Comparing Web Frameworks

JSF, Spring MVC, Stripes, Struts 2, Tapestry and Wicket

Matt Raible
matt@raibledesigns.com

Raible Designs

Today's Agenda

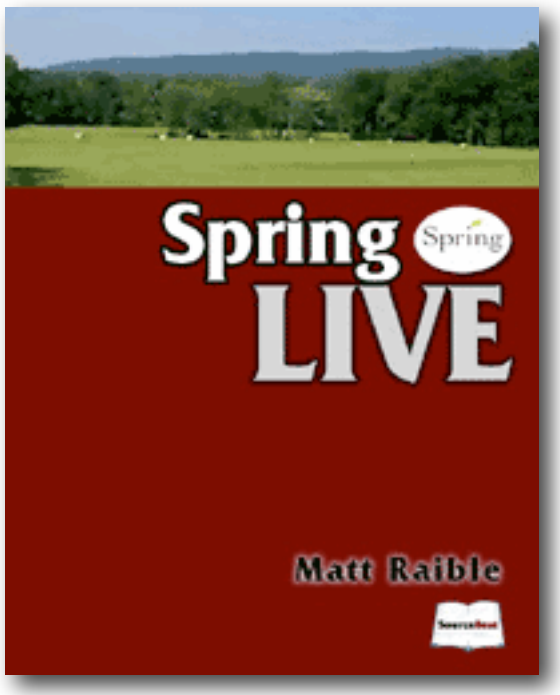
- Introductions
- Pros and Cons
- Sweetspots
- Web Framework Comparison: What each does well
- Conclusion
- Q and A

Introductions

- Your experience with webapps?
- Your experience with Java EE?
- What do you want to get from this session?
- Experience with Maven, Tomcat, Hibernate, Spring?
- Web Framework Experience:
  - Struts, Spring MVC, WebWork, Tapestry, JSF

Matt

Raible



AppFuse

Roller

the <display:*> tag library

Struts Menu

Java Community Process MEMBER

XDoclet
Attribute Oriented Programming

Struts 2

tapestry

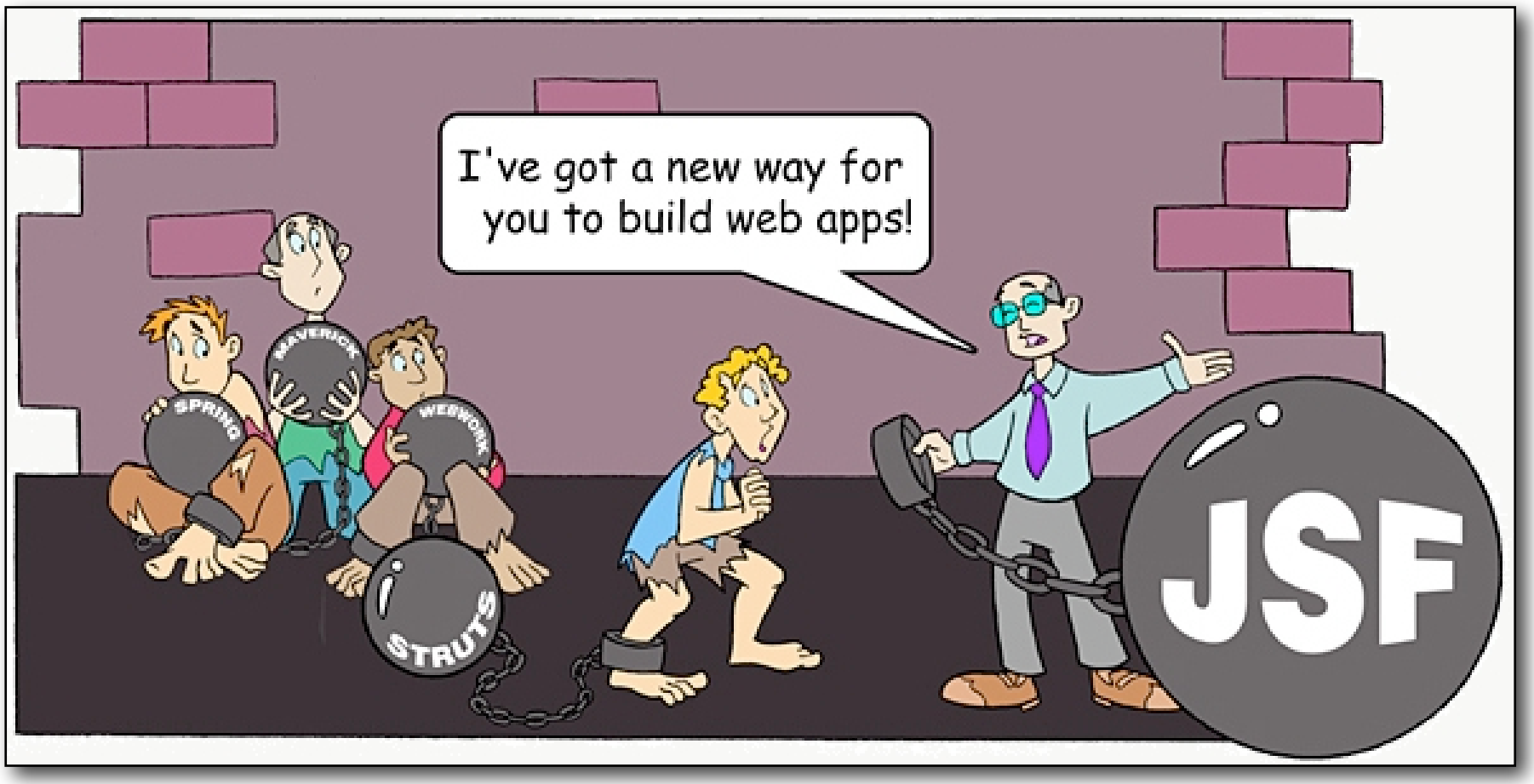
Spring 2.0

Stripes

My Experience

- **Struts 1**: used since June 2001 - same time 1.0 was released.
- **Spring MVC**: used since January 2004 - before 1.0 was released.
- **Struts 2**: used since July 2004.
- **Tapestry**: used since July 2004.
- **JSF**: used since July 2004 - both Sun's RI and MyFaces.
- **Stripes and Wicket**: learned them last week ;-)

Pros and Cons



JSF

- Pros:
  - Java EE Standard - lots of demand and jobs
  - Fast and easy to develop with initially
  - Lots of component libraries
- Cons:
  - Tag soup for JSPs
  - Doesn't play well with REST or Security
  - No single source for implementation

Spring MVC

- Pros:
  - Lifecyle for overriding binding, validation, etc.
  - Integrates with many view options seamlessly: JSP/JSTL, Tiles, Velocity, FreeMarker, Excel, XSL, PDF
  - Inversion of Control makes it easy to test
- Cons:
  - Configuration intensive - lots of XML
  - Almost too flexible - no common parent Controller
  - No built-in Ajax support

Struts 2

- Pros:
  - Simple architecture - easy to extend
  - Tag Library is easy to customize with FreeMarker or Velocity
  - Controller-based or page-based navigation
- Cons:
  - Documentation is poorly organized
  - Too much concentration on new features
  - Googling results in Struts 1.x documentation

Stripes

- Pros:
  - No XML - Convention over Configuration
  - Good documentation (easy to learn)
  - Enthusiastic community
- Cons:
  - Small Community
  - Not as actively developed as other projects
  - Hard-coded URLs in ActionBeans

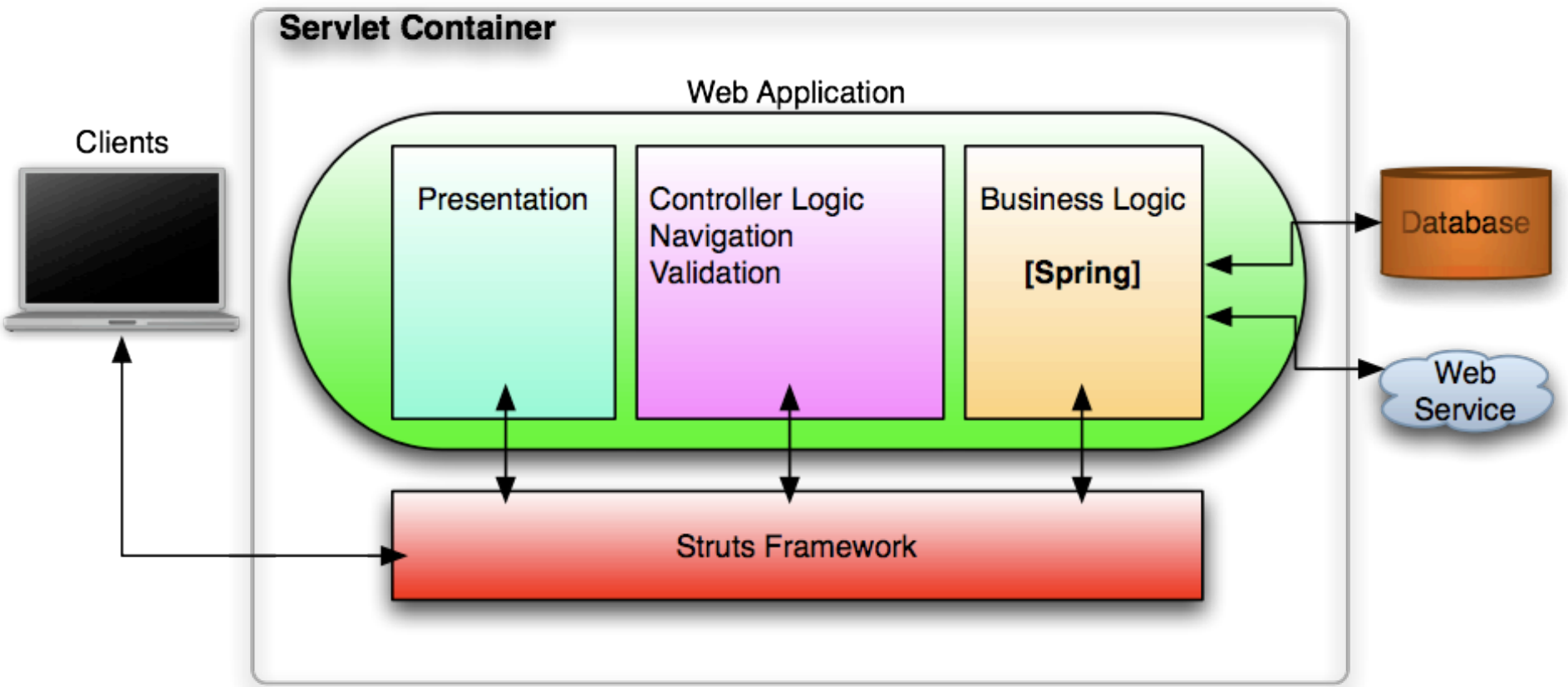
Tapestry

- Pros:
  - Very productive once you learn it
  - Templates are HTML - great for designers
  - Lots of innovation between releases
- Cons:
  - Documentation very conceptual, rather than pragmatic
  - Steep learning curve
  - Long release cycles - major upgrades every year

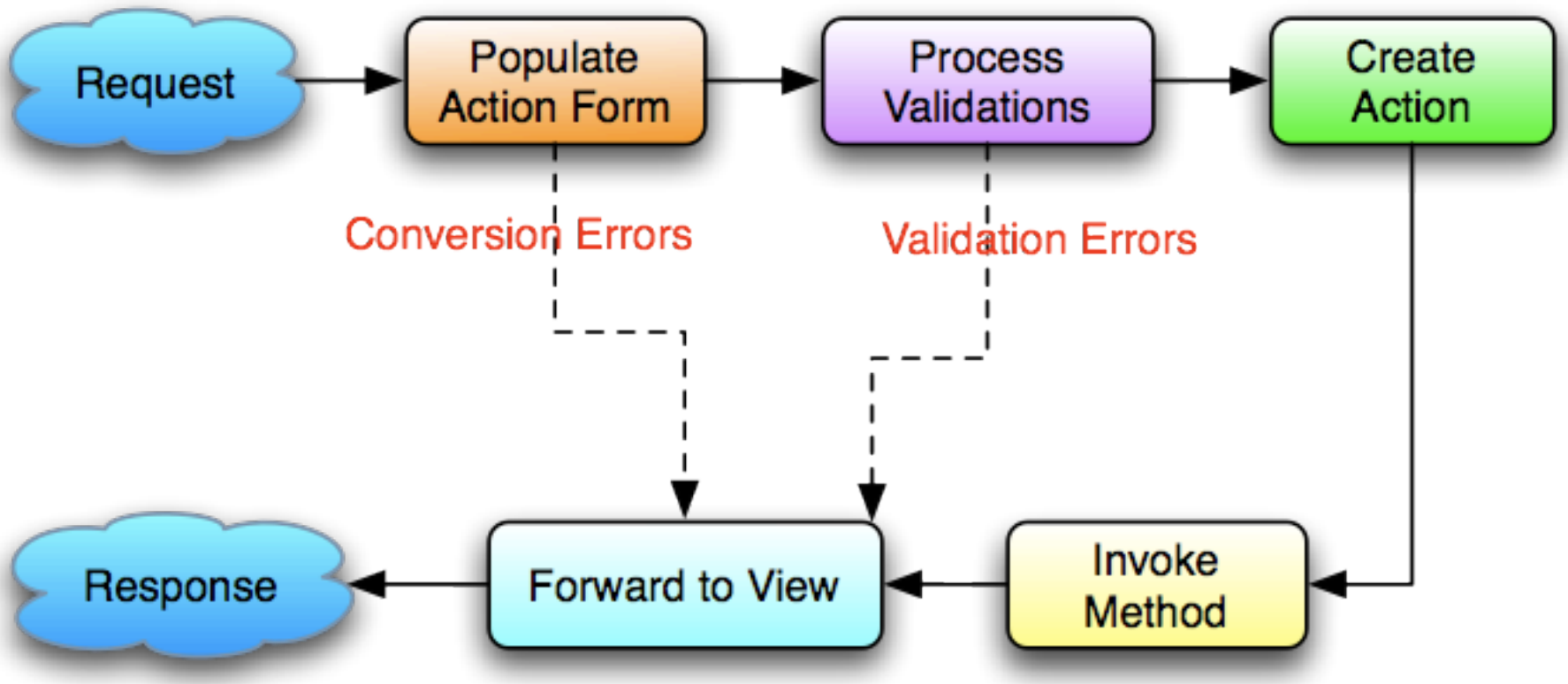
Wicket

- Pros:
  - Great for Java developers, not web developers
  - Tight binding between pages and views
  - Active community - support from the creators
- Cons:
  - HTML templates live next to Java code
  - Need to have a good grasp of OO
  - The Wicket Way - everything done in Java

Struts



Struts Lifecycle



Struts Action

```
public class UserAction extends DispatchAction {
    private UserManager mgr = null;

    public void setUserManager(UserManager userManager) {
        this.mgr = userManager;
    }

    public ActionForward delete(ActionMapping mapping, ActionForm form,
                                HttpServletRequest request,
                                HttpServletResponse response)
            throws Exception {
        DynaActionForm userForm = (DynaActionForm) form;
        User user = (User) userForm.get("user");

        mgr.removeUser(request.getParameter("user.id"));

        ActionMessages messages = new ActionMessages();
        messages.add(ActionMessages.GLOBAL_MESSAGE,
                new ActionMessage("user.deleted", user.getFullName()));

        saveMessages(request.getSession(), messages);

        return mapping.findForward("users");
    }
```

Struts Action

```
public class UserAction extends DispatchAction {
    private UserManager mgr = null;

    public void setUserManager(UserManager userManager) {
        this.mgr = userManager;
    }

    public ActionForward delete(ActionMapping mapping, ActionForm form,
                                HttpServletRequest request,
                                HttpServletResponse response)
            throws Exception {
        DynaActionForm userForm = (DynaActionForm) form;
        User user = (User) userForm.get("user");

        mgr.removeUser(request.getParameter("user.id"));

        ActionMessages messages = new ActionMessages();
        messages.add(ActionMessages.GLOBAL_MESSAGE,
                new ActionMessage("user.deleted", user.getFullName()));

        saveMessages(request.getSession(), messages);

        return mapping.findForward("users");
    }
```

Struts Action

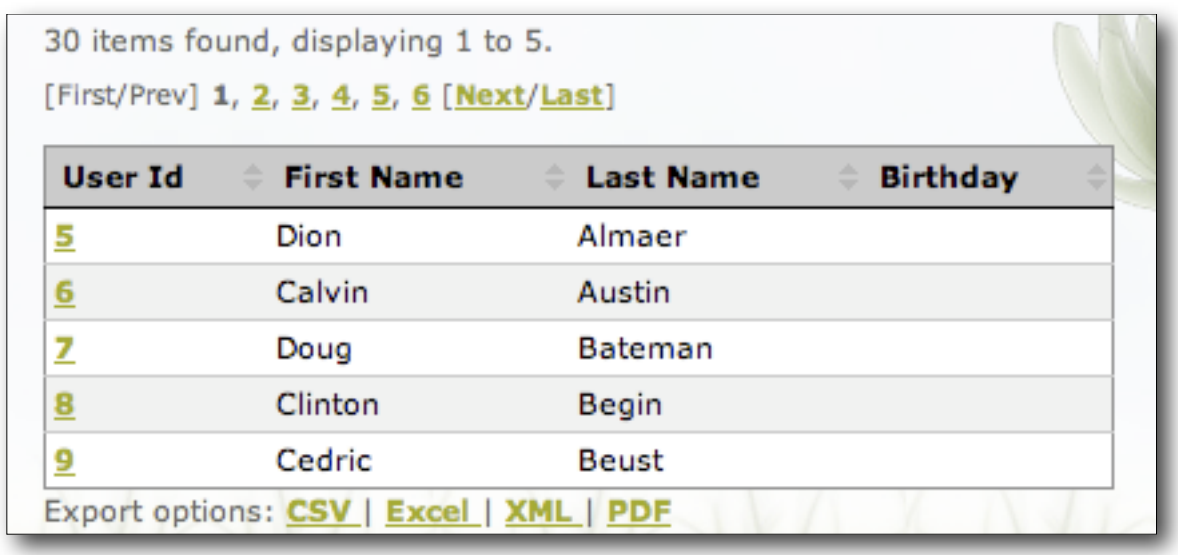
```
public class UserAction extends DispatchAction {
    private UserManager mgr = null;

    public void setUserManager(UserManager userManager) {
        this.mgr = userManager;
    }

    public ActionForward delete(ActionMapping mapping, ActionForm form,
                                HttpServletRequest request,
                                HttpServletResponse response)
            throws Exception {
        DynaActionForm userForm = (DynaActionForm) form;
        User user = (User) userForm.get("user");

        mgr.removeUser(request.getParameter("user.id"));

        ActionMessages messages = new ActionMessages();
        messages.add(ActionMessages.GLOBAL_MESSAGE,
                new ActionMessage("user.deleted", user.getFullName()));

        saveMessages(request.getSession(), messages);

        return mapping.findForward("users");
    }
```

struts-config.xml

```
<form-bean name="userForm" type="org.apache.struts.validator.DynaValidatorForm">
    <form-property name="user" type="org.appfuse.model.User"/>
</form-bean>


<action path="/user" type="org.springframework.web.struts.DelegatingActionProxy"
    name="userForm" scope="request" parameter="method" validate="false">
    <forward name="list" path="/userList.jsp"/>
    <forward name="edit" path="/userForm.jsp"/>
</action>


<plug-in className="org.springframework.web.struts.ContextLoaderPlugIn">
    <set-property property="contextConfigLocation"
                  value="/WEB-INF/action-servlet.xml"/>
</plug-in>
```

Display Tag

```
<display:table name="users" class="list" requestURI="" id="userList" export="true">
    <display:column property="id" sort="true" href="editUser.html"
        paramId="id" paramProperty="id" titleKey="user.id"/>
    <display:column property="firstName" sort="true" titleKey="user.firstName"/>
    <display:column property="lastName" sort="true" titleKey="user.lastName"/>
    <display:column titleKey="user.birthday" sort="true" sortProperty="birthday">
        <fmt:formatDate value="${userList.birthday}" pattern="${datePattern}"/>
    </display:column>
</display:table>
```



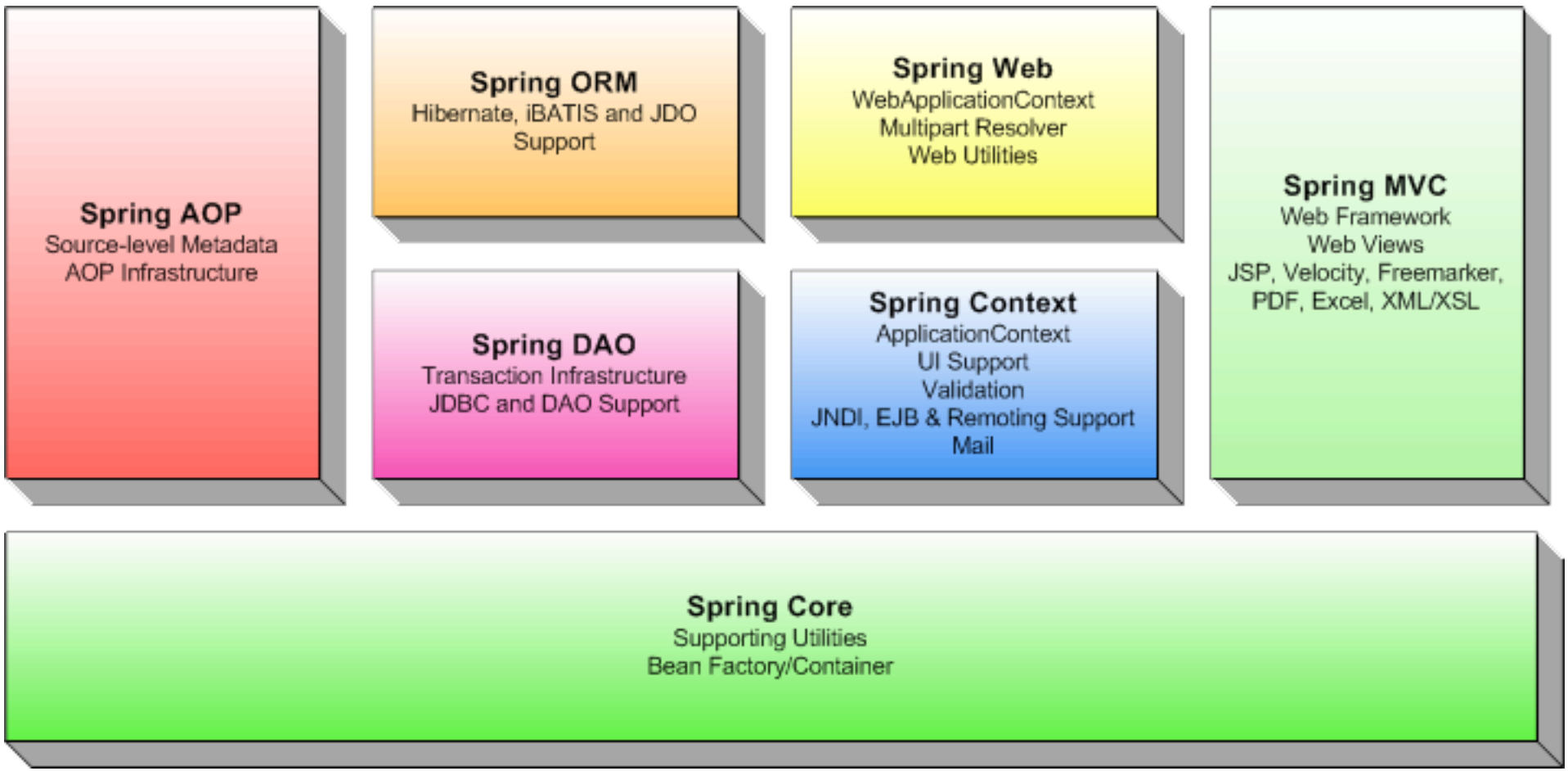
Struts: JSP View

```
<html:form action="/user" focus="user.firstName" onsubmit="return validateUserForm(this)">
<input type="hidden" name="method" value="save"/>
<html:hidden property="user.id"/>
<table class="detail">
<tr>
    <th><label for="user.firstName"><fmt:message key="user.firstName"/>:</label></th>
    <td><html:text property="user.firstName" styleId="user.firstName"/></td>
</tr>
<tr>
    <th><label for="user.lastName"><fmt:message key="user.lastName"/>:</label></th>
    <td>
        <html:text property="user.lastName" styleId="user.lastName"/>
        <span class="fieldError"><html:errors property="user.lastName"/></span>
    </td>
</tr>
    <th><label for="user.birthday"><fmt:message key="user.birthday"/>:</label></th>
    <td>
        <c:set var="datePattern"><fmt:message key="date.format"/></c:set>
        <input type="text" size="11" name="user.birthday" id="user.birthday"
            value="<fmt:formatDate value="${userForm.map.user.birthday}"
            pattern="${datePattern}"/>"/> [${datePattern}]
    </td>
<tr>
```

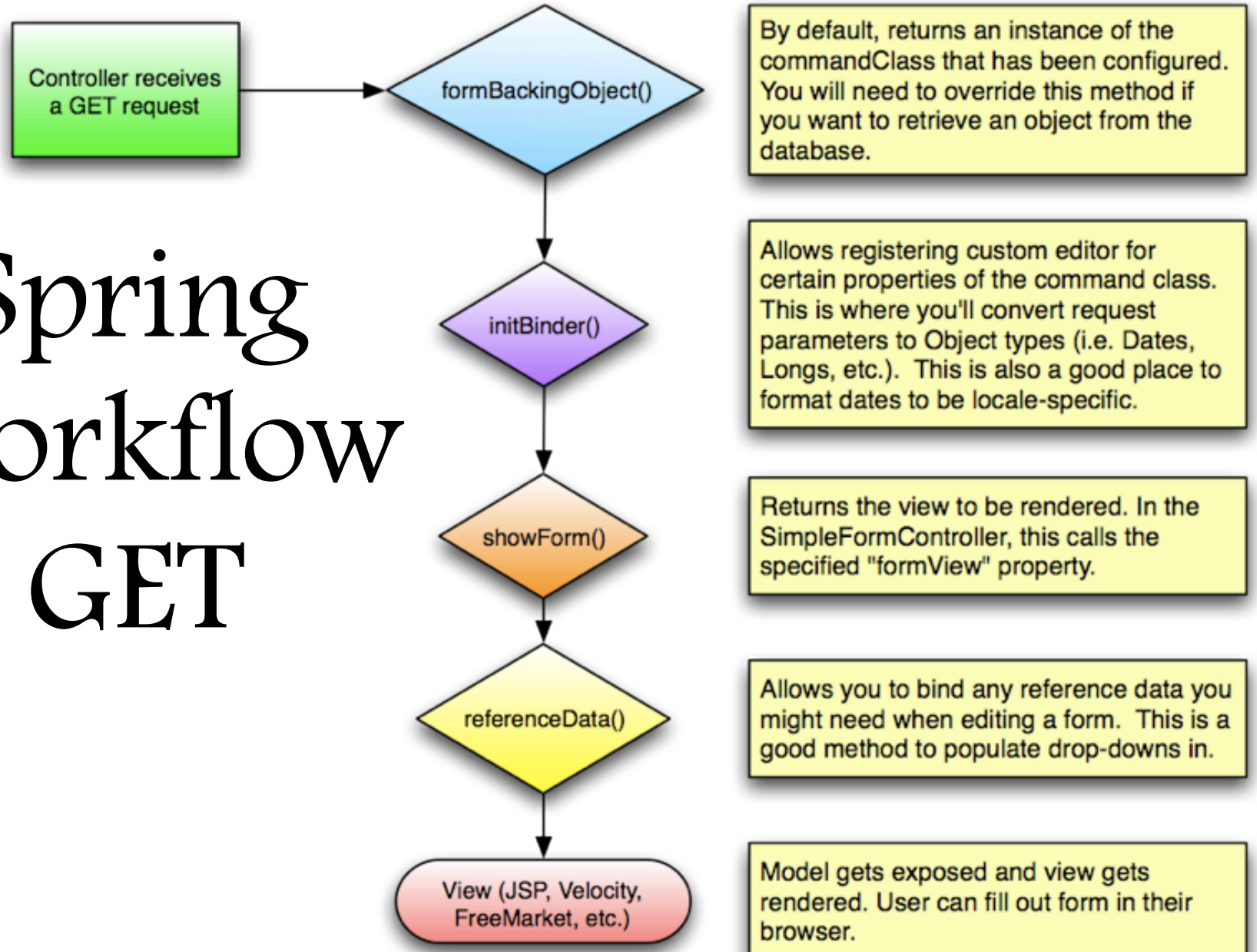
Struts: JSP View

```
<html:form action="/user" focus="user.firstName" onsubmit="return validateUserForm(this)">
<input type="hidden" name="method" value="save"/>
<html:hidden property="user.id"/>
<table class="detail">
<tr>
    <th><label for="user.firstName"><fmt:message key="user.firstName"/>:</label></th>
    <td><html:text property="user.firstName" styleId="user.firstName"/></td>
</tr>
<tr>
    <th><label for="user.lastName"><fmt:message key="user.lastName"/>:</label></th>
    <td>
        <html:text property="user.lastName" styleId="user.lastName"/>
        <span class="fieldError"><html:errors property="user.lastName"/></span>
    </td>
</tr>
    <th><label for="user.birthday"><fmt:message key="user.birthday"/>:</label></th>
    <td>
        <c:set var="datePattern"><fmt:message key="date.format"/></c:set>
        <input type="text" size="11" name="user.birthday" id="user.birthday"
            value="<fmt:formatDate value="${userForm.map.user.birthday}"
            pattern="${datePattern}"/>"/> [${datePattern}]
    </td>
<tr>
```

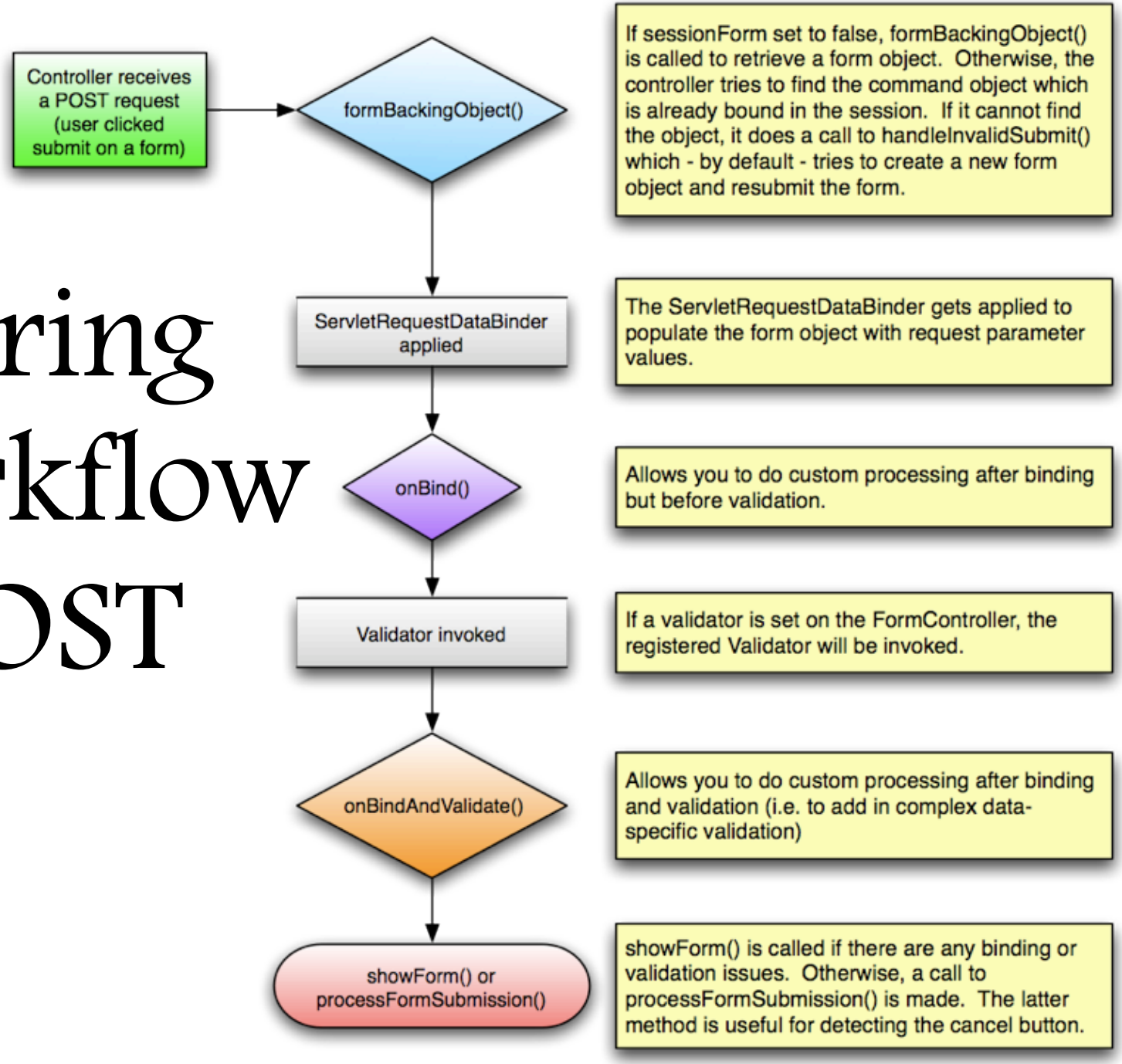
Spring MVC



Spring Workflow GET



Spring Workflow POST



Spring Controller

```
public class UserController implements Controller {
    private final Log log = LogFactory.getLog(UserController.class);
    private UserManager mgr = null;

    public void setUserManager(UserManager userManager) {
        this.mgr = userManager;
    }

    public ModelAndView handleRequest(HttpServletRequest request,
                                      HttpServletResponse response)
    throws Exception {
        if (log.isDebugEnabled()) {
            log.debug("entering 'handleRequest' method...");
        }

        return new ModelAndView("userList", "users", mgr.getUsers());
    }
}
```

Spring Controller

```
public class UserController implements Controller {
    private final Log log = LogFactory.getLog(UserController.class);
    private UserManager mgr = null;

    public void setUserManager(UserManager userManager) {
        this.mgr = userManager;
    }

    public ModelAndView handleRequest(HttpServletRequest request,
                                      HttpServletResponse response)
    throws Exception {
        if (log.isDebugEnabled()) {
            log.debug("entering 'handleRequest' method...");
        }

        return new ModelAndView("userList", "users", mgr.getUsers());
    }
}
```

Spring: Configuration

```
<bean id="userController" class="org.appfuse.web.UserController">
    <property name="userManager" ref="userManager"/>
</bean>


<bean id="viewResolver"
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="viewClass" value="org.springframework.web.servlet.view.JstlView"/>
    <property name="prefix" value="/"/>
    <property name="suffix" value=".jsp"/>
 </bean>


<bean id="urlMapping"
    class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
    <property name="mappings">
        <value>
            /users.html=userController
        </value>
    </property>
</bean>
```

Spring JSP View

```
<form:form commandName="user" method="post">
<form:errors path="*" cssClass="error"/>
<form:hidden path="id" />
<table class="detail">
<tr>
    <th><label for="firstName">
        <fmt:message key="user.firstName"/>:</label></th>
    <td>
        <form:input path="firstName" id="firstName"/>
        <form:errors path="firstName" cssClass="fieldError"/>
    </td>
</tr>
<tr>
    <th><label for="lastName" class="required">
        * <fmt:message key="user.lastName"/>:</label></th>
    <td>
        <form:input path="lastName" id="firstName"/>
        <form:errors path="lastName" cssClass="fieldError"/>
    </td>
</tr>
```

Spring JSP View

```
<form:form commandName="user" method="post">
<form:errors path="*" cssClass="error"/>
<form:hidden path="id" />
<table class="detail">
<tr>
    <th><label for="firstName">
        <fmt:message key="user.firstName"/>:</label></th>
    <td>
        <form:input path="firstName" id="firstName"/>
        <form:errors path="firstName" cssClass="fieldError"/>
    </td>
</tr>
<tr>
    <th><label for="lastName" class="required">
        * <fmt:message key="user.lastName"/>:</label></th>
    <td>
        <form:input path="lastName" id="firstName"/>
        <form:errors path="lastName" cssClass="fieldError"/>
    </td>
</tr>
```

Spring FreeMarker View

```
<form method="post" action="<@spring.url '/editUser.html'/>">
<@spring.formHiddenInput "user.id"/>
<table>
<tr>
    <th><label for="firstName"><@spring.message "user.firstName"/>:</label></th>
    <td>
        <@spring.formInput "user.firstName", 'id="firstName"'/>
        <@spring.showErrors "<br>", "fieldError"/>
    </td>
</tr>
<tr>
    <th><label for="lastName"><@spring.message "user.lastName"/>:</label></th>
    <td>
        <@spring.formInput "user.lastName", 'id="lastName"'/>
        <@spring.showErrors "<br>", "fieldError"/>
    </td>
</tr>
```

Spring Velocity View

```
<form method="post" action="#springUrl('/editUser.html')">
#springFormHiddenInput("user.id" '')
<table>
<tr>
    <th><label for="firstName">#springMessage("user.firstName"):</label></th>
    <td>
        #springFormInput("user.firstName" 'id="firstName"')
        #springShowErrors("<br/>" "fieldError")
    </td>
</tr>
<tr>
    <th><label for="lastName">#springMessage("user.lastName"):</label></th>
    <td>
        #springFormInput("user.lastName" 'id="lastName"')
        #springShowErrors("<br/>" "fieldError")
    </td>
</tr>
```

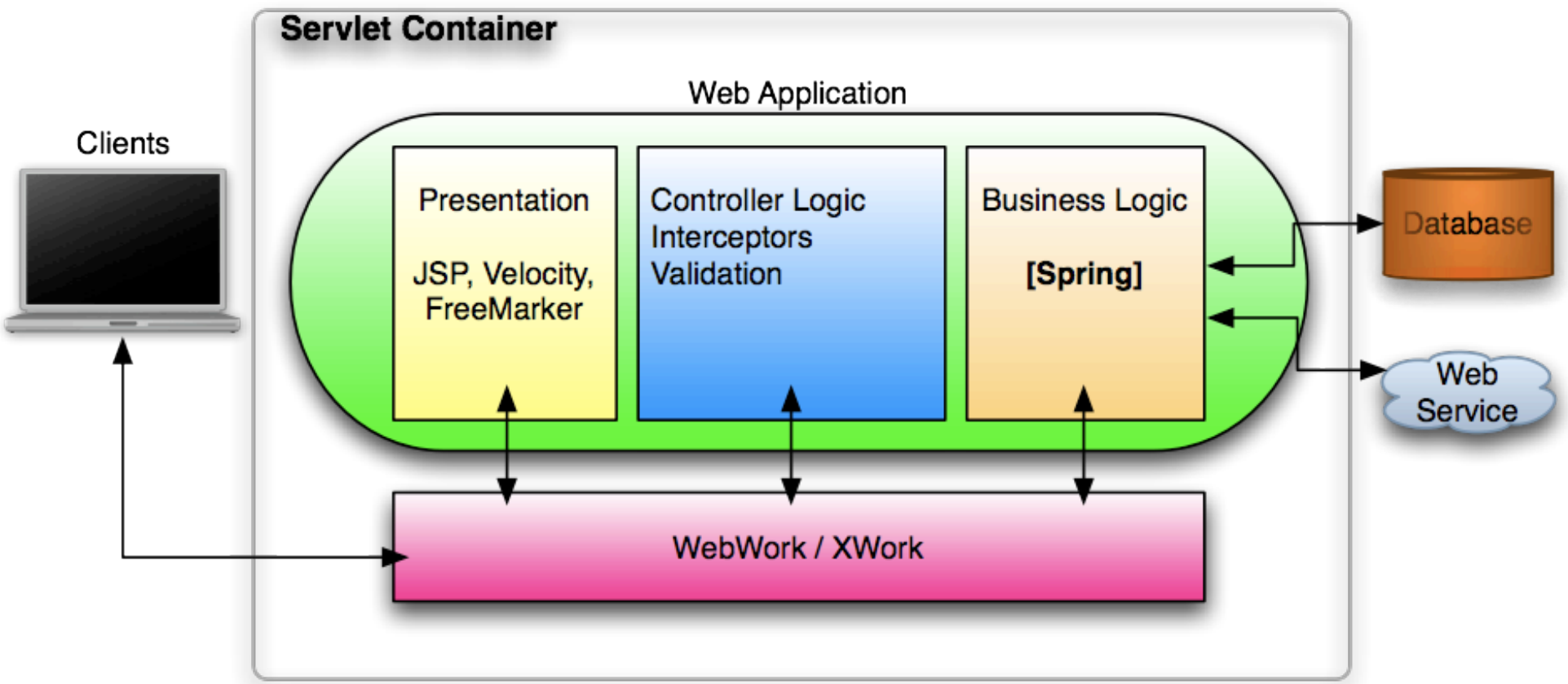
Spring Web Flow

- A web framework that allows you to define *page flows* in your application
- In XML (or programmatically in Java), you specify simple or complex navigation rules
- Navigation rules enforced based on the String value that a method returns (similar to JSF)
- Can call middle-tier methods declaratively - no controllers needed!

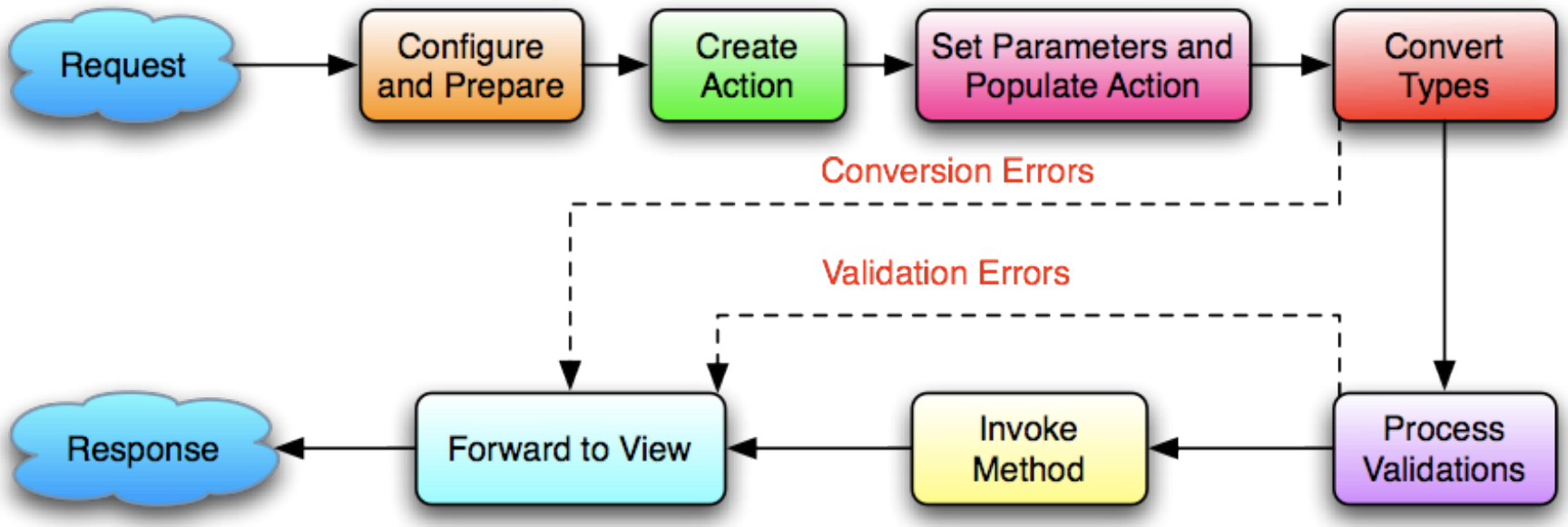
Spring Web Flow

```
<webflow id="userFlow" start-state="setupForm">

    <action-state id="setupForm">
        <action bean="userFormAction"/>
        <transition on="success" to="display.nameForm"/>
    </action-state>

    <view-state id="display.nameForm" view="flow/name">
        <transition on="submit" to="display.addressForm">
            <action bean="userFormAction" method="bindAndValidate"/>
        </transition>
        <transition on="cancel" to="finish"/>
    </view-state>

    <view-state id="display.addressForm" view="flow/address">
        <transition on="previous" to="display.nameForm">
            <action bean="userFormAction" method="bindAndValidate"/>
        </transition>
        <transition on="submit" to="display.otherForm">
            <action bean="userFormAction" method="bindAndValidate"/>
        </transition>
        <transition on="cancel" to="finish"/>
    </view-state>
```

WebWork



WebWork Lifecycle



WebWork Action

```
public class UserAction extends ActionSupport {
    private UserManager mgr;
    private User user;
    private String id;

    public void setUserManager(UserManager userManager) {
        this.mgr = userManager;
    }

    public void setId(String id) {
        this.id = id;
    }

    public User getUser() {
        return user;
    }

    public String edit() {
        // check for an add
        if (id != null) {
            user = mgr.getUser(id);
        } else {
            user = new User();
        }
        return SUCCESS;
    }
```

WebWork Interceptors

```
public class ValidationInterceptor extends AroundInterceptor {

    protected void after(ActionInvocation dispatcher, String result) throws Exception {
    }

    protected void before(ActionInvocation invocation) throws Exception {
        Action action = invocation.getAction();
        String context = invocation.getProxy().getActionName();

        final Map parameters = ActionContext.getContext().getParameters();
        // don't validate on cancel, delete or GET
        if (ServletActionContext.getRequest().getMethod().equals("GET")) {
            log.debug("Cancelling validation, detected GET request");
        } else if (parameters.containsKey("cancel") || parameters.containsKey("delete")) {
            log.debug("Cancelling validation, detected clicking cancel or delete");
        } else {
            ActionValidatorManager.validate(action, context);
        }
    }
}
```

xwork.xml

```
<!-- List of Users -->
<action name="users" class="userAction" method="list">
    <result name="success">userList.jsp</result>
    <result name="input">userList.jsp</result>
</action>

<!-- Edit User -->
<action name="editUser" class="userAction" method="edit">
    <result name="success">userForm.jsp</result>
    <result name="input">userList.jsp</result>
</action>

<!-- Save User -->
<action name="saveUser" class="userAction">
    <result name="cancel" type="redirect">users.html</result>
    <result name="delete" type="redirect">users.html</result>
    <result name="input">userForm.jsp</result>
    <result name="success" type="chain">saveUserWithValidation</result>
</action>
```

WebWork JSP View

```
<ww:form name="userForm" action="saveUser" method="post" validate="true">
    <ww:hidden name="user.id" value="%{user.id}"/>

    <ww:textfield label="%{getText('user.firstName')}" name="user.firstName"
        value="%{user.firstName}" id="user.firstName"/>

    <ww:textfield label="%{getText('user.lastName')}" name="user.lastName"
        value="%{user.lastName}" required="true"/>

    <ww:datepicker label="%{getText('user.birthday')}" name="user.birthday"
        size="11"/>
```

WebWork DatePicker


```
<ww:form name                                    e="true">
    <ww:hidde

    <ww:textf                                    r.firstName"
        value

    <ww:textf                                    .lastName"
        value

    <ww:datep                                    r.birthday"
        size=
```

Page-based Navigation

```
<%@ include file="/common/taglibs.jsp"%>

<h2>Author Blogs</h2>

<ww:action name="authors" id="%{authors}" namespace="default"/>

<div class="item">
    <ww:iterator value="#authors.authors" status="index">
        <a href="<ww:property value="blog.feedUrl"/>">
            <img src="${ctxPath}/images/icons/xml.gif" alt="XML Feed"/></a>
        <a href="<ww:property value="blog.url"/>"><ww:property value="name"/></a>
        <br />
    </ww:iterator>
</div>
```

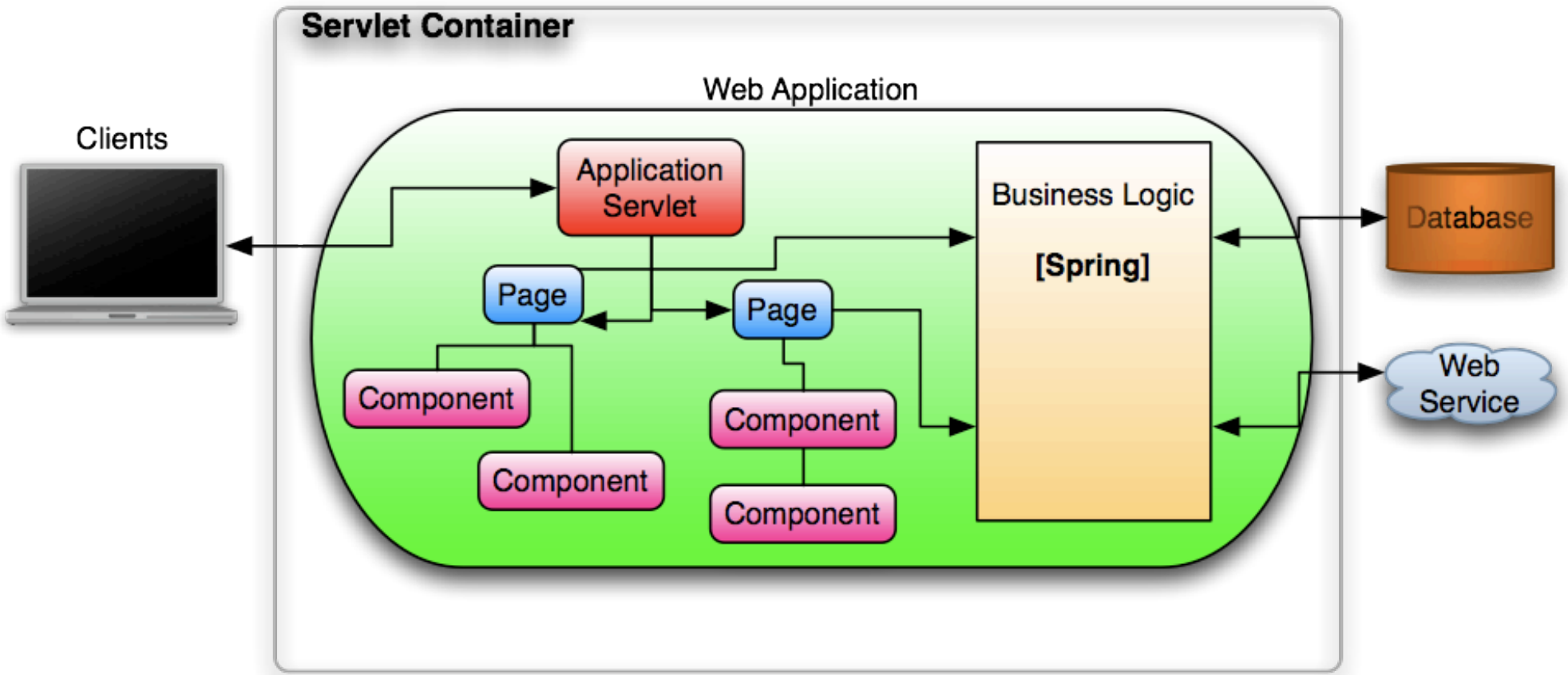
OGNL

```
<ww:form name="userForm" action="saveUser" method="post" validate="true">
    <ww:hidden name="user.id" value="%{user.id}"/>

    <ww:textfield label="%{getText('user.firstName')}" name="user.firstName"
        value="%{user.firstName}" id="user.firstName"/>

    <ww:textfield label="%{getText('user.lastName')}" name="user.lastName"
        value="%{user.lastName}" required="true"/>
</tr>
    <th><label for="user.birthday"><fmt:message key="user.birthday"/>:</label></th>
    <td>
        <ww:set name="birthday" scope="request"
            value="(user.birthday instanceof java.util.Date) ? user.birthday : ''"/>
        <input type="text" size="11" name="user.birthday" id="user.birthday"
            value="<fmt:formatDate value="${birthday}" pattern="${datePattern}"/>"/>
            [${datePattern}]
    </td>
<tr>
```

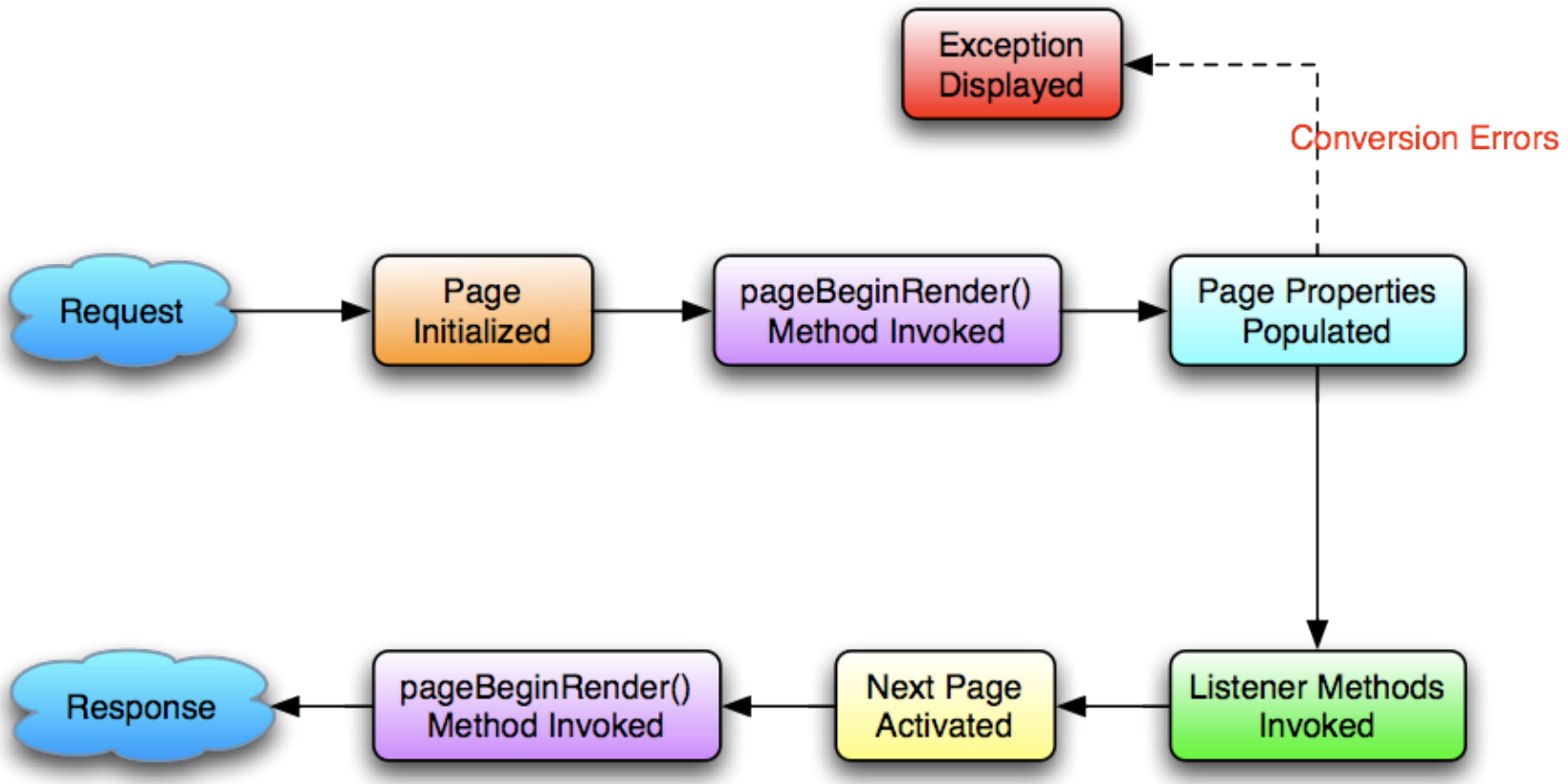
New in WebWork 2.2

- Built-in Ajax Support: DWR and Dojo
- Spring as default inversion of control container
- Changed from front-controller servlet to filter
- Much better client-side validation support
- QuickStart and Annotations
- Soon to be Struts Action 2.0

Tapestry



Tapestry Lifecycle



Tapestry Page

```
public abstract class UserForm extends BasePage {

    public abstract UserManager getUserManager();
    public abstract void setUser(User user);
    public abstract User getUser();

    public void save(IRequestCycle cycle) {
        if (log.isDebugEnabled()) {
            log.debug("entered 'save' method");
        }

        getUserManager().saveUser(getUser());

        UserList nextPage = (UserList) cycle.getPage("users");
        nextPage.setMessage(getMessages().format("user.saved",
                                getUser().getFullName()));
        throw new PageRedirectException(nextPage);
    }
```

Tapestry Configuration

```
<application name="tapestry">
    <page name="Home" specification-path="/pages/home.page"/>
    <page name="users" specification-path="/pages/users.page"/>
    <page name="userForm" specification-path="/pages/userForm.page"/>

    <library id="contrib"
        specification-path="/org/apache/tapestry/contrib/Contrib.library"/>
</application>
```

Page Configuration

```
<page-specification class="org.appfuse.web.UserForm">
    <bean name="delegate" class="org.apache.tapestry.valid.ValidationDelegate"/>

    <component id="form" type="Form">
        <binding name="delegate" value="ognl:beans.delegate"/>
        <binding name="clientValidationEnabled" value="true"/>
    </component>

    <property name="user"/>
    <inject property="userManager" object="spring:userManager"/>

    <component id="lastNameField" type="TextField">
        <binding name="value" value="user.lastName"/>
        <binding name="validators" value="validators:required"/>
        <binding name="displayName" value="message:lastName"/>
    </component>

</page-specification>
```

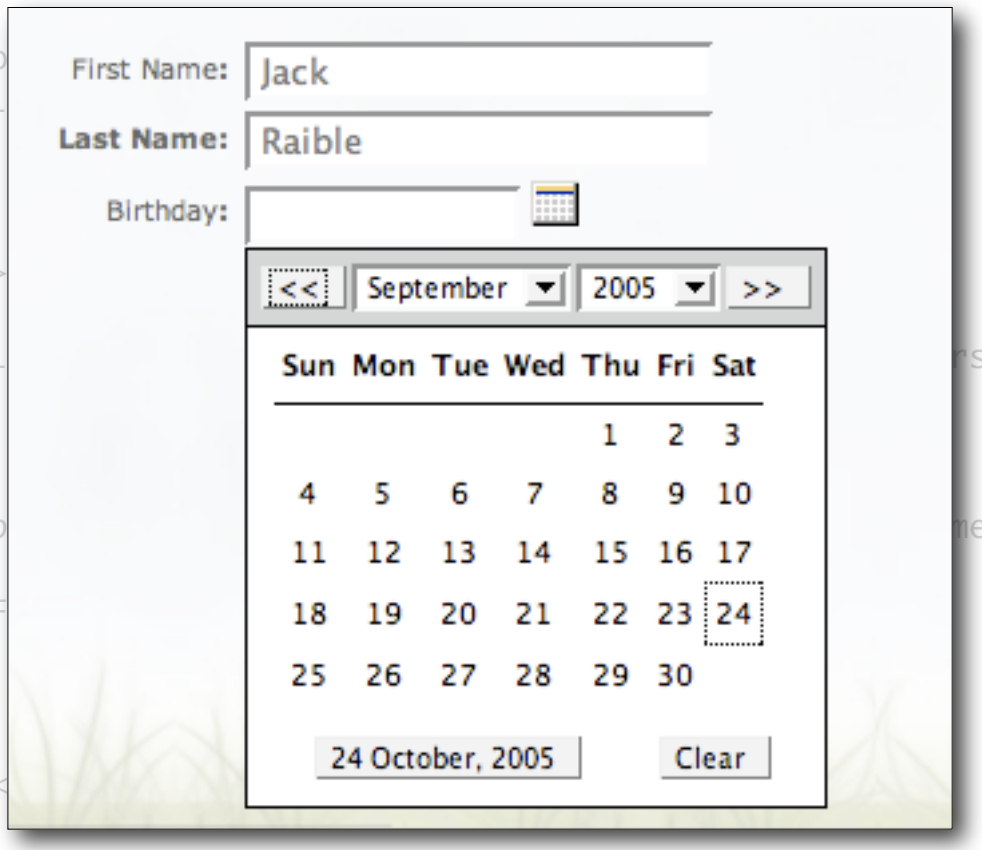
Tapestry HTML View

```
<form jwcid="@Form" delegate="ognl:beans.delegate" name="userForm">
<input type="hidden" jwcid="@Hidden" value="ognl:user.id"/>
<table class="detail">
<tr>
    <th>
        <label for="firstName"><span key="firstName">First Name</span></label>:
    </th>
    <td><input jwcid="@TextField" type="text" value="ognl:user.firstName" id="firstName"/></td>
</tr>
<tr>
    <th>
        <label jwcid="@FieldLabel" field="ognl:components.lastNameField">Last Name</label>:
    </th>
    <td><input jwcid="lastNameField" type="text" id="lastName"/></td>
</tr>
<tr>
    <th>
        <label for="birthday"><span key="birthday">Birthday</span></label>:
    </th>
    <td>
        <input jwcid="@DatePicker" format="message:date.format" type="text"
            size="11" value="ognl:user.birthday" id="birthday"/>
    </td>
</tr>
```

Tapestry HTML View

```
<form jwcid="@Form" delegate="ognl:beans.delegate" name="userForm">
<input type="hidden" jwcid="@Hidden" value="ognl:user.id"/>
<table class="detail">
<tr>
    <th>
        <label for="firstName"><span key="firstName">First Name</span></label>:
    </th>
    <td><input jwcid="@TextField" type="text" value="ognl:user.firstName" id="firstName"/></td>
</tr>
<tr>
    <th>
        <label jwcid="@FieldLabel" field="ognl:components.lastNameField">Last Name</label>:
    </th>
    <td><input jwcid="lastNameField" type="text" id="lastName"/></td>
</tr>
<tr>
    <th>
        <label for="birthday"><span key="birthday">Birthday</span></label>:
    </th>
    <td>
        <input jwcid="@DatePicker" format="message:date.format" type="text"
            size="11" value="ognl:user.birthday" id="birthday"/>
    </td>
</tr>
```

DatePicker Component

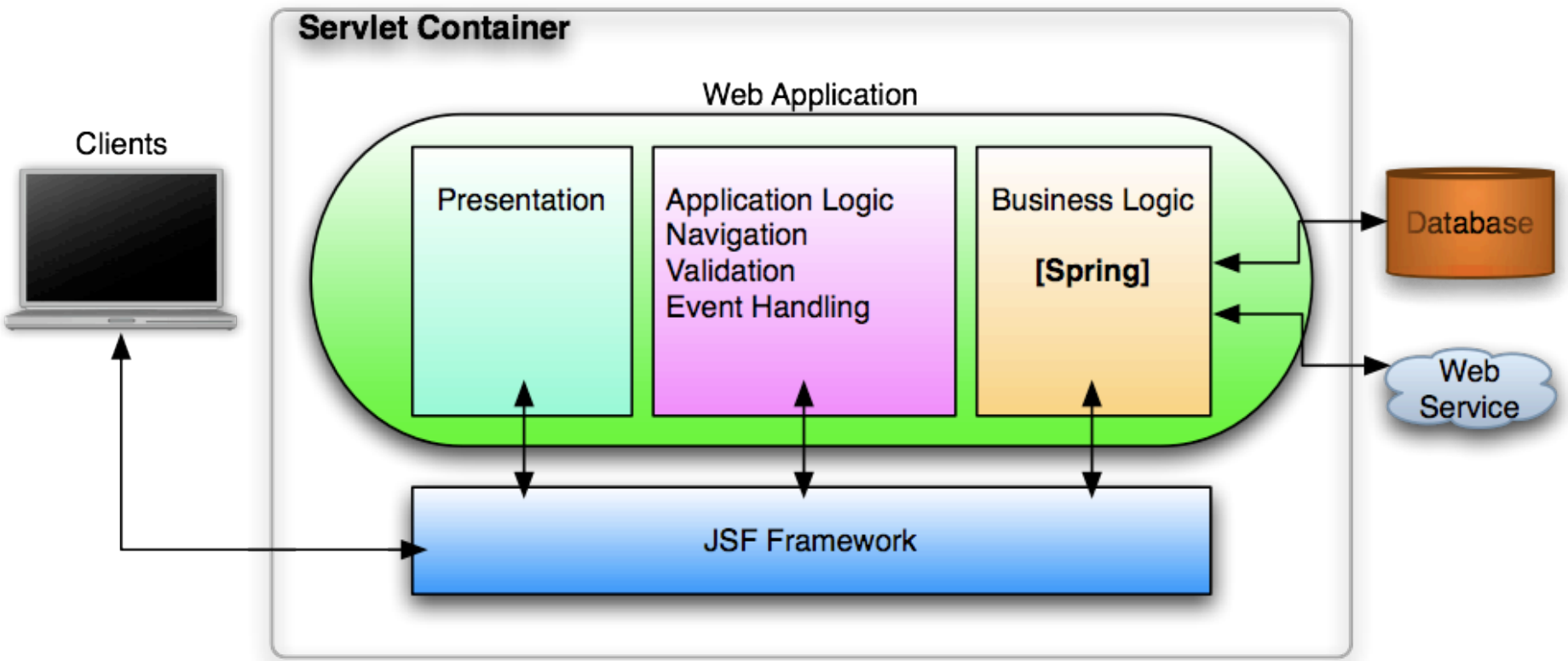



```
<form jwcid="@Form" delegate="o
<input type="hidden" jwcid="@Hi
<table class="detail">
<tr>
    <th>
        <label for="firstName">
    </th>
    <td><input jwcid="@TextFiel                                    rstName"/></td>
</tr>
<tr>
    <th>
        <label jwcid="@FieldLab                                    me</label>:
    </th>
    <td><input jwcid="lastNameF
</tr>
<tr>
    <th>
        <label for="birthday"><
    </th>
    <td>
        <input jwcid="@DatePicker" format="message:date.format" type="text"
            size="11" value="ognl:user.birthday" id="birthday"/>
    </td>
</tr>
```

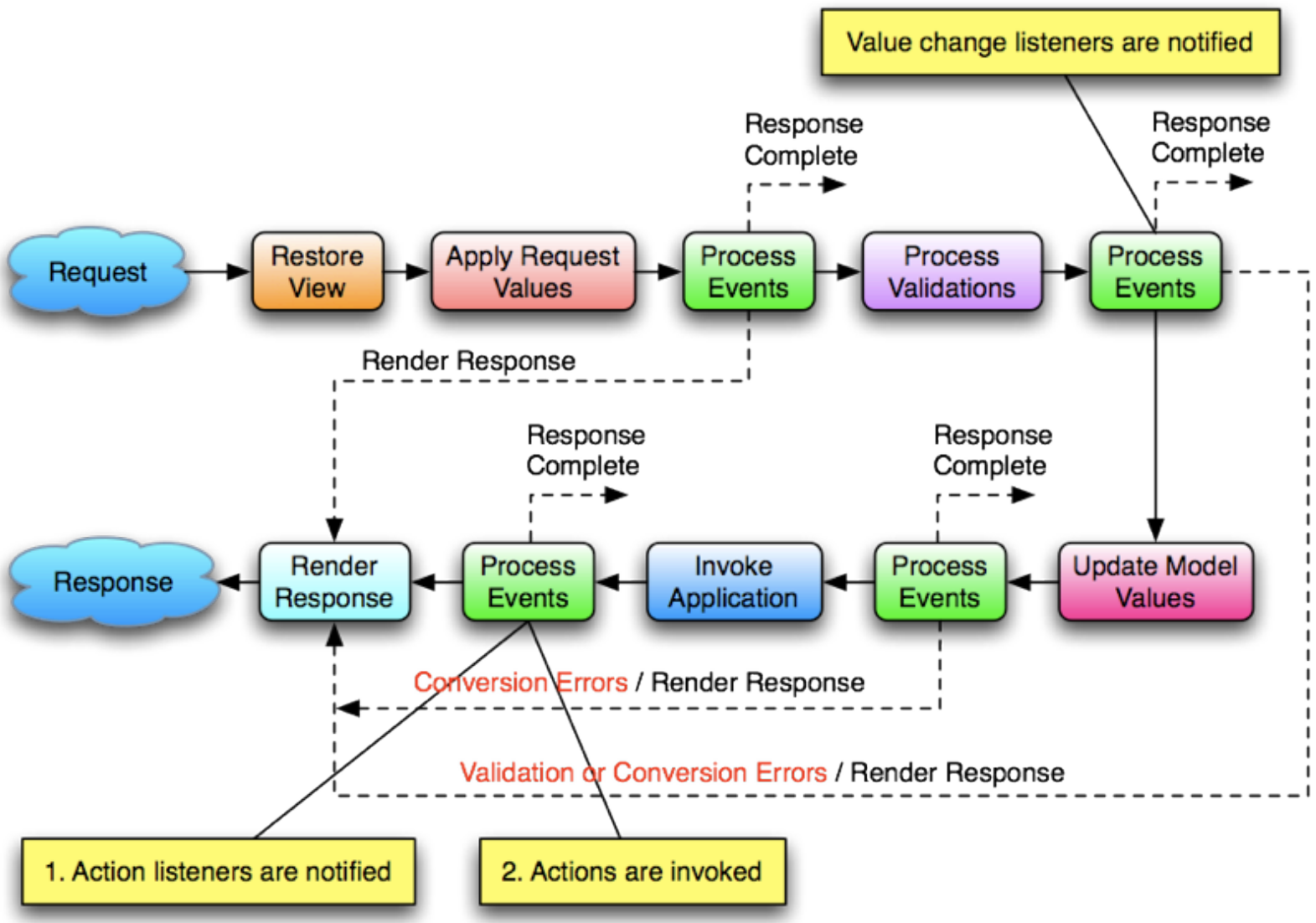
New in Tapestry 4.0

- Rich Annotation support
- Extremely configurable - now based on Hivemind
- Less code required - page-backing objects simpler
- Friendly URLs support
- Tacos Ajax Components

JSF



JSF



JSF Managed Bean

```
public class UserForm {
    private String id;
    public User user = new User();
    public UserManager mgr;

    public void setId(String id) {
        this.id = id;
    }

    public void setUser(User user) {
        this.user = user;
    }

    public void setUserManager(UserManager userManager) {
        this.mgr = userManager;
    }

    public String edit() {
        if (id != null) {
            // assuming edit
            setUser(mgr.getUser(id));
        }

        return "success";
    }
```

faces-config.xml

```
<application>
    <variable-resolver>
        org.springframework.web.jsf.DelegatingVariableResolver
    </variable-resolver>
    <locale-config>
        <default-locale>en</default-locale>
        <supported-locale>en</supported-locale>
        <supported-locale>es</supported-locale>
    </locale-config>
    <message-bundle>messages</message-bundle>
</application>

<navigation-rule>
    <from-view-id>/userForm.jsp</from-view-id>
    <navigation-case>
        <from-outcome>cancel</from-outcome>
        <to-view-id>/userList.jsp</to-view-id>
    </navigation-case>
    <navigation-case>
        <from-outcome>success</from-outcome>
        <to-view-id>/userList.jsp</to-view-id>
        <redirect/>
    </navigation-case>
</navigation-rule>
```

faces-config.xml

```
<application>
    <variable-resolver>
        org.springframework.web.jsf.DelegatingVariableResolver
    </variable-resolver>
    <locale-config>
        <default-locale>en</default-locale>
        <supported-locale>en</supported-locale>
        <supported-locale>es</supported-locale>
    </locale-config>
    <message-bundle>messages</message-bundle>
</application>

<navigation-rule>
    <from-view-id>/userForm.jsp</from-view-id>
    <navigation-case>
        <from-outcome>*</from-outcome>
        <to-view-id>/userList.jsp</to-view-id>
        <redirect/>
    </navigation-case>
</navigation-rule>
```

faces-config.xml

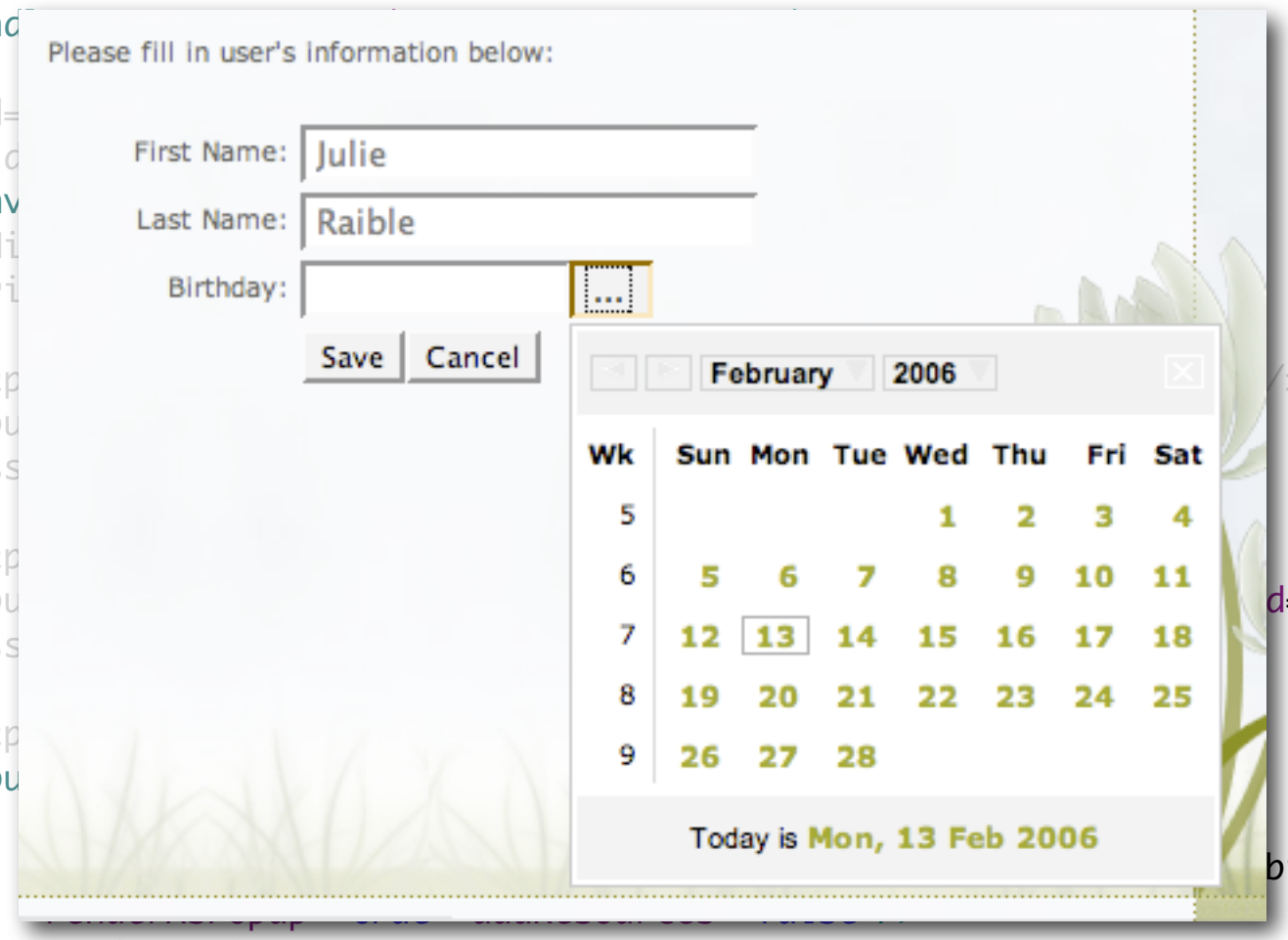
```
<managed-bean>
    <managed-bean-name>userForm</managed-bean-name>
    <managed-bean-class>org.appfuse.web.UserForm</managed-bean-class>
    <managed-bean-scope>request</managed-bean-scope>
    <managed-property>
        <property-name>id</property-name>
        <value>#{param.id}</value>
    </managed-property>
    <managed-property>
        <property-name>userManager</property-name>
        <value>#{userManager}</value>
    </managed-property>
</managed-bean>
```

JSF JSP View

```
<f:view>
<f:loadBundle var="messages" basename="messages"/>

<h:form id="userForm">
<h:inputHidden value="#{userForm.user.id}">
    <f:convertNumber/>
</h:inputHidden>
<h:panelGrid columns="3" styleClass="detail" columnClasses="label">

    <h:outputLabel for="firstName" value="#{messages['user.firstName']}"/>
    <h:inputText value="#{userForm.user.firstName}" id="firstName"/>
    <h:message for="firstName" styleClass="errorMessage"/>

    <h:outputLabel for="lastName" value="#{messages['user.lastName']}"/>
    <h:inputText value="#{userForm.user.lastName}" id="lastName" required="true"/>
    <h:message for="lastName" styleClass="errorMessage"/>

    <h:outputLabel for="birthday" value="#{messages['user.birthday']}"/>
    <t:inputCalendar monthYearRowClass="yearMonthHeader"
            weekRowClass="weekHeader" id="birthday"
            currentDayCellClass="currentDayCell" value="#{userForm.user.birthday}"
            renderAsPopup="true" addResources="false"/>
    <h:message for="birthday" styleClass="errorMessage"/>
```

JSF JSP View

```
<f:view>
<f:loadBund
<h:form id=
<h:inputHid
    <f:conv
</h:inputHi
<h:panelGri
    <h:outp
    <h:inpu
    <h:mess
    <h:outp
    <h:inpu                                          d="true"/>
    <h:mess
    <h:outp
    <t:inpu                                          birthday}"
    <h:message for="birthday" styleClass="errorMessage"/>
```



New in JSF 1.2

- Unified EL - better support for JSTL
- Focus on ease of use
- Java Studio Creator 2.0
- Many Ajax Components and Examples
- Open Source: ADF Faces, MyFaces/Tomahawk, Facelets

Sweetspots

Purpose of Experiment

- Discuss various open source Java web frameworks
- Highlight what each does well
- Debunk some myths
- Find out framework author's opinions of other frameworks
- Learn about the future direction of the framework
- Find out what the author's think of Ruby on Rails

Who Represented?

- JSF, Jacob Hookom
- RIFE, Geert Bevin
- Seam, Gavin King
- Spring MVC, Rob Harrop
- Spring Web Flow, Rob Harrop and Keith Donald
- Stripes, Tim Fennell
- Struts 1, Don Brown
- Tapestry, Howard Lewis Ship
- Trails, Chris Nelson
- WebWork, Patrick Lightbody
- Wicket, Eelco Hillenius

What's your "sweet spot"?

- **JSF:** For when you want to bring desktop-like functionality to the browser with the reliance of a standard specification and large amounts of third-party features.
- **Spring MVC:** Integrates a number of different technologies and as a result is applicable to a wide range of project types. It should be considered a strategic base platform for web application development.

What's your "sweet spot"?

- **Stripes:** Applications with lots of complex data interactions. Its type conversion, binding, and validation are very powerful and make it easy to manage large, complex forms and map them directly to domain objects, etc.
- **Tapestry:** Real strengths come through on medium- to large-sized projects (although you can have fun even on a single-page application). Those are the projects where you'll get leverage by being able to easily create new components.

What's your "sweet spot"?

- **WebWork:** Usually fits in best with small teams that are willing to get their hands dirty and learn a lot about the open source tools they use. WebWork is not meant for the “armchair programmers” who prefer drag-and-drop development.
- **Wicket:** Well suited for intranet/extranet applications, where the UI is relatively complex and where you want to make the best use of your developer resources.

What's your opinion?

The Smackdown

Evaluation Criteria

- **Ajax Support**: Is it built-in and easy to use?
- **Bookmark-ability**: Can users bookmark pages and return to them easily?
- **Validation**: How easy is it to use and does it support client-side (JavaScript) validation?
- **Testability**: How easy is it to test Controllers out of container?

Evaluation Criteria, cont.

- **Post and Redirect**: How does the framework handle the duplicate post problem?
- **Internationalization**: How is i18n supported and how easy is it to get messages in Controllers?
- **Page Decoration**: What sort of page decoration/ composition mechanisms does the framework support?
- **Community and Support**: Can you get questions answered quickly (and respectfully)?

Evaluation Criteria, cont.

- **Tools**: Is there good tool (particularly IDE) support for the framework?
- **Marketability of Skills**: If you learn the framework, will it help you get a job?
- **Job Count**: What is the demand for framework skills on dice.com and indeed.com?

Ajax Support

- Is Ajax support built-in and easy to use?
  - JSF: No Ajax support, use ICEfaces and Ajax4JSF
  - Stripes: No libraries, supports streaming results
  - Struts 2: Dojo built-in, plugins for GWT, JSON
  - Spring MVC: No, use DWR & Spring MVC Extras
  - Tapestry: Dojo built-in in 4.1
  - Wicket:

Bookmarking and URLs

- Using CMA allows users to bookmark pages. They can click the bookmark, login and go directly to the page.
  - JSF does a POST for everything - URLs not even considered
  - Stripes uses conventions, but you can override
  - Struts 2 has **namespaces** - makes it easy
  - Spring MVC allows **full URL control**
  - Tapestry still has somewhat ugly URLs
  - Wicket allows pages/URLs to be **mounted**

Validation

- Validation should be easy to configure, be robust on the client side and either provide good out of the box messages or allow you to easily customize them.
  - JSF has ugly default messages, but easiest to configure
  - Spring MVC allows you to use Commons Validator - a mature solution
  - Struts 2 uses OGNL for powerful expressions - client-side only works when specifying rules on Actions
  - Tapestry has very robust validation - good messages without need to customize
  - Stripes and Wicket do validation in Java - no client-side

Testability

- Spring and WebWork allow easy testing with mocks (e.g. EasyMock, jMock, Spring Mocks)
- Tapestry appears difficult to test because page classes are abstract, Creator class simplifies
- JSF page classes can be easily tested and actually look a lot like WebWork actions
- Wicket has WicketTester, a powerful solution
- Stripes has Servlet API Mock and MockRoundtrip

Post and Redirect

- The duplicate-post problem, what is it?
- Easiest way to solve: redirect after POST
- Is there support for allowing success messages to live through a redirect?
  - Spring MVC allows you to add parameters to a redirect
  - Stripes, Tapestry and Wicket all have "flash" support
  - Struts 2 requires a custom solution
  - JSF requires a custom solution, i18n messages difficult to get in page beans

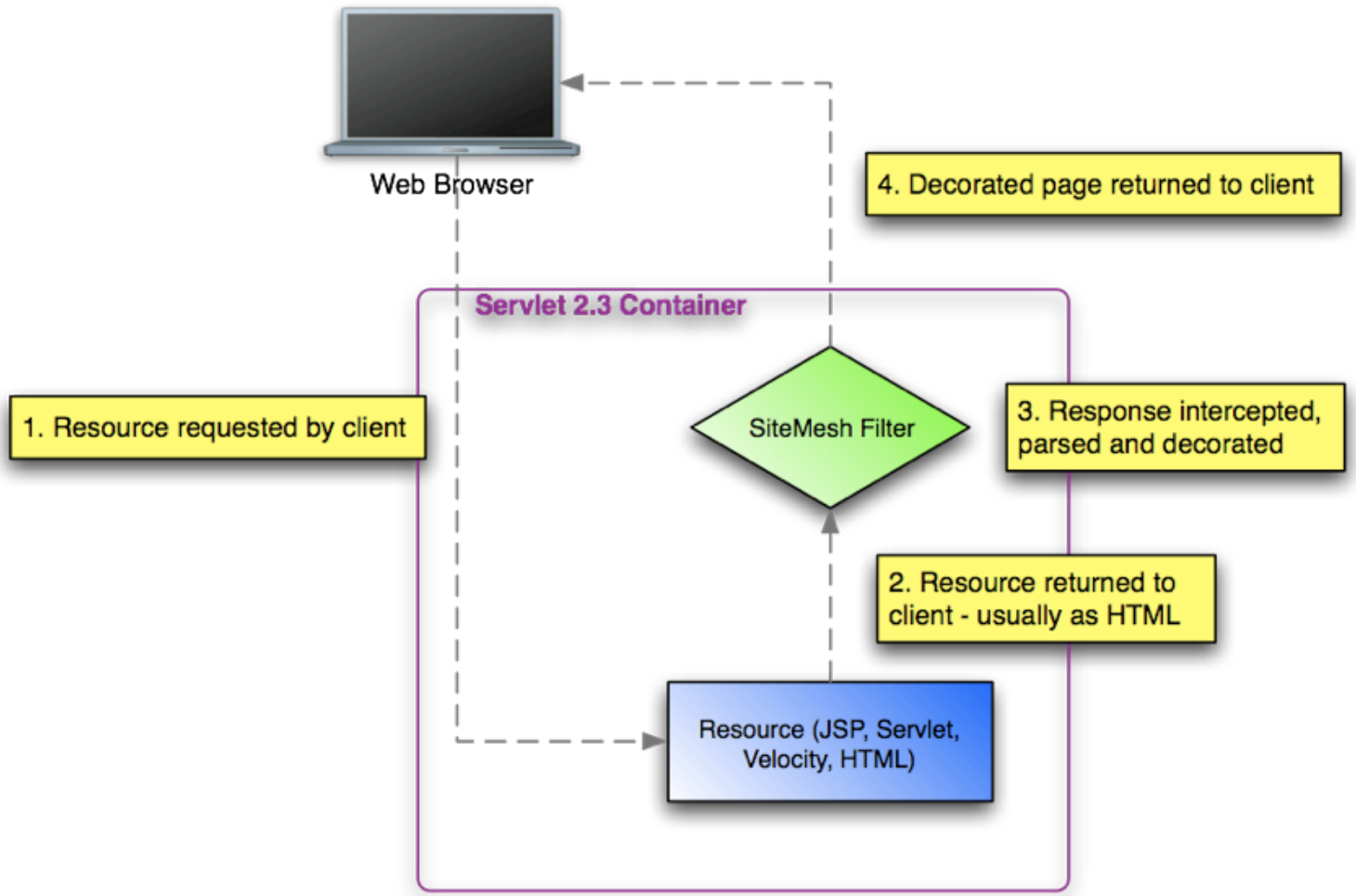
Internationalization

- JSTL's <fmt:message> tag makes it easy
- No standard for getting i18n messages in controller classes
- Stripes, Spring MVC and JSF use a single ResourceBundle per locale
- Struts 2, Tapestry and Wicket advocate separate files for each page/action
- JSF requires resource bundle to be declared on each page
- Tapestry's is awesome

Page Decoration

- Tiles Experience: used since it first came out
- SiteMesh is much easier to setup and use
- Tiles can be used in Struts 2, Spring and JSF
  - Requires configuration for each page
- SiteMesh can be used with all frameworks
  - Requires very little maintenance after setup
- SiteMesh not supported or recommended for use with JSF, Tapestry or Wicket

SiteMesh



SiteMesh: web.xml

```
<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>
    <dispatcher>REQUEST</dispatcher>
    <dispatcher>FORWARD</dispatcher>
</filter-mapping>
```

/WEB-INF/sitemesh.xml

```
<sitemesh>
    <property name="decorators-file" value="/WEB-INF/decorators.xml"/>
    <excludes file="${decorators-file}"/>
    <page-parsers>
        <parser default="true"
            class="com.opensymphony.module.sitemesh.parser.FastPageParser"/>
        <parser content-type="text/html"
            class="com.opensymphony.module.sitemesh.parser.FastPageParser"/>
        <parser content-type="text/html;charset=ISO-8859-1"
            class="com.opensymphony.module.sitemesh.parser.FastPageParser"/>
    </page-parsers>

    <decorator-mappers>
        <mapper class="com.opensymphony.module.sitemesh.mapper.ConfigDecoratorMapper">
            <param name="config" value="${decorators-file}"/>
        </mapper>
    </decorator-mappers>
</sitemesh>
```

/WEB-INF/decorators.xml

```
<decorators defaultdir="/decorators">
    <excludes>
        <pattern>/demos/*</pattern>
        <pattern>/resources/*</pattern>
    </excludes>
    <decorator name="default" page="default.jsp">
        <pattern>/*</pattern>
    </decorator>
</decorators>
```

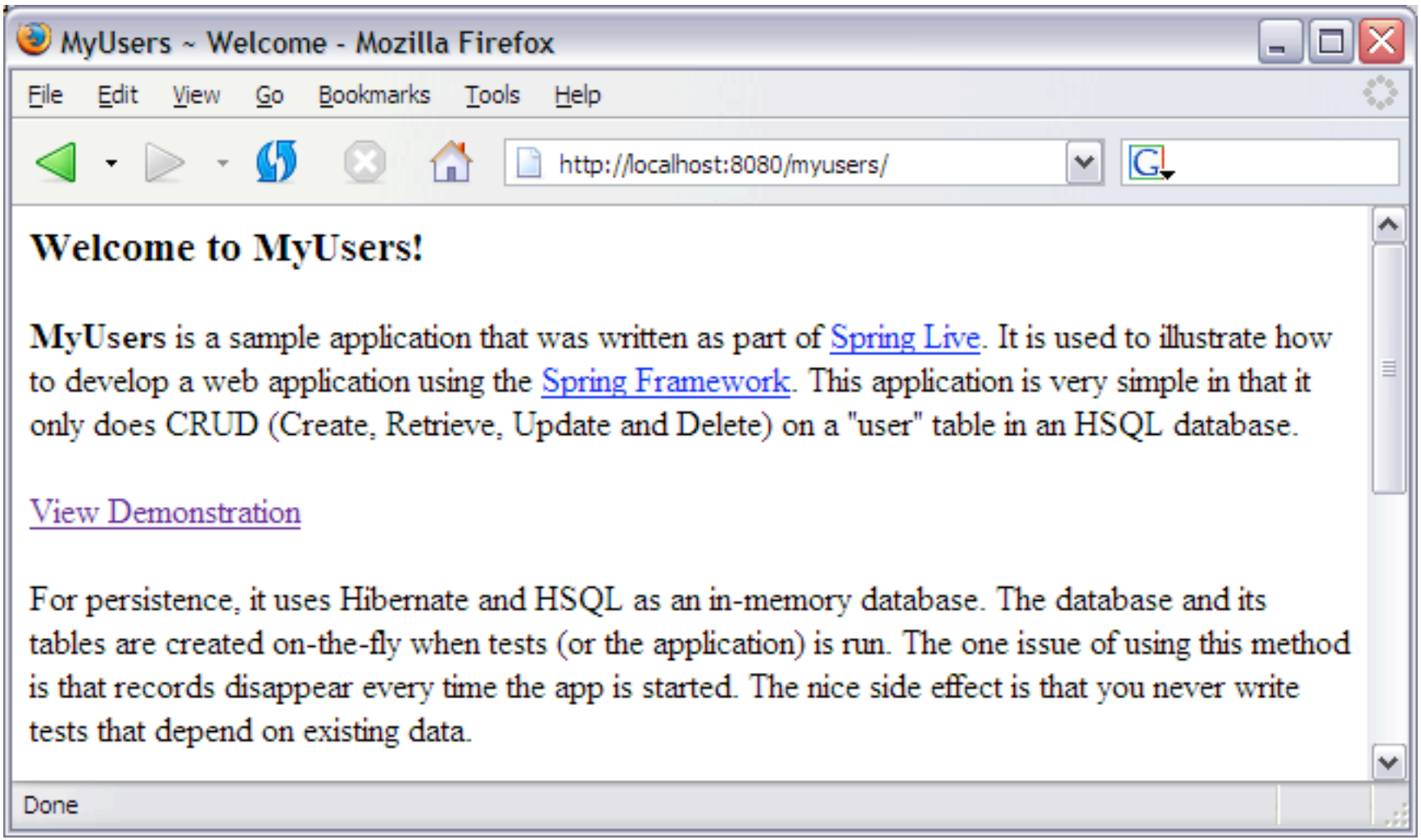
Sample Decorator

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<%@ include file="/taglibs.jsp"%>

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
    <title><decorator:title default="Equinox"/></title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
    <link href="${ctx}/styles/global.css" type="text/css" rel="stylesheet"/>
    <link href="${ctx}/images/favicon.ico" rel="SHORTCUT ICON"/>
    <script type="text/javascript" src="${ctx}/scripts/global.js"></script>
    <decorator:head/>
</head>
<body<decorator:getProperty property="body.id" writeEntireProperty="true"/>>
<a name="top"></a>

    <div id="content">
        <%@ include file="/messages.jsp"%>
        <decorator:body/>
    </div>

</body>
```

Sample Decorator

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<%@ include file="/taglibs.jsp"%>

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
    <title><decorator:title default="Equinox"/></title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
    <link href="${ctx}/styles/global.css" type="text/css" rel="stylesheet"/>
    <link href="${ctx}/images/favicon.ico" rel="SHORTCUT ICON"/>
    <script type="text/javascript" src="${ctx}/scripts/global.js"></script>
    <decorator:head/>
</head>
<body<decorator:getProperty property="body.id" writeEntireProperty="true"/>>
<a name="top"></a>

    <div id="content">
        <%@ include file="/messages.jsp"%>
        <decorator:body/>
    </div>

</body>
```

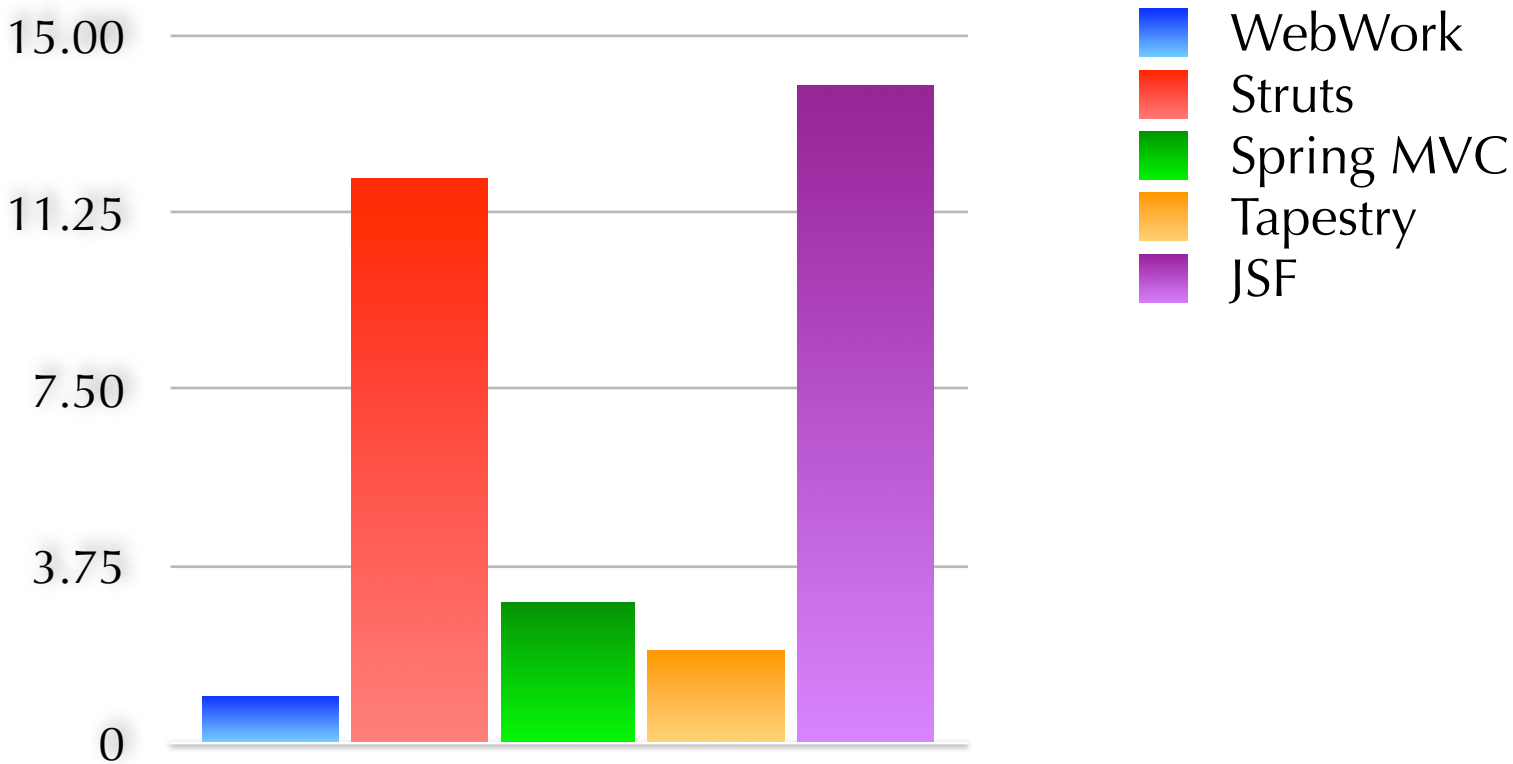
Before SiteMesh



Tools

- Spring has Spring IDE - only does XML validation, not a UI/web tool
- WebWork has EclipseWork
- Tapestry has Spindle - great for coders
- JSF has many, and they're getting better and better
- Stripes and Wicket don't have any official tools
- NetBeans has support for: Struts *, JSF (+Facelets), Tapestry and Wicket (no Stripes or Spring MVC)

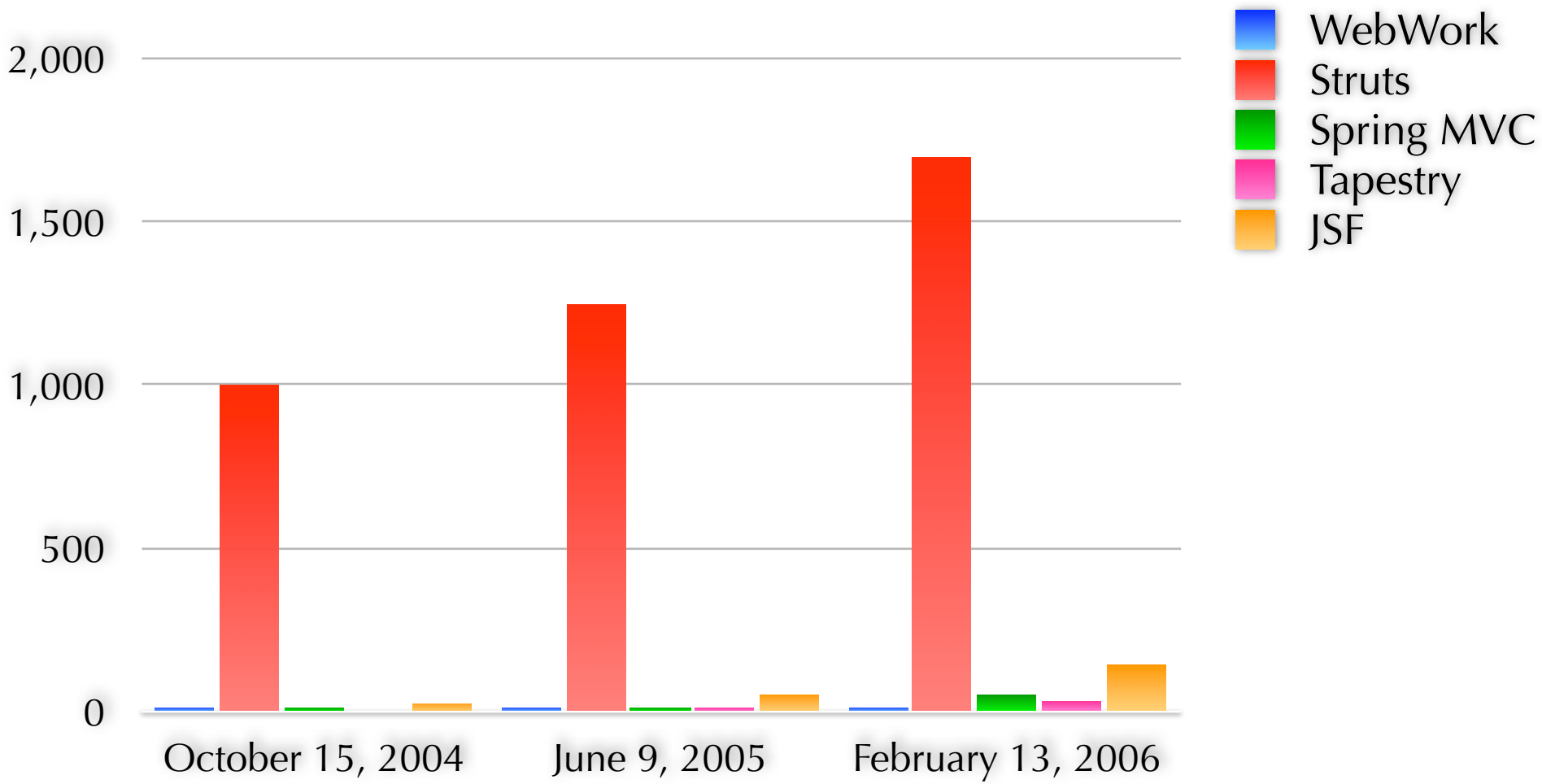
Tools Available



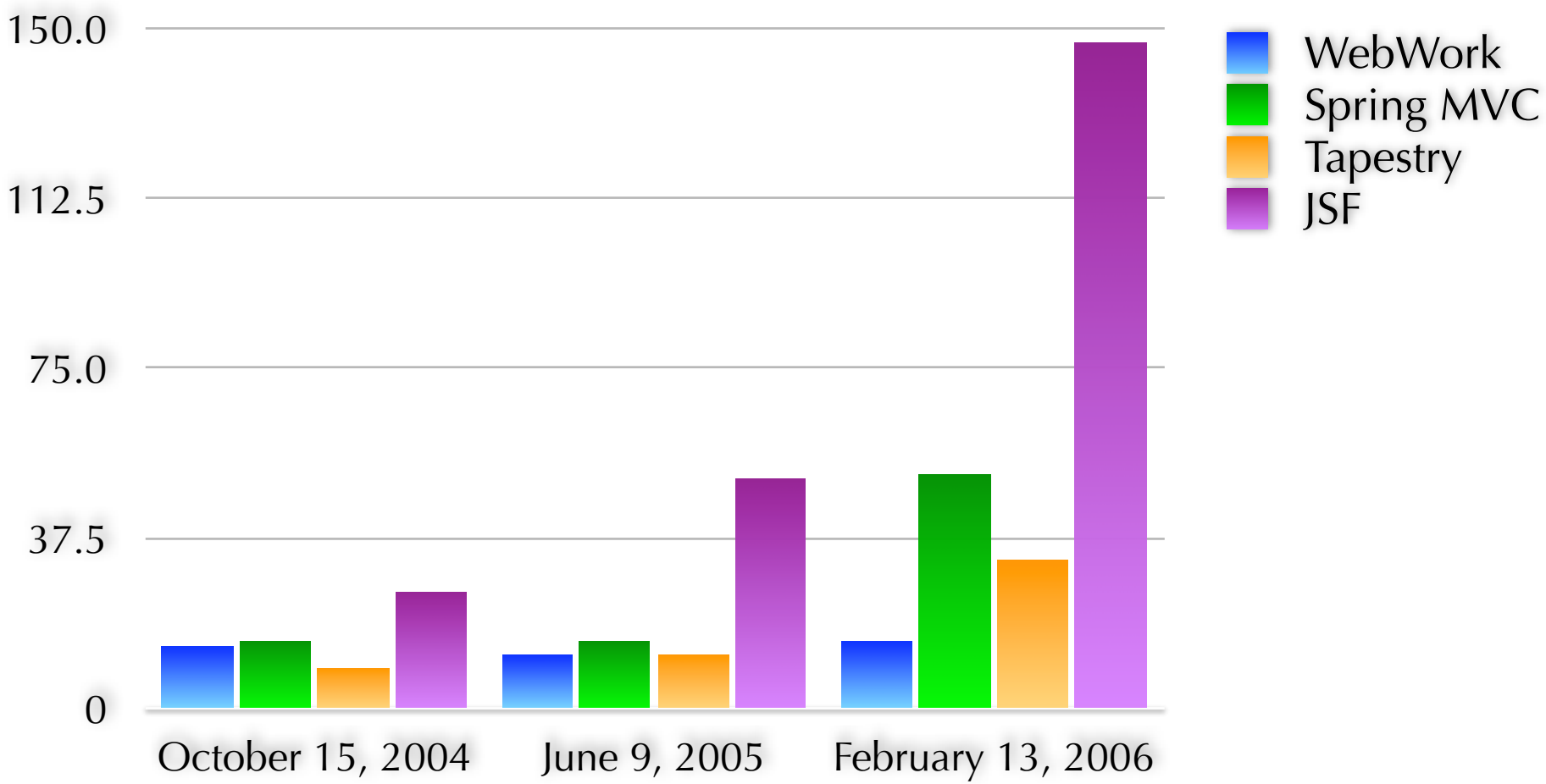
Marketability of Skills

- Struts is still in high-demand and widely-used
- Spring is getting more press, but mostly due to the framework's other features
- WebWork is gaining ground, but very scarce on job boards
- Tapestry is even more scarce - needs more marketing
- JSF is quickly becoming popular

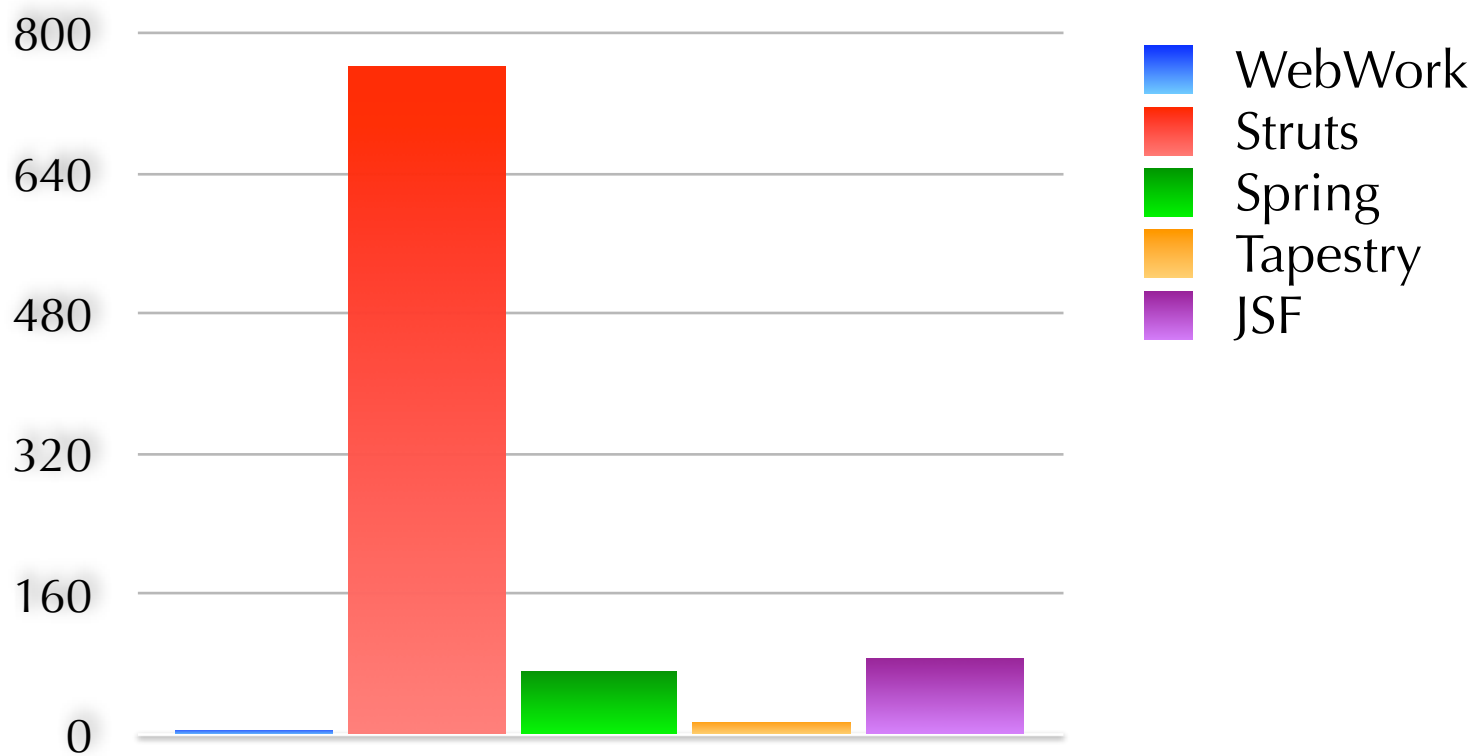
Dice Job Count



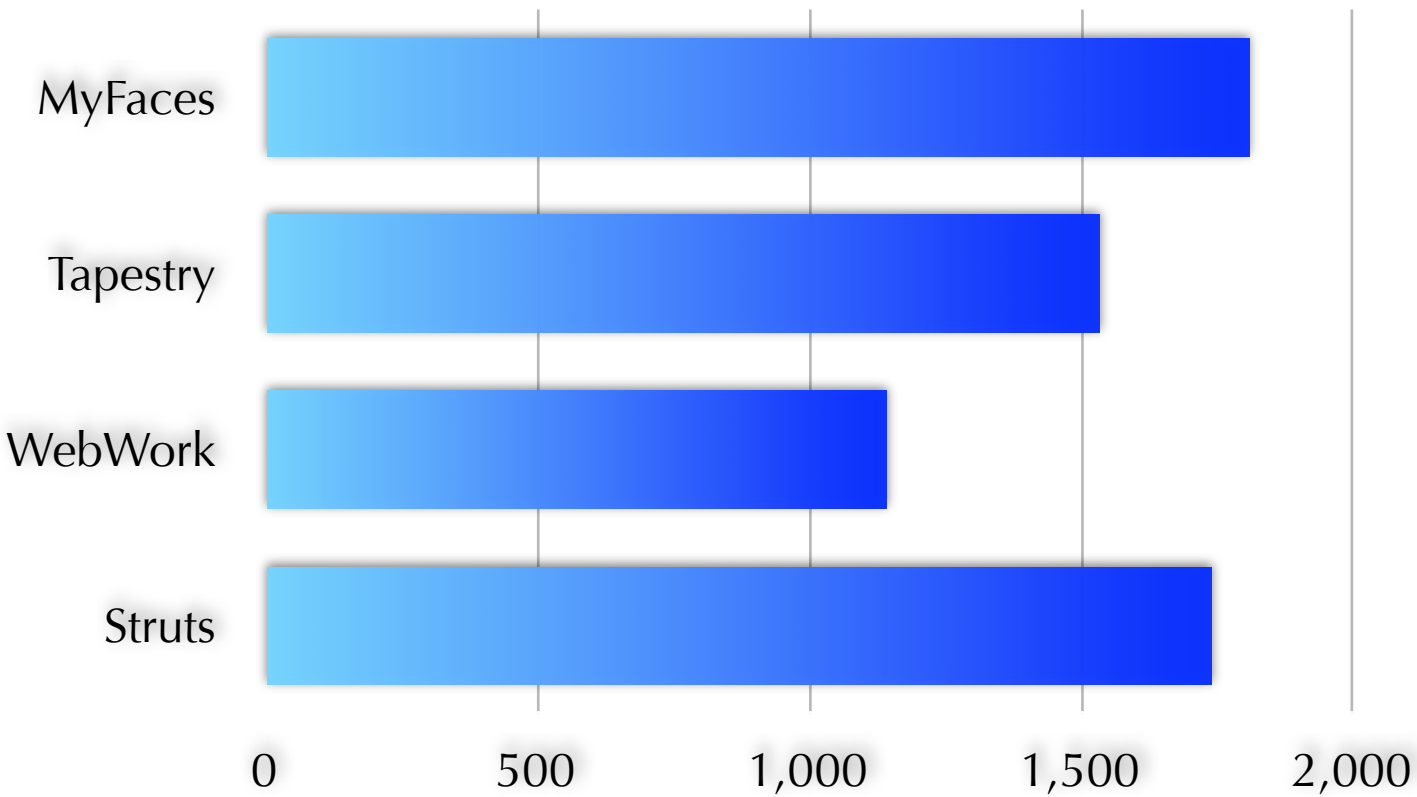
Jobs sans Struts



Employer Search on Monster.com
Resumes posted 2/11 - 2/13/2006

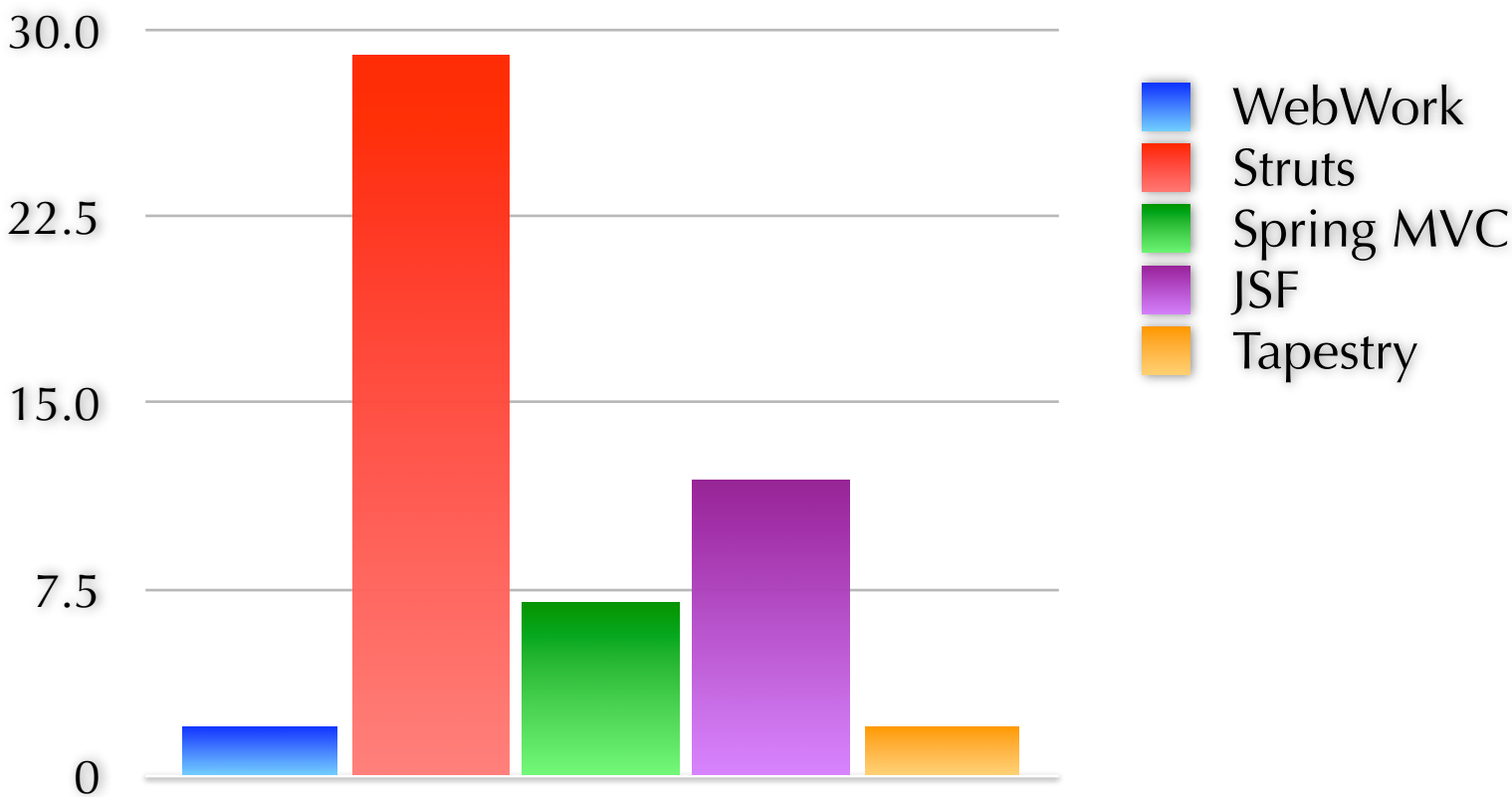


Mailing List Traffic



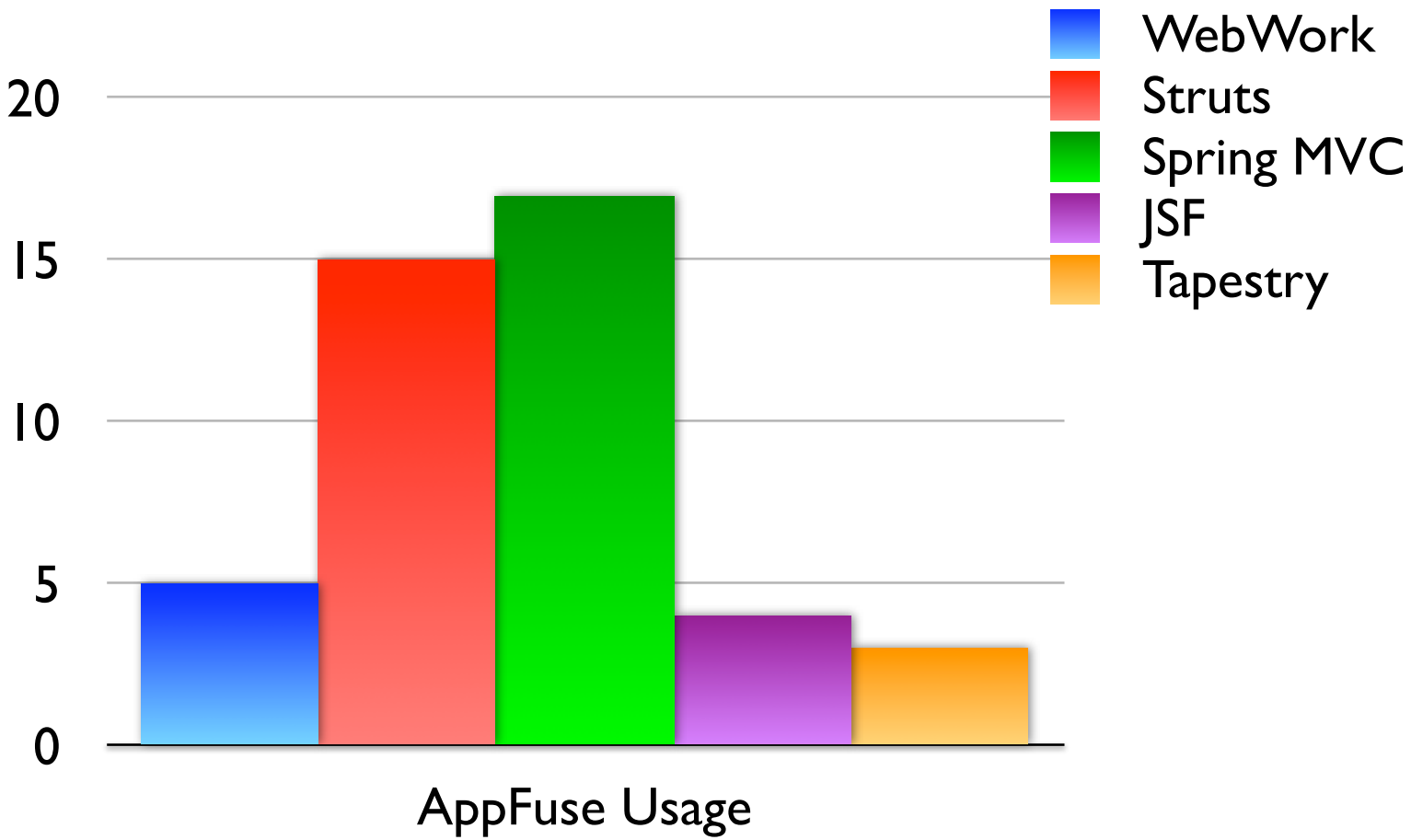
** Spring MVC is not listed here because they have a forum instead of a mailing list and I couldn't figure out a way to count the number of messages for each month.*

Books on Amazon



Which would I choose?

What do others think?



http://raibledesigns.com/page/rd?entry=spring_mvc_the_most_popular

Resources

- Download sample apps from this presentation
  - http://equinox.dev.java.net/framework-comparison
- Struts - http://struts.apache.org
  - **StrutsTestCase**: http://strutstestcase.sf.net
- Spring MVC - http://www.springframework.org
  - **Spring IDE**: http://www.springide.org
  - **Gaijin Studio**: http://gaijin-studio.sf.net
- WebWork - http://opensymphony.org/webwork
  - **Eclipse Plugin**: http://sf.net/projects/eclipsework
  - **IDEA Plugin**: http://wiki.opensymphony.com/display/WW/IDEA+Plugin

Resources, cont.

- Tapestry - http://jakarta.apache.org/tapestry
  - **Spindle**: http://spindle.sourceforge.net
- JSF - http://java.sun.com/j2ee/javaserverfaces and http://myfaces.apache.org
  - **Java Studio Creator**: http://sun.com/software/products/jscreator
  - **MyEclipse**: http://myeclipseide.com
- **IDEA**: http://www.jetbrains.com/idea
- **SiteMesh**: http://opensymphony.com/sitemesh

Resources, cont.

- Testing Frameworks
  - **JUnit**: http://junit.org
  - **EasyMock**: http://easymock.org
  - **jMock**: http://jmock.org
  - **jWebUnit**: http://jwebunit.sourceforge.net
  - **Canoo WebTest**: http://webtest.canoo.com
  - **Tapestry Test Assist**: http://howardlewisship.com/blog/2004/05/tapestry-test-assist.html
- **AppFuse** - http://appfuse.org

Books

- **Struts in Action**, Ted Husted and Team
- **Struts Live**, Rick Hightower and Jonathan Lehr
- **Spring Live**, Matt Raible
- **Pro Spring**, Rob Harrop and Jan Machacek
- **Spring in Action**, Craig Walls and Ryan Breidenbach
- **Professional Java Development with Spring**, Rod Johnson, Juergen Hoeller and Team

Books, cont.

- **WebWork Live**, Matthew Porter
- **WebWork in Action**, Patrick Lightbody and Team
- **Tapestry Live**, Warner Onstine
- **Tapestry in Action**, Howard Lewis Ship
- **Core JSF**, David Geary and Cay Horstmann
- **JSF in Action**, Kito Mann

Grails

GWT

Seam

What's Next?

Django

Ruby on Rails

Trails

OpenLaszlo

Who cares?

"If it works, use it!"

Questions?

matt@raibledesigns.com

Raible Designs