


Spring 2.0 Kickstart

Thomas Risberg
Matt Raible

 Spring Forward 2006

Spring 2.0 Kickstart

Web MVC

 Spring Forward 2006

Introductions

- How many people are current using Spring MVC?
- What about Struts, Tapestry or JSF?
- What do you hope to learn today?



Introduction



Matt Raible

- ▶ Spring and Web Frameworks Practice Leader for Virtuas Open Source Solutions
- ▶ JCP Expert Group for Java EE, JSF and Bean Validation
- ▶ Founder of AppFuse and Equinox projects
- ▶ Author "Spring Live" from SourceBeat
- ▶ Dad, Husband and Montana Native



What's new in Spring 2.0

- JSP Tag Library for Forms
- Convention over Configuration:
 - ControllerClassNameHandlerMapping
 - ModelMap: don't need to specify model names
 - RequestToViewNameTranslator

<http://www.springframework.org/docs/reference/mvc.html#mvc-coc>

Spring MVC Basics

- web.xml Configuration
- Controller
- Spring XML Configuration
- SimpleFormController
- View options and Form Tag Library
- Validation

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd" version="2.4">

  <servlet>
    <servlet-name>kickstart</servlet-name>
    <servlet-class>
      org.springframework.web.servlet.DispatcherServlet
    </servlet-class>
    <load-on-startup>0</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>kickstart</servlet-name>
    <url-pattern>*.htm</url-pattern>
  </servlet-mapping>
</web-app>
```

Loading middle-tier beans

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd" version="2.4">

  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:/repository-config.xml</param-value>
  </context-param>

  <listener>
    <listener-class>
      org.springframework.web.context.ContextLoaderListener
    </listener-class>
  </listener>

  ...
</web-app>
```

Controller Interface

- Has *handleRequest()* method that returns a `ModelAndView`
- Base interface for all controllers:
handleRequest() can be called in unit tests
- **ModelAndView**: a class that holds both `Model` and a `View`
- **AbstractCommandController**: use for populating a command object with request parameters

ModelAndView

- Many constructor options make it easy to use
- View names are logical names that are configured by ViewResolvers
- Model can be Map or a JavaBean object

```
public ModelAndView(String viewName) {  
    this.viewName = viewName;  
}
```

```
public ModelAndView(String viewName, Map model) {  
    this.viewName = viewName;  
    this.model = model;  
}
```

```
public ModelAndView(String viewName, String modelName,  
                    Object modelObject) {  
    this.viewName = viewName;  
    addObject(modelName, modelObject);  
}
```

Controller

```
public class CustomerController implements Controller {
    private CustomerService customerService;

    public void setCustomerService(CustomerService userService) {
        this.customerService = userService;
    }

    public ModelAndView handleRequest(HttpServletRequest request,
                                     HttpServletResponse response)
        throws Exception {
        return new ModelAndView().addObject(customerService.getListOfCustomers());
    }
}
```

```
<display:table name="customers" class="list" requestURI="" id="customer" export="true">
```

Configuration

- Controllers are configured as bean definitions in *kickstart-servlet.xml* where *kickstart* is the name of the DispatcherServlet in *web.xml*

```
<bean name="/customers.htm" class="spring.kickstart.web.CustomerController">  
  <property name="customerService" ref="customerService"/>  
</bean>
```

OR

```
<bean id="customerController" class="spring.kickstart.web.CustomerController">  
  <property name="customerService" ref="customerService"/>  
</bean>
```

URL Mapping

- **BeanNameUrlHandlerMapping** is the default - where URLs are matched to bean names
- **SimpleUrlHandlerMapping** provides central means of configuring URLs and allows interceptors

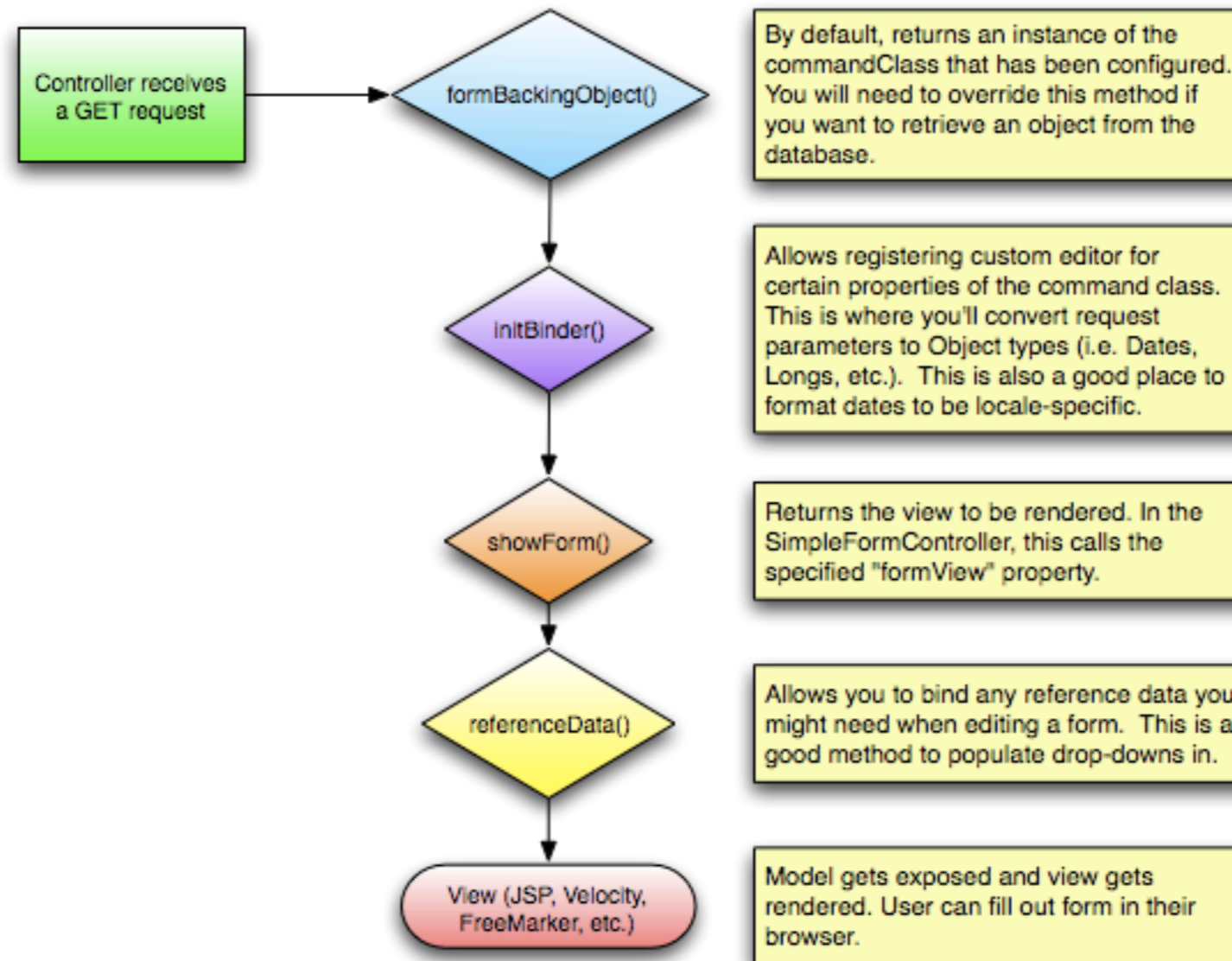
```
<bean id="urlMapping"  
      class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">  
  <property name="mappings">  
    <props>  
      <prop key="/customers.htm">customerController</prop>  
      <prop key="/customerform.htm">customerFormController</prop>  
    </props>  
  </property>  
</bean>
```

- **ControllerClassNameHandlerMapping** allows you to use convention-over-configuration

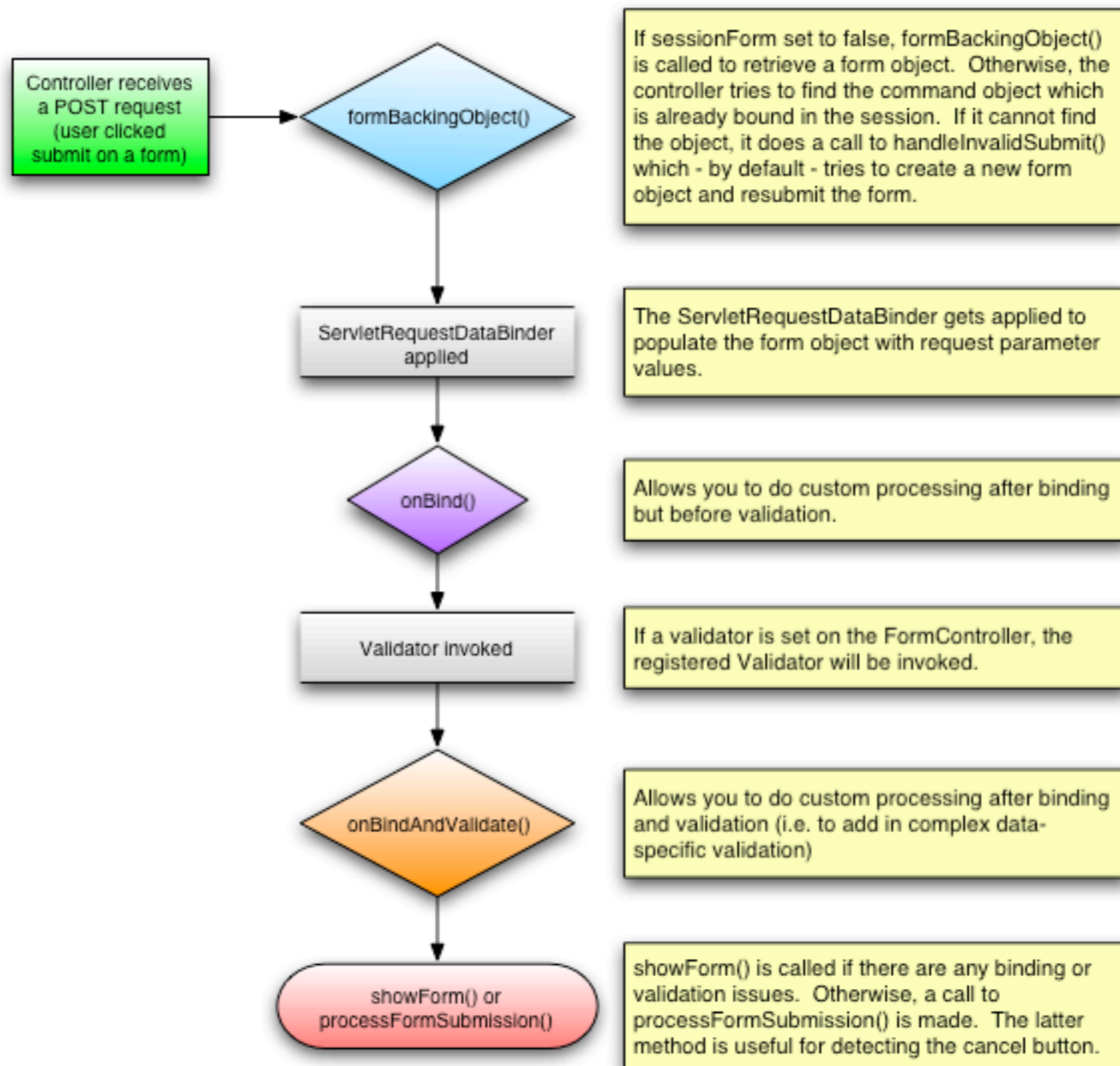
Form Controllers

- **SimpleFormController**: best to use for processing forms
- **AbstractWizardFormController**: use for processing wizards
- **AbstractFormController**: parent of both Simple/AbstractWizardFormControllers. Requires before/after view names to be configured programmatically
- **ThrowawayController**: command pattern with single execute() method

HTTP GET Lifecycle



POST Lifecycle



formBackingObject()

```
protected Object formBackingObject(HttpServletRequest request)
    throws ServletException {
    String id = request.getParameter("id");

    if ((id != null) && !id.equals("")) {
        Customer customer = customerService.locateCustomerById(new Long(id));

        if (customer == null) {
            return new Customer();
        }

        return customer;
    } else {
        return new Customer();
    }
}
```

initBinder()

```
protected void initBinder(HttpServletRequest request,
                          ServletRequestDataBinder binder) {
    // convert java.util.Date
    SimpleDateFormat dateFormat = new SimpleDateFormat("MM/dd/yyyy");
    dateFormat.setLenient(false);
    binder.registerCustomEditor(Date.class, null,
                                new CustomDateEditor(dateFormat, true));

    // convert java.lang.Long
    binder.registerCustomEditor(Long.class, null,
                                new CustomNumberEditor(Long.class, null, true));
}
```

doSubmitAction()

```
protected void doSubmitAction(Object object) throws Exception {  
    Customer customer = (Customer) object;  
    if (customer.getId() == null) {  
        customerService.addNewCustomer(customer);  
    } else {  
        customerService.updateCustomer(customer);  
    }  
}
```

SimpleFormControllerTest

```
public class CustomerFormControllerTest extends AbstractDependencyInjectionSpringContextTests {
    private CustomerFormController form;

    public void setCustomerFormController(CustomerFormController form) {
        this.form = form;
    }

    protected String[] getConfigLocations() {
        return new String[]{"file:*/kickstart-servlet.xml", "repository-config.xml"};
    }

    public void testEdit() throws Exception {
        form = new CustomerFormController(new CustomerServiceMock());
        // verify controller can grab user
        MockHttpServletRequest request = new MockHttpServletRequest("GET", "");
        request.addParameter("id", "1");
        ModelAndView mv = form.handleRequest(request, new MockHttpServletRequestResponse());
        assertEquals("customer", form.getCommandName());
        Customer customer = (Customer) mv.getModel().get(form.getCommandName());
        assertEquals(new Long(1), customer.getId());
    }
}
```

Bean Definition

```
<bean id="customerFormController" class="spring.kickstart.web.CustomerFormController">  
  <constructor-arg ref="customerService"/>  
  <property name="commandName" value="customer"/>  
  <property name="commandClass" value="spring.kickstart.domain.Customer"/>  
  <property name="successView" value="redirect:customers.htm"/>  
</bean>
```

Bean Definition

```
<bean id="customerFormController" class="spring.kickstart.web.CustomerFormController">
  <constructor-arg ref="customerService"/>
  <property name="commandName" value="customer"/>
  <property name="commandClass" value="spring.kickstart.domain.Customer"/>
  <property name="successView" value="redirect:customers.htm"/>
</bean>
```

```
public class CustomerFormController extends SimpleFormController {
    private CustomerService customerService;

    public CustomerFormController(CustomerService customerService) {
        setCommandClass(Customer.class);
        setCommandName("customer");
        this.customerService = customerService;
    }
    ...
}
```

View Options

- **JavaServer Pages:** includes support for JSTL (i18n, etc.)
- **Tiles:** allows you to use Tiles like you would with Struts - excellent for page composition
- **Velocity:** includes convenience macros to simplify form development
- **FreeMarker:** macros for form development
- **XSLT, PDF, Excel:** create classes to render view
- **JasperReports:** nice open-source reporting engine

View Resolvers

- Bean definition that defines how Spring MVC should resolve views
- Provide de-coupling between Controllers and view technology

JSP/JSTL ViewResolver

```
<!-- View Resolver for JSPs -->  
<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>
```

Velocity ViewResolver

```
<!-- Velocity Configurer and View Resolver -->
<bean id="velocityConfig"
      class="org.springframework.web.servlet.view.velocity.VelocityConfigurer">
  <property name="resourceLoaderPath" value="/" />
</bean>

<bean id="viewResolver"
      class="org.springframework.web.servlet.view.velocity.VelocityViewResolver">
  <property name="dateToolAttribute" value="date" />
  <property name="exposeSpringMacroHelpers" value="true" />
  <property name="requestContextAttribute" value="rc" />
  <property name="cache" value="true" />
  <property name="prefix" value="/" />
  <property name="suffix" value=".vm" />
</bean>
```

FreeMarker ViewResolver

```
<!-- FreeMarker Configurer and View Resolver -->
<bean id="freemarkerConfig"
      class="org.springframework.web.servlet.view.freemarker.FreeMarkerConfigurer">
  <property name="templateLoaderPath" value="/" />
  <property name="freemarkerSettings">
    <props>
      <prop key="datetime_format">MM/dd/yyyy</prop>
    </props>
  </property>
</bean>

<bean id="viewResolver"
      class="org.springframework.web.servlet.view.freemarker.FreeMarkerViewResolver">
  <property name="exposeSpringMacroHelpers" value="true" />
  <property name="requestContextAttribute" value="rc" />
  <property name="prefix" value="/" />
  <property name="suffix" value=".ftl" />
</bean>
```

JSP vs. Velocity vs. FreeMarker

JSP 2.0 + JSTL

```
<spring:bind path="user.*">
  <c:if test="{not empty status.errorMessages}">
    <div class="error">
      <c:forEach var="error" items="{status.errorMessages}">
        <c:out value="{error}" escapeXml="false"/><br />
      </c:forEach>
    </div>
  </c:if>
</spring:bind>
```

...

```
<form method="post" action="{c:url value='/editUser.html' />"
  onsubmit="return validateUser(this)" name="userForm">
  <spring:bind path="user.id">
  <input type="hidden" name="id" value="{status.value}"/>
  </spring:bind>
  <table class="detail">
  <tr>
    <th><label for="firstName"><fmt:message key="user.firstName"/>:</label></th>
    <td>
      <spring:bind path="user.firstName">
      <input type="text" name="firstName" id="firstName" value="{status.value}"/>
      <span class="fieldError">{status.errorMessage}</span>
      </spring:bind>
    </td>
  </tr>
```

Even easier in 2.0

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

Spring "form" tags

- checkbox
- errors
- form
- hidden
- input
- label
- option
- options
- password
- radiobutton
- select
- textarea

Velocity

```
#set($springXhtmlCompliant = true)

...

#springBind("user.*")
#if ($status.error)
<div class="error">
  #foreach ($error in $status.errorMessages)
    ${error}<br/>
  #end
</div>
#end

...

<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>
</table>
</form>
```

FreeMarker

```
<#import "/spring.ftl" as spring/>
<#assign xhtmlCompliant = true in spring>
...

<@spring.bind "user.*"/>
<#if spring.status.error>
<div class="error">
  <#list spring.status.errorMessagees as error>
    ${error}<br/>
  </#list>
</div>
</#if>
...

<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"'/>
    <span class="fieldError">
      ${spring.status.errorMessage}</span>
    </td>
</tr>
</table>
```

Validation

- Many different options for doing Validation
 - Spring's Validator interface
 - Commons Logging from Spring Modules
 - Valang from Spring Modules
 - Bean Validation Framework

Validator Interface

```
public class CustomerValidator implements Validator {
    private MessageSource messageSource;

    @Required
    public void setMessageSource(MessageSource messageSource) {
        this.messageSource = messageSource;
    }

    public boolean supports(Class clazz) {
        return clazz.equals(Customer.class);
    }

    public void validate(Object obj, Errors errors) {
        String arg1 = messageSource.getMessage("customer.name", null,
            LocaleContextHolder.getLocale());

        ValidationUtils.rejectIfEmptyOrWhitespace(errors,
            "name", "errors.required", new Object[] {arg1}, "Value required.");
    }
}
```

Commons Validator

- Spring support created by Daniel Miller in April 2004
- Moved from Spring CVS sandbox to Spring Modules project in April 2005
- Validation rules specified in */WEB-INF/validation.xml*
- Validators (client and server-side) configured in */WEB-INF/validator-rules.xml*

Bean Definitions

```
<bean id="validatorFactory"
      class="org.springframework.validation.commons.DefaultValidatorFactory">
  <property name="validationConfigLocations">
    <list>
      <value>/WEB-INF/validation.xml</value>
      <value>/WEB-INF/validator-rules.xml</value>
    </list>
  </property>
</bean>

<bean id="beanValidator"
      class="org.springframework.validation.commons.DefaultBeanValidator">
  <property name="validatorFactory" ref="validatorFactory"/>
</bean>
```

validation.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE form-validation PUBLIC
    "-//Apache Software Foundation//DTD Commons Validator Rules Configuration 1.3.0//EN"
    "http://jakarta.apache.org/commons/dtds/validator_1_3_0.dtd">

<form-validation>
  <formset>
    <form name="customer">
      <field property="name" depends="required">
        <arg0 key="customer.name"/>
      </field>
    </form>
  </formset>
</form-validation>
```

Client-side Validation

- Form's *onsubmit* handler:

```
<form:form method="post" action="customerform.htm"  
  onsubmit="return validateCustomer(this)" name="customerForm">
```

- JavaScript tags after form:

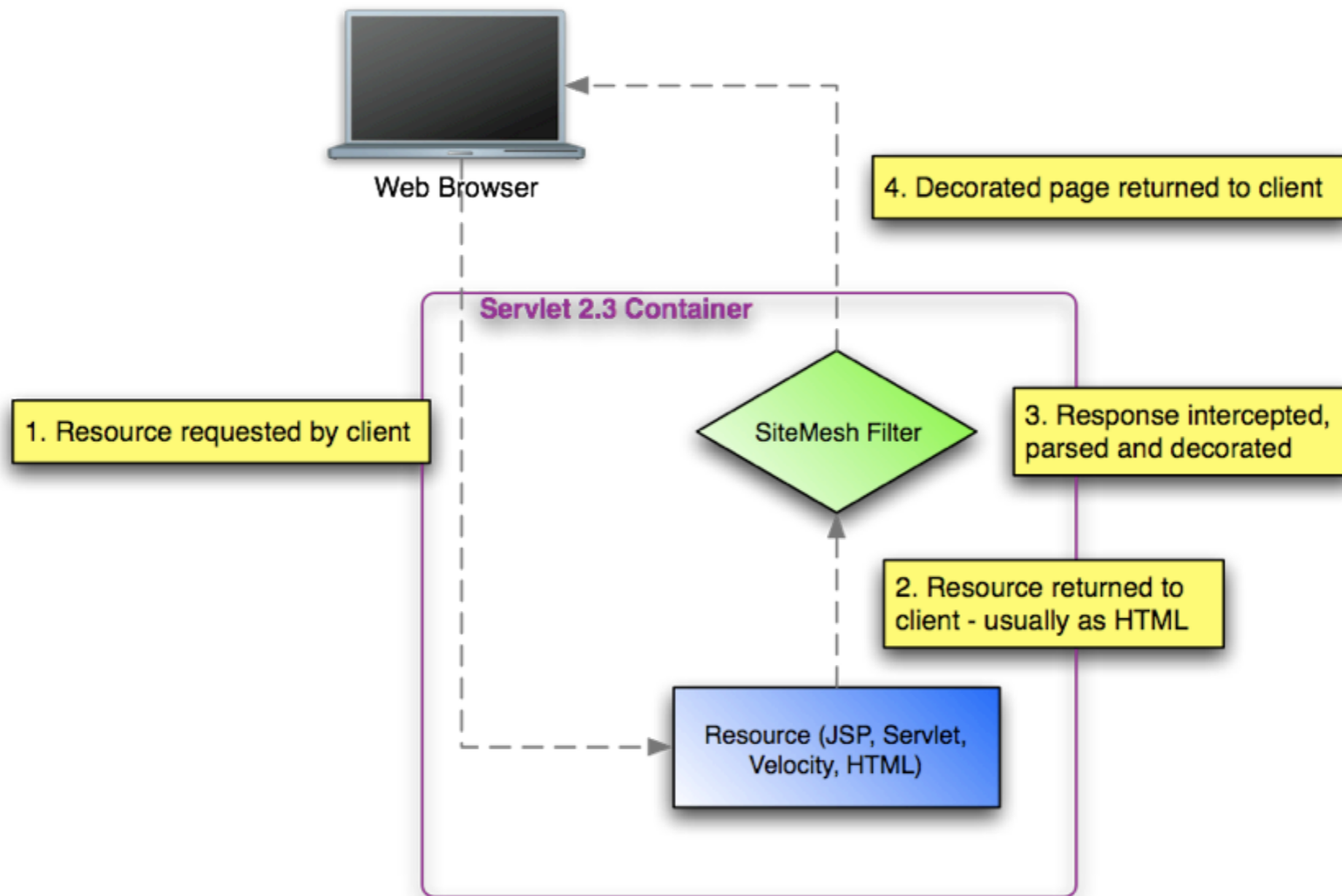
```
<v:javascript formName="customer" staticJavascript="false"  
  xhtml="true" cdata="false"/>  
<script type="text/javascript"  
  src="<c:url value="/scripts/validator.jsp"/>"></script>
```

- /scripts/validator.jsp

```
<%@ page language="java" contentType="text/javascript" %>  
<%@ taglib uri="http://www.springmodules.org/tags/commons-validator"  
  prefix="v" %>
```

```
<v:javascript dynamicJavascript="false" staticJavascript="true"/>
```

Decorating with 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>
</html>
```

Sample Decorator

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

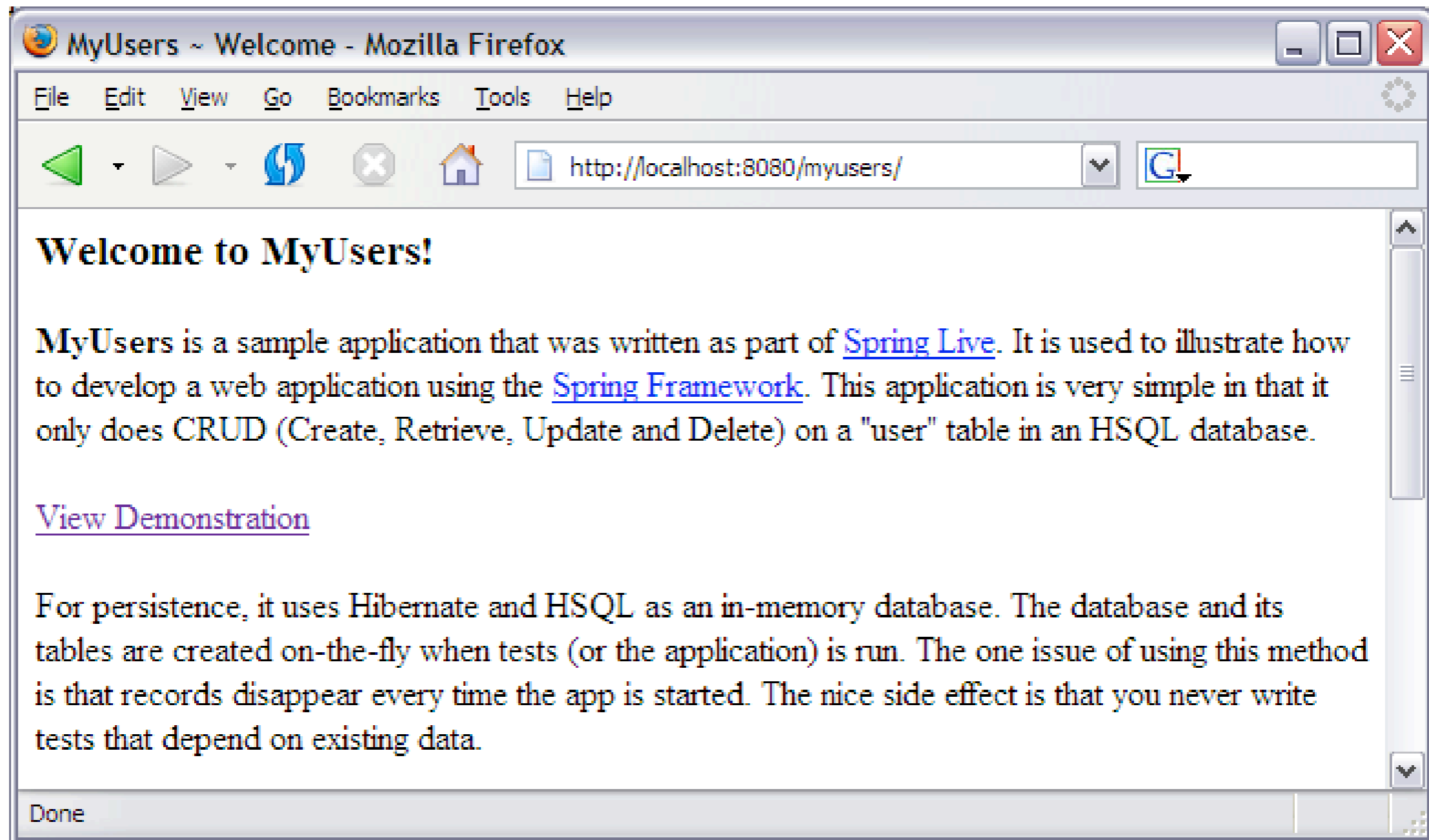
<%@ include file="/taglibs.jsp"%>

<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>

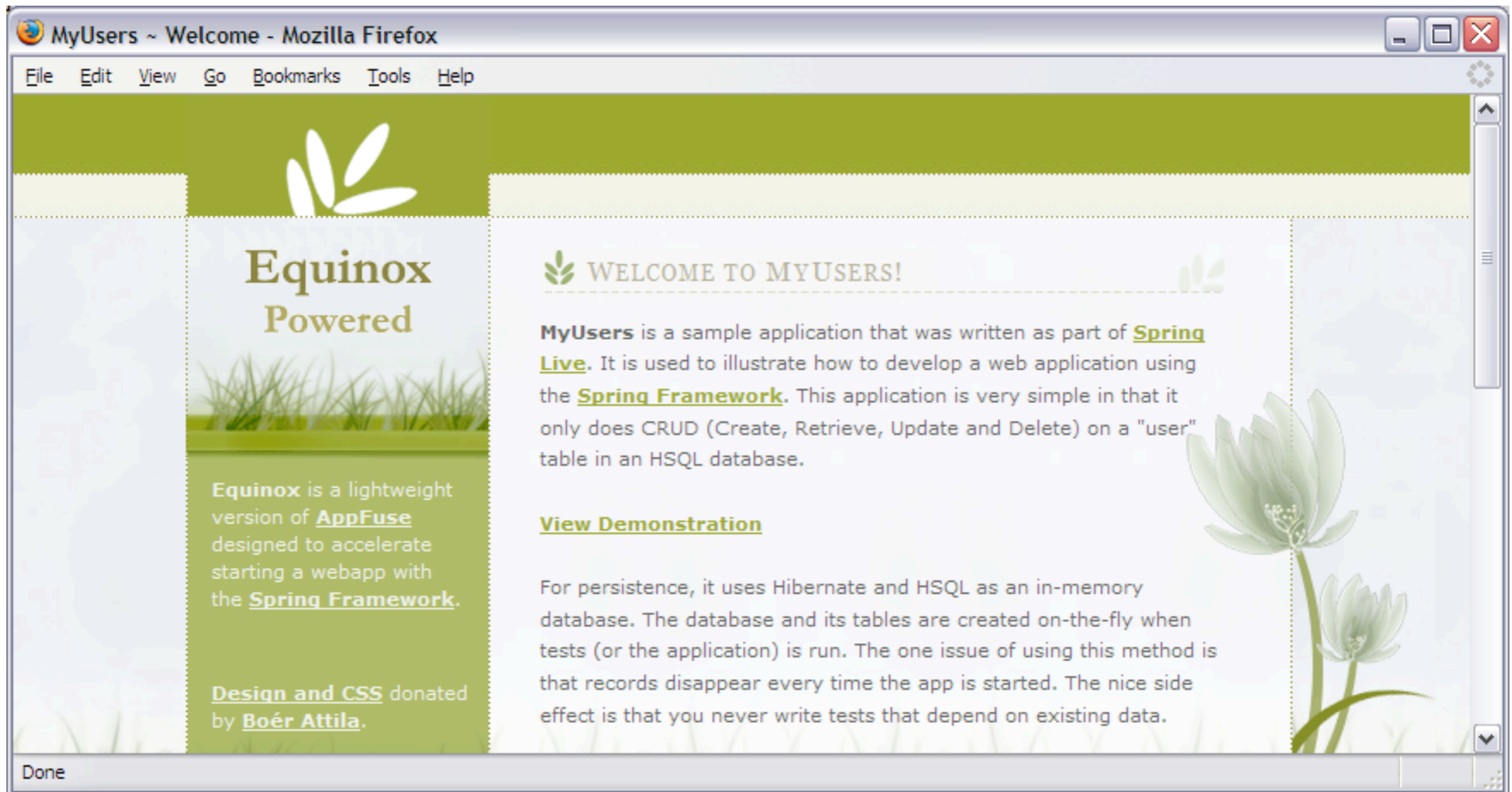
    <div id="content">
        <%@ include file="/messages.jsp"%>
        <decorator:body/>
    </div>

</body>
</html>
```

Before SiteMesh



After SiteMesh



Exception Handling

- **action-servlet.xml:**

```
<bean id="exceptionResolver"
      class="org.springframework.web.servlet.handler.SimpleMappingExceptionResolver">
  <property name="exceptionMappings">
    <props>
      <prop key="org.springframework.dao.DataAccessException">
        dataAccessFailure
      </prop>
    </props>
  </property>
</bean>
```

- **dataAccessFailure.jsp:**

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

<h3>Data Access Failure</h3>

<p>${requestScope.exception.message}</p>


<p><a href="<c:url value='/' />">#171; Home</a></p>
```

Resources

- **Web:** www.springframework.org and forum.springframework.org
- **Print:** Spring Live, Pro Spring, Spring in Action, Professional Spring Development, Expert Spring MVC and Web Flow
- **Examples:** JPetstore, Petclinic, AppFuse Equinox

Questions?

trisberg@springdeveloper.com
mraible@virtuas.com

 Spring Forward 2006