems with Pure Java Servlets - sample code

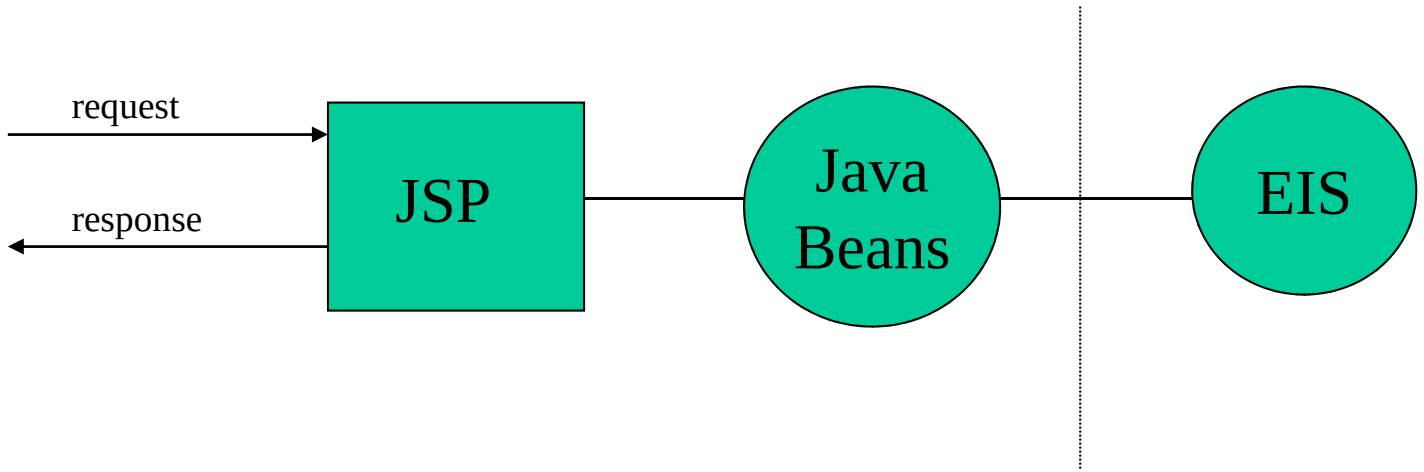
```
public void printHTMLReport(PrintWriter out, Results results) {
   println(out, "<b>Poll:</b> " + poll.getId() + "<br>");
   println(out, "<b>Question:</b> " + poll.getQuestion() + "<br>");
   out.print("<b>Candidates:</b> ");
   this.printCandidates(out);
   println(out, "<br>");
   println(out, "<br>");
   println(out, "<b>The winner is...</b> " + results.getWinner() + "!<br>");
   println(out, "<br><br>");
   println(out, "Round-by-round summary:<br><br>");
…
…
}
```

Problems with Pure Java Servlets

- Servlets have to take care of presentation - which is not a good thing for it to do
- HTML tags in the Java code
- Hard to maintain and extend (every content change requires the compilation of Java classes)
- Frustration of Java developers who are good at writing Java code but not HTML

Problems with JSPs + Java Beans (Model 1)

No controller - each page is responsible to direct control to the next page.

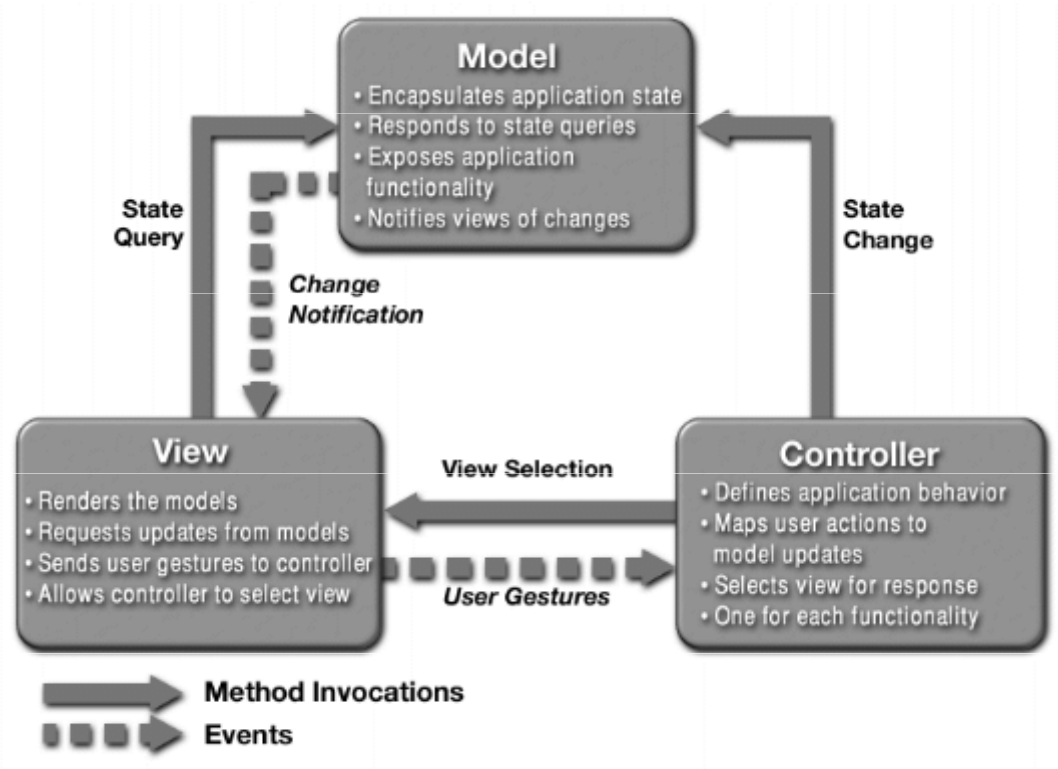


Problems with JSPs + Java Beans (Model 1)

- Java code in HTML tags
- Hard to maintain and change also
- Frustration of content developer

MVC and Model II

- Aim to separate the flow control, business logic and presentation

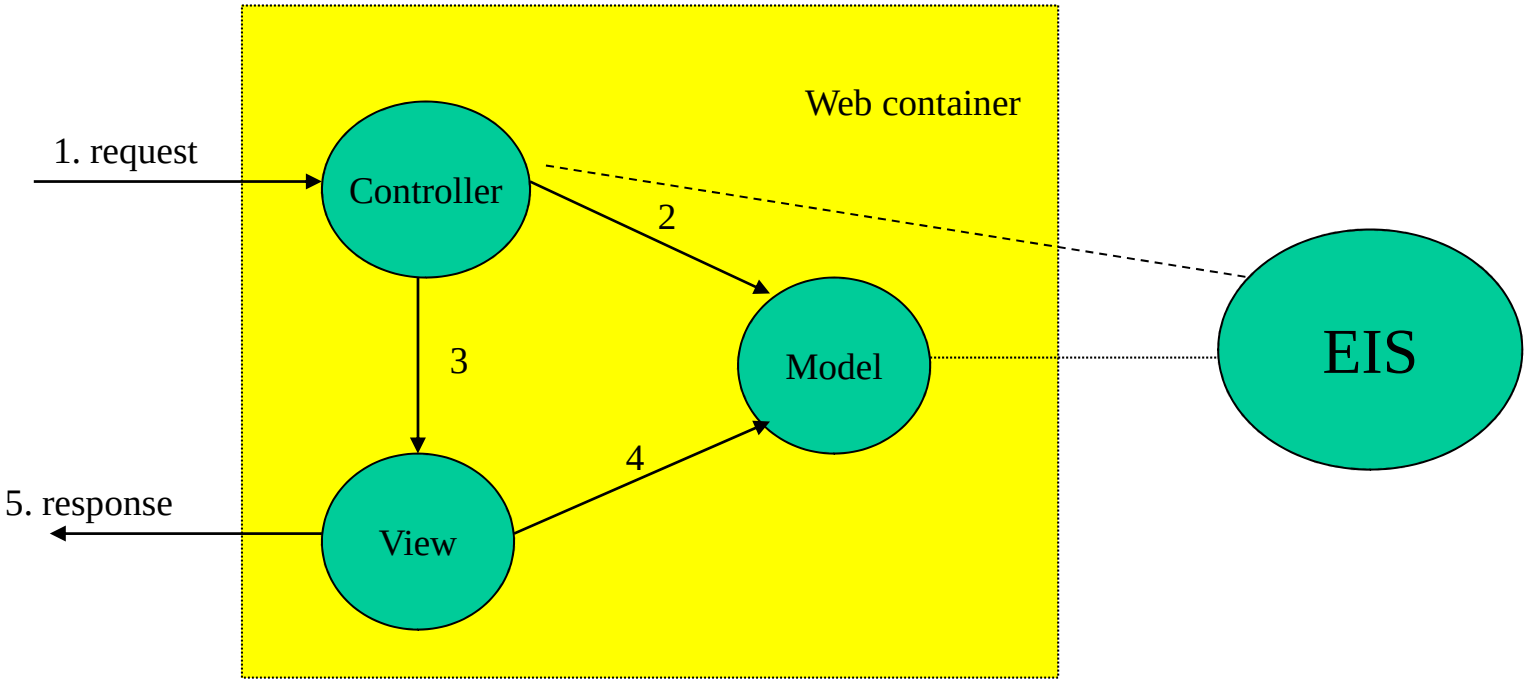


MVC and Model II

- **Model -** this is you data in the system. Such as a poll, the profile of a poll administrator.
- **View -** how data is represented to the user. For instance the view in a web application could be an HTML page, an image or other media.
- **Controller -** is the glue between the model and the view. It is responsible for controlling the flow as well as processing updates from the model to the view and visa versa

MVC and Model II

- Model II - uses a controller



MVC and Model II

- Controller - Java servelts
- Model - Java Beans (domain objects, value objects)
- View - JSP, XML + XSLT ….

MVC and Model II

- Many implementations available for choice
- Expresso
- Barracuda
- Cocoon
- WebWork
- Velocity, Tea, WebMacro
- ….

MVC and Model II

- And of course, you can build your own
- But, Why? Regardless how you implement, you probably will end up with something similar or close to Struts.

Break 1

A good scientist is a person with original ideas. A good engineer is a person who makes a design that works with as few original ideas as possible.

--- Freeman Dyson

Jakarta Struts - What is it?

- A flexible control layer based on standard technologies like Java Servlets, JavaBeans, ResourceBundles, and Extensible Markup Language (XML).
- An implementation of MVC Model II
- A standardized and extensible web application design/implementation framework

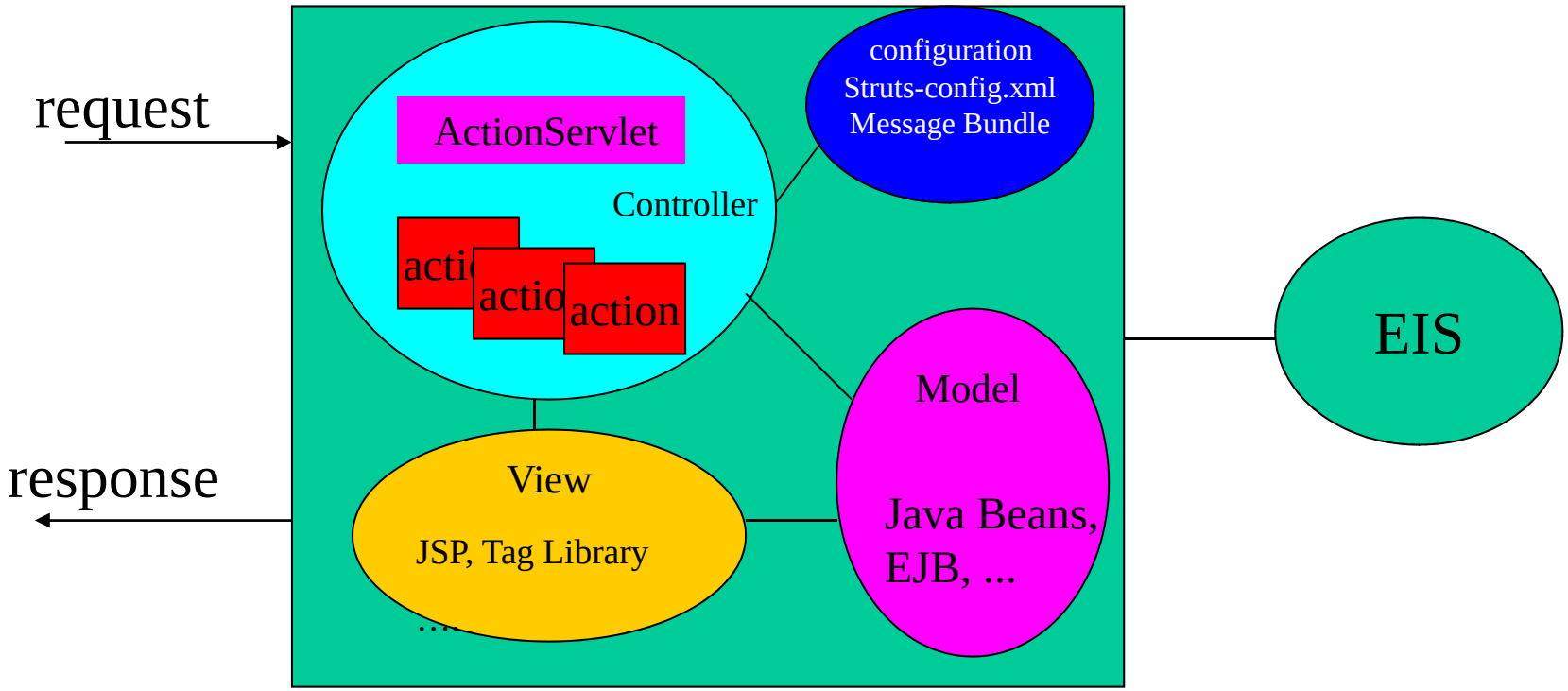
Jakarta Struts - General Info

- **Craig McClanahan** created the original framework and donated it to Apache Software Foundation (ASF) in 2000.
- Current stable release - 1.0.2.
- 1.1 is in beta release.
- The framework provides 9 package with 250 classes

Jakarta Struts - Features

- An Model II implementation of MVC
- Big base packages and classes ready to use
- Rich Custom Tag Libraries ready to use
- Easy support for internationalization (I18N)

Jakarta Struts - the whole picture



Jakarta Struts - the controller

- **ActionServlet** - Responsible for flow of control - 1. receive all requests, 2. route/delegate request to Action for handling, 3. return response
- One instance per web application
- Uses a configuration file called struts-config.xml to read mapping data, which specify control flow information
- Provided by the framework already. Ready to use but can also be extended

Jakarta Struts - the controller

- **Action -** the actual class which processes the request relevant to the business operation. Based on action mappings defined in an XML configuration file, the controller dispatches a call to the **perform** method of the appropriate Action Class.
- Command pattern
- Framework provides some actions. You need to implement you own action class for business purpose.

Jakarta Struts - the model

- You are responsible for creating it - Java Bean, EJB, etc.
- Should be reusable domain objects and value objects to represent a state.
- The framework works fine with any model implementation.

Jakarta Struts - the view

- JSP
- Custom Tag Library
- Resource Bundle

Jakarta Struts - the view

- Struts provides rich custom tag library to dramatically reduce Java code in your jsp.
- Example: HTML, BEAN, LOGIC, TEMPLATES,…
- We will see how handy they are in the case study

Jakarta Struts - the others

- ActionForm - encapsulate request/session data, typically used for input forms, provides a central point to validate form data.

Jakarta Struts - the others

- ActionMessage and ActionError

Jakarta Struts - the configuration

- Glue the system together
- web.xml and struts-config.xml

Jakarta Struts - the configuration

- Web.xml:
  - define the controller servlet (ActionServlet)with initial parameters;
  - define URL mapping to be handled by the controller servlet;
  - define the Struts tag libraries.

Jakarta Struts - the configuration

- struts-config.xml
  - Glue all components together and set the rules for the application
  - Data Source Configuration
  - Form Bean Definitions
  - Action Mapping Definitions
  - Global Forward Definitions

Break 2

A manager, an engineer, and a developer are going down a hill in a jeep when it suddenly loses control and goes into a ditch. They each attempt to offer a solution to the problem. The manager says 'Let's have a meeting to discuss a solution'. The engineer says 'Let's disassemble the jeep, and examine every part to find out where the problem is.' The developer says, 'Let's push it back up the hill and try again.

http://www.middleware-company.com/offer/patternsday1/article4.shtml

Case Study:
An Online Instant Poll

- Introduction - When I was having a hard time to define an appropriate application to really have my hands wet with Struts, I spotted Alex Chaffee's JiffyPoll (**http://www.purpletech.com/irv/index.html**). I like his idea but not all his implementation. Then I began to apply the struts framework to it and extends its functionality …

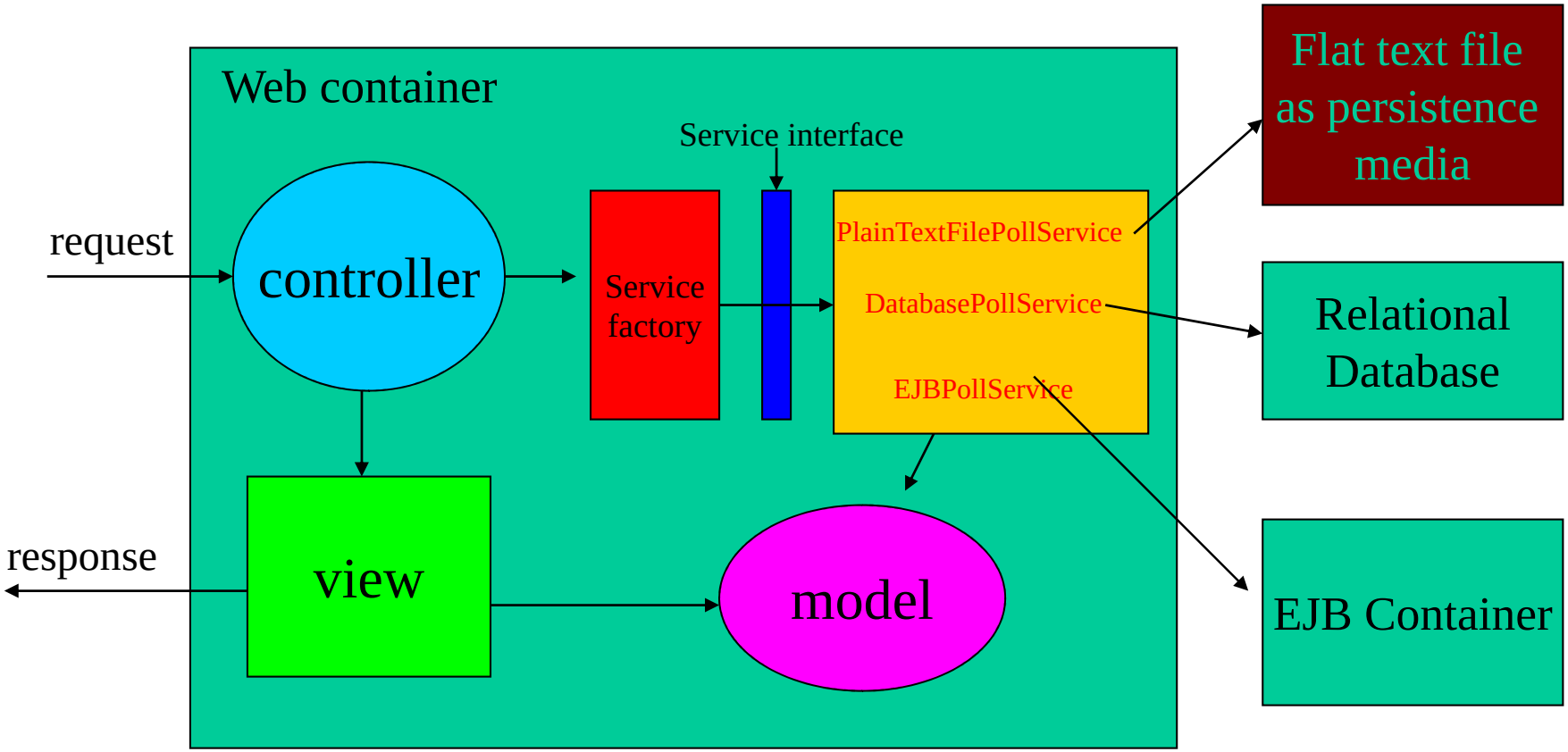
Case Study:
An Online Instant Poll

- What does it do?
  - As a normal user - you can select and vote a poll, you can view the results of any poll, you can also see the information about a poll.
  - As an administrator - you can create a new poll right away. You can also disable/enable a poll owned by you.

Case Study:
An Online Instant Poll

- What does it do - Let us try it.

Case Study:
An Online Instant Poll



Case Study:
An Online Instant Poll

- Benefits of the design
  - The real business logic is de-coupled from the controller, the model and the view
  - Increase of the reusability of the components - the service is completely reusable, the model (domain object and value object) is fully reusable also.

Case Study:
An Online Instant Poll

- The controller - directly use ActionServlet provided by the framework

- The action classes - delegate to service provider which does the real work.

Case Study:
An Online Instant Poll

- The real business logic - de-coupled from the framework and stands as a independent/reusable component

Case Study:
An Online Instant Poll

- The Model - reusable Java Beans, domain objects and value objects.

Case Study:
An Online Instant Poll

- The view - use the data provided in the model
- Rich use of tag library provide by the framework

- Show some sample JSPs and includes

- Almost no Java code in JSPs

Case Study:
An Online Instant Poll

- Internationalization - easy to be implemented like a breeze

- Demo

Jakarta Struts - Why it is good

- Promote clean separation of components

Jakarta Struts - Why it is good

- Increase the modularity, reusability of components

Jakarta Struts - Why it is good

- Focus on design

Jakarta Struts - Why it is good

- Very use case driven

Jakarta Struts - Why it is good

- What you need to do is clear
- You can really focus on the real work - the business logic

Jakarta Struts - Why it is good

- Easy internationalization

Jakarta Struts - Why it is good

- Free
- Open Source
- Implemented by savvy developer/expert
- large user community

New features in 1.1 - what makes it better

- Declarative exception handling

```
<action    path="/login"
        type="net.indyjug.weili.struts.security.LoginAction"
        name="loginForm"
        scope="request"
        validate="true"
        input="/jsp/login.jsp">
        <exception
            key="error.invalidlogin"
            path=”/jsp/login.jsp"
            scope="request"
            type=”net.indyjug.weili.struts.exception.InvalidLoginException"/>
        <forward name="success" path="/jsp/administration.jsp" />
        <forward name="failure" path="/jsp/login.jsp"  />
 </action>
```

New features in 1.1 - what makes it better

- Dynamic Form Bean

New features in 1.1 - what makes it better

- Plug-in

New features in 1.1 - what makes it better

- Tiles

The learning curve

- Not so easy for the first time. After that life is easy.
- Manageable
- Custom tag library may make mad at the beginning. Do not quit. They are so handy once you know how to use them.

Fast Track:
Getting Started

- What you need
  - JDK 1.2 and above
  - Web Contained compliant with Servlet API 2.2 and JSP API 1.1
  - An XML parser compliant with JAXP 1.1 or above (this is usually provided by the web container)

Fast Track:
Getting Started

- The framework provides an application template for you to use and extend
- Always continue on top of something which is working.
- Build the whole structure first and leave the concrete implementation later on (example)

Question?