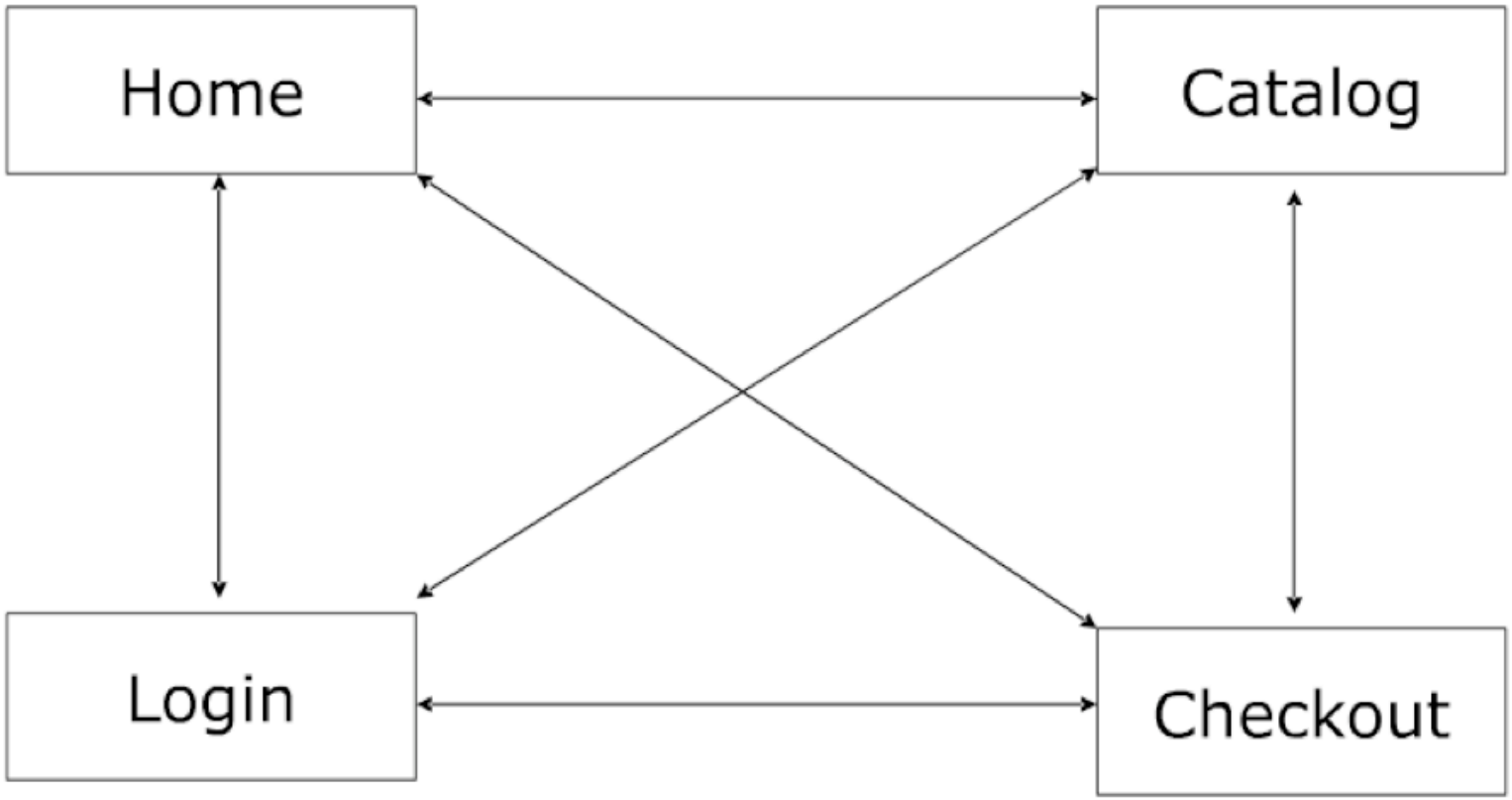
Model 2 MVC

http://www.javaworld.com/javaworld/jw-12-1999/jw-12-ssj-jspmvc.html

Life without MVC



Caos

With Mediator Pattern

Better!!!

Mediator Pattern

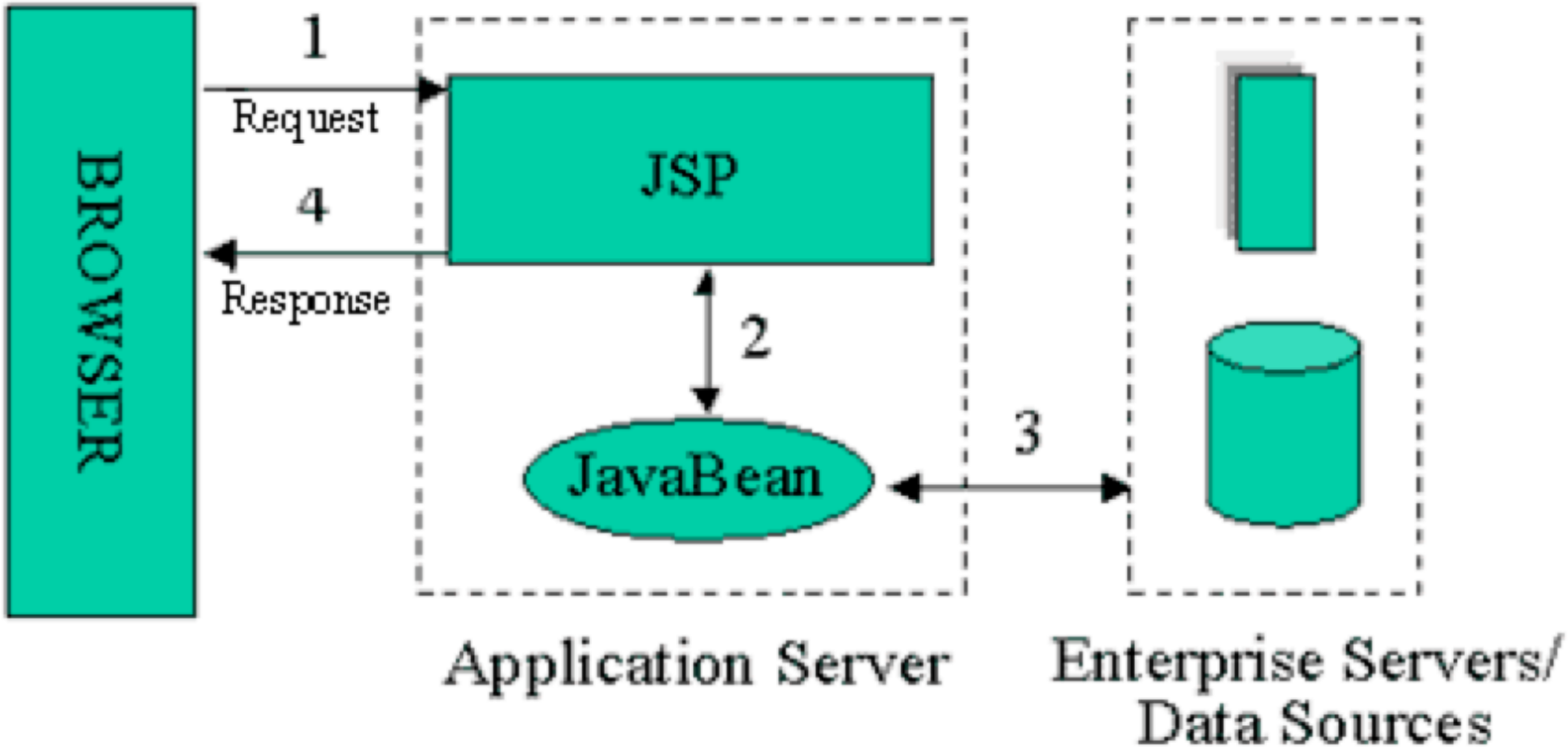
The mediator pattern provides a unified interface to a set of interfaces in a subsystem. This pattern is considered to be a behavioral pattern due to the way it can alter the program's running behavior.

A program may be made up of a large number of classes. The logic and computation is distributed among these classes. The problem of communication between these classes may become more complex. This makes the program harder to read and maintain: any change may affect code in several other classes.

With the mediator pattern, communication between objects is encapsulated with a mediator object. Objects no longer communicate directly with each other, but instead communicate through the mediator. This reduces the dependencies between communicating objects, thereby lowering the coupling.

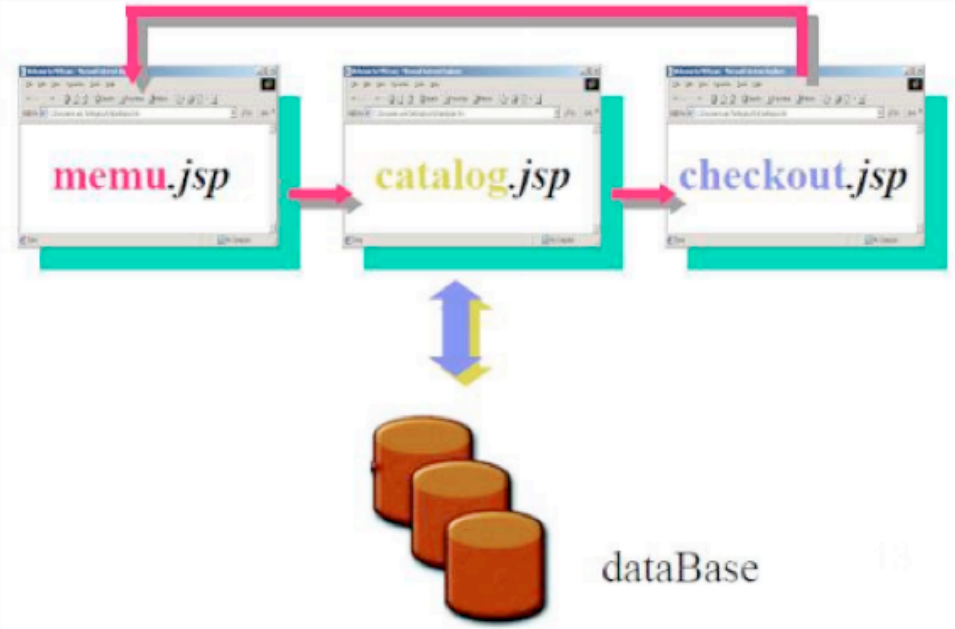
From Wikipedia

MVC model 1

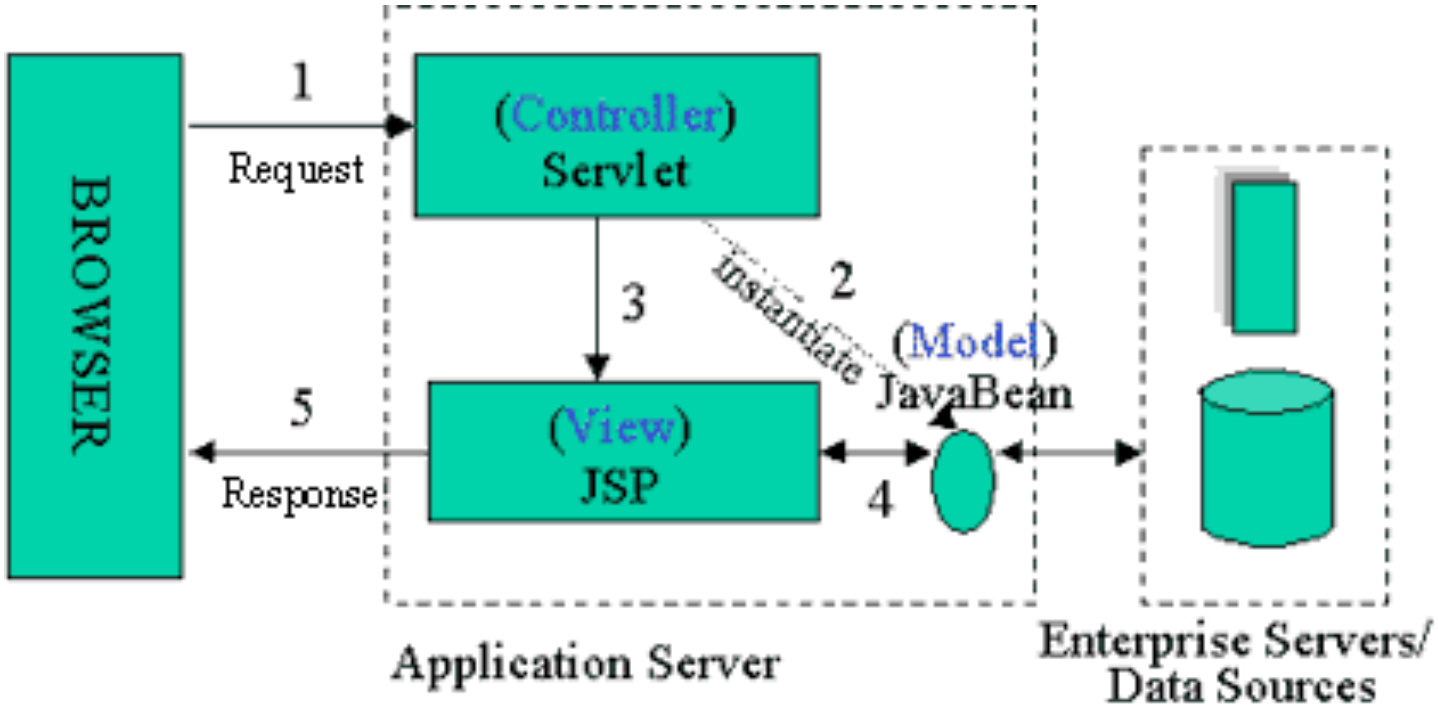

BROWSER
1
Request
4
Response
JSP
2
JavaBean
3
Application Server
Enterprise Servers/
Data Sources

Shortcomings of MVC model 1

- The model is page - centric
- Difficult of code maintenaince
- Mixing of HTML + Java code is a problem



MVC model 2

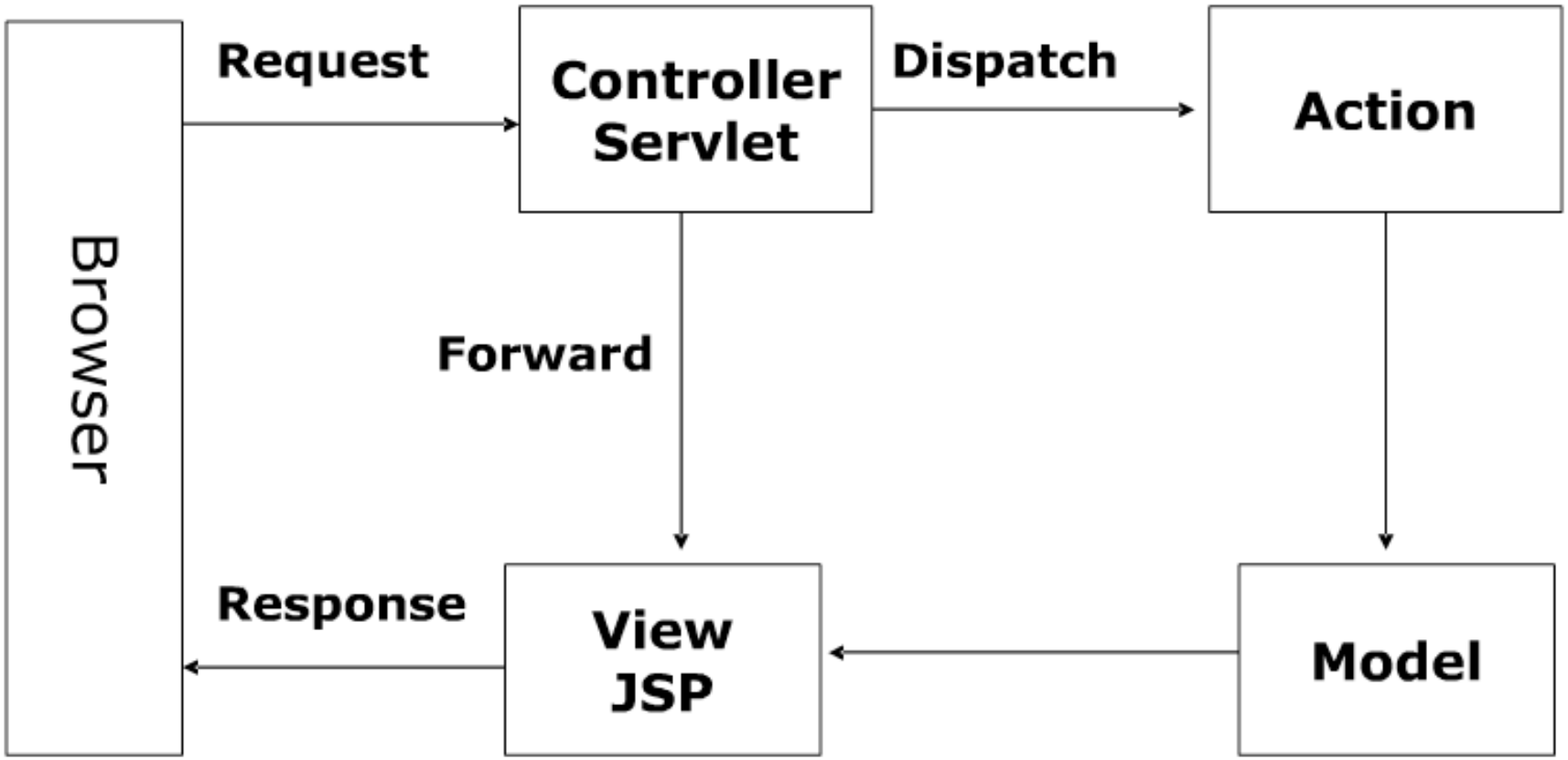

BROWSER
1
Request
(Controller)
Servlet
2
instantiate
3
(Model)
JavaBean
5
Response
(View)
JSP
4
Application Server
Enterprise Servers/
Data Sources

Benefits of MVC model 2

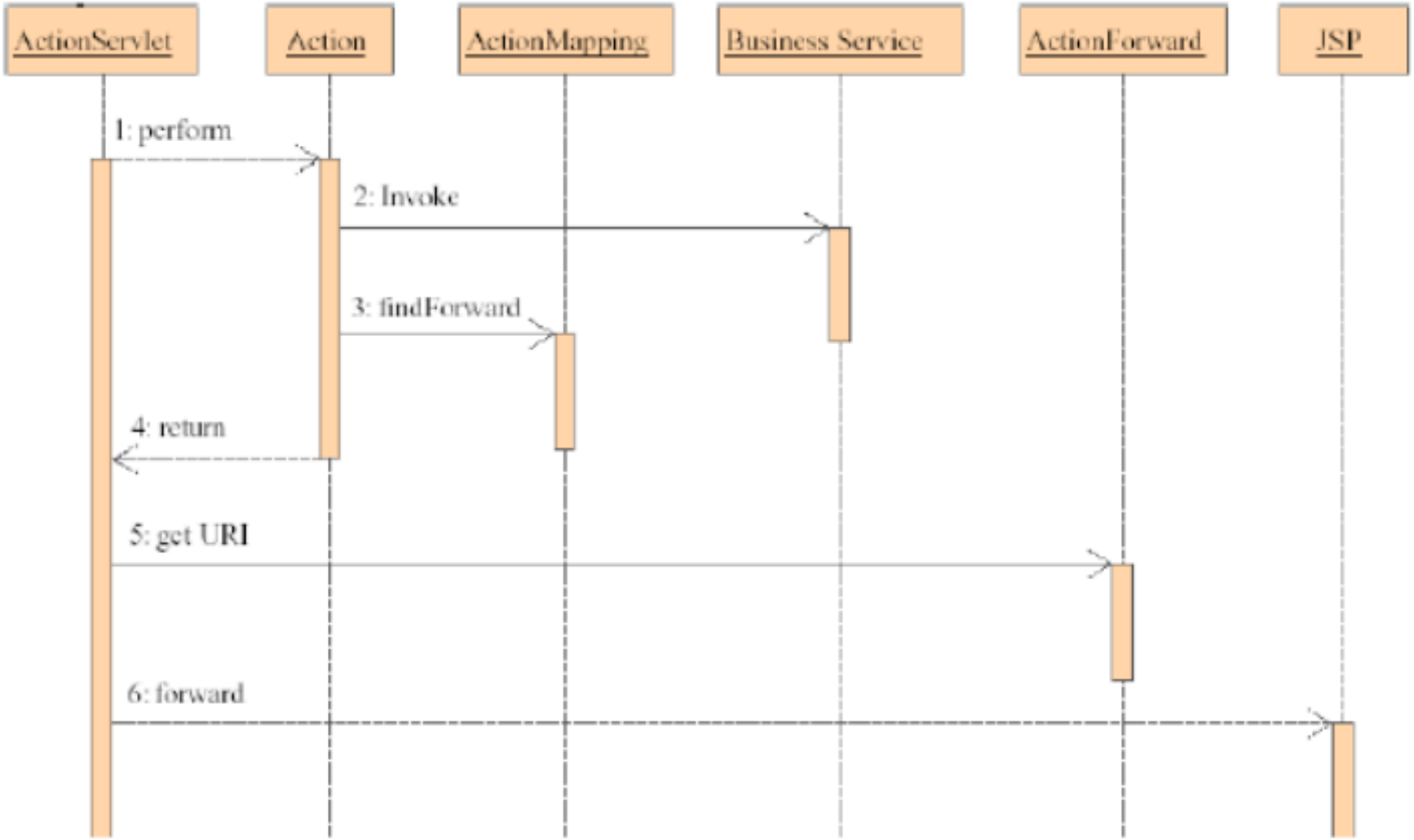
- rapid to build
- easier to test
- easier to maintain
- easier to extend

- By spliting the work into an ActionManager and a ViewManager you can get a clearer and more maintainable code
- Framweorks as Struts and Struts2 have already built-in action and view managers making easy for us the development

Struts1 implementation of MVC2



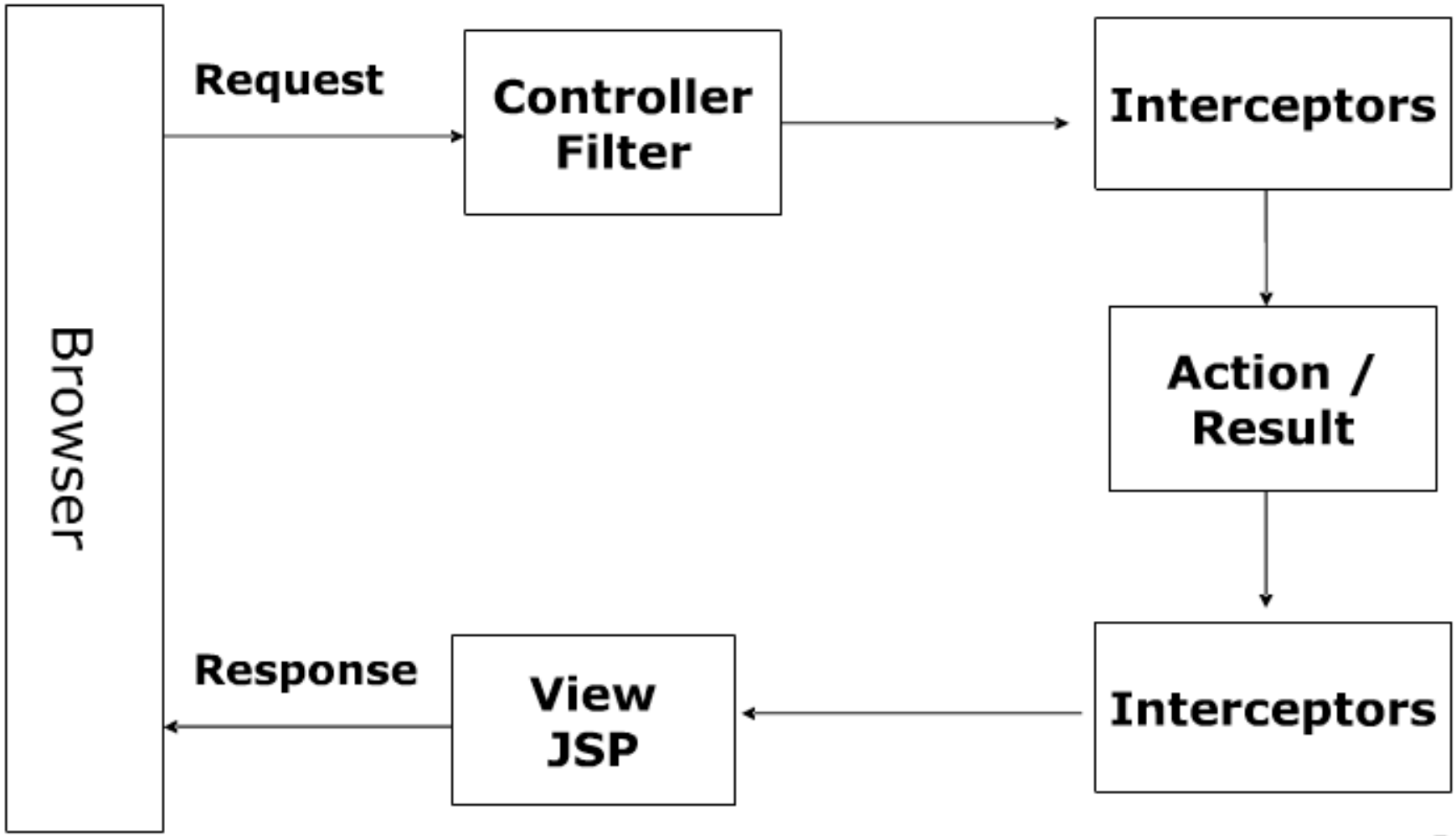
Struts1 implementation of MVC2



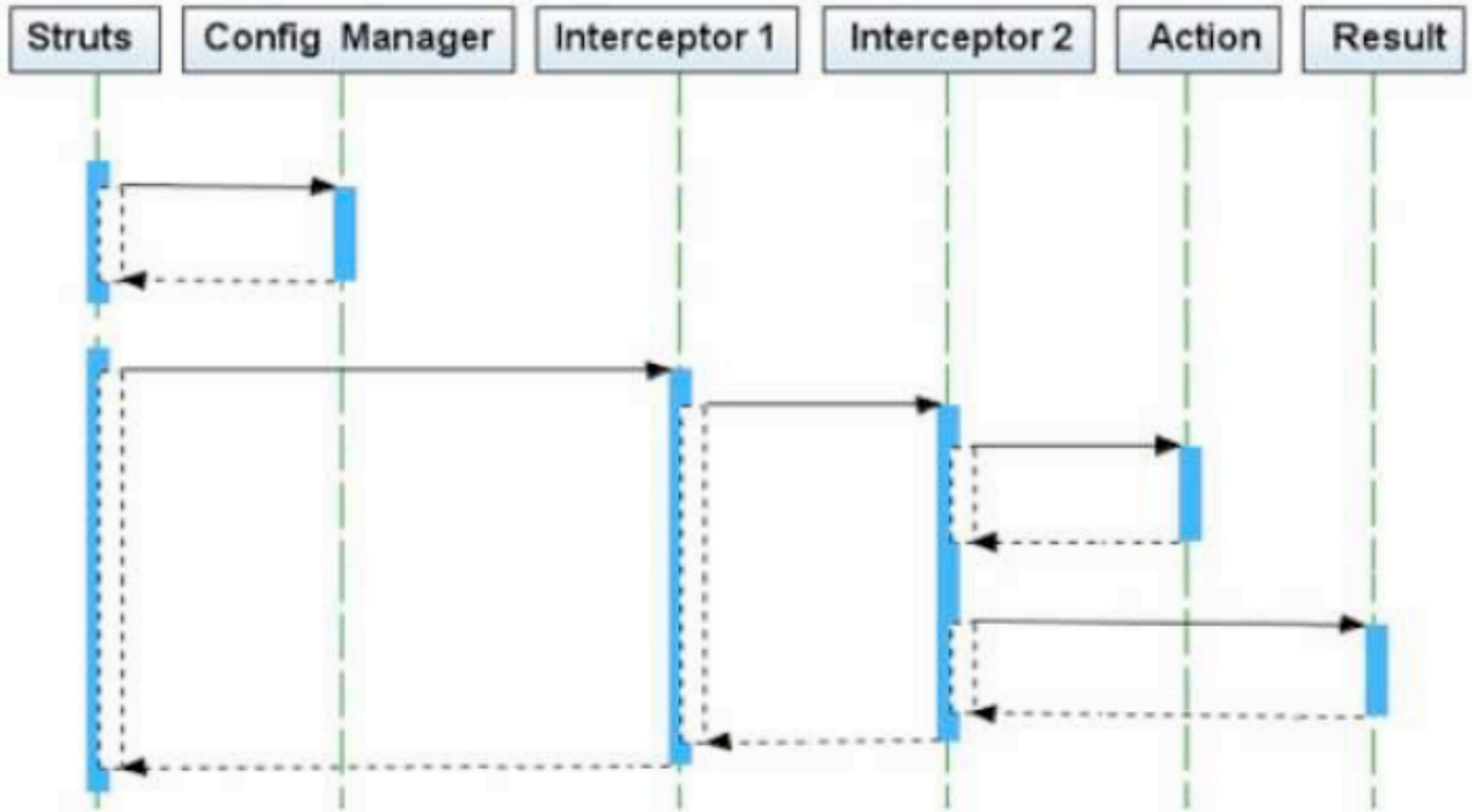
Struts1 implementation of MVC2

- Advantages of struts (1 or 2)
  - page navigation management
  - user input validation
  - consistent layout
  - extensibility
- If you uses Struts you should stick to the following unwritten rules:
  - No Java code in JSPs, all business logic should reside in Java classes called action classes.
  - Use the Expression Language (if you are using JSP 2.0) or JSTL in JSPs to access business model.
  - Little or no writing of custom tags (because they are hard to code).

Struts2 implementation of MVC2

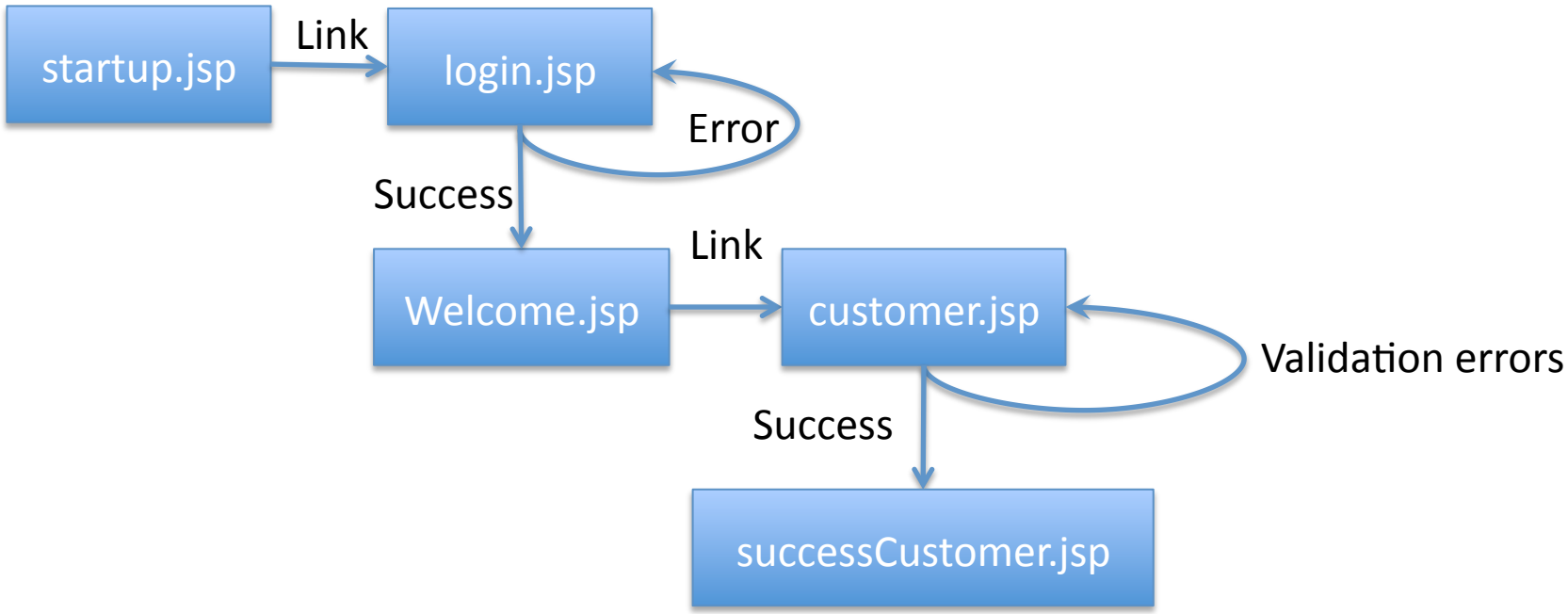


Struts2 implementation of MVC2



Introduction to Struts2

A miniapp


startup.jsp
Link
login.jsp
Error
Success
Welcome.jsp
Link
customer.jsp
Validation errors
Success
successCustomer.jsp

Install struts2 in Netbeans

http://plugins.netbeans.org/plugin/23467/netbeans-struts2-support

http://forums.netbeans.org/topic39893.html

For Eclipse see

http://viralpatel.net/blogs/2009/12/tutorial-create-struts-2-application-eclipse-example.html

sun-web.xml - base

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sun-web-app PUBLIC "-//Sun Microsystems, Inc.//DTD GlassFish Application Server 3.0 Servlet 3.0//EN" "http://www.sun.com/software/appserver/dtds/sun-web-app_3_0-0.dtd">
<sun-web-app error-url="">
  <context-root>/WebApplication3</context-root>
  <class-loader delegate="true"/>
  <jsp-config>
    <property name="keepgenerated" value="true">
      <description>Keep a copy of the generated servlet class' java code.</description>
    </property>
  </jsp-config>
</sun-web-app>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
  <filter>
    <filter-name>struts2</filter-name>
    <filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-class>
  </filter>
  <filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
  <session-config>
    <session-timeout>
      30
    </session-timeout>
  </session-config>
  <welcome-file-list>
    <welcome-file>startup.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

Use Struts2

Use Struts2 - how

Default page to be loaded

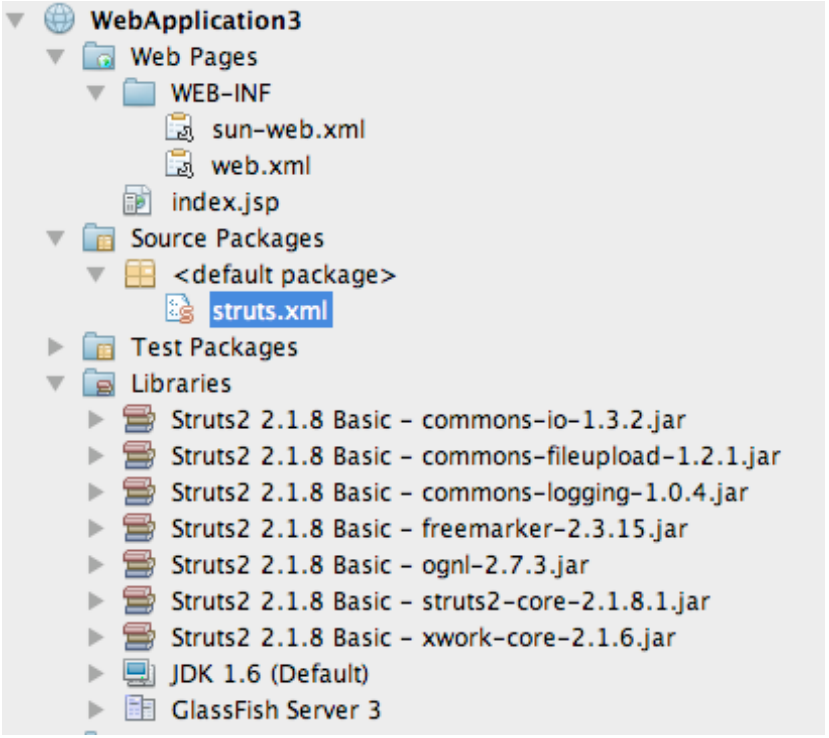
NOTE: in the generated base, the welcome file is index.jsp

struts.xml - base

```
<!DOCTYPE struts PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
"http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>
    <!-- Configuration for the default package. -->
    <package name="default" extends="struts-default">

    </package>
</struts>
```



struts.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
   "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
   "http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>
   <constant name="struts.enable.DynamicMethodInvocation"
      value="false" />
   <constant name="struts.devMode" value="false" />
   <constant name="struts.custom.i18n.resources"
      value="ApplicationResources" />

   <package name="default" extends="struts-default" namespace="/">
      <action name="login"
         class="Demo.LoginAction">
         <result name="success">Welcome.jsp</result>
         <result name="error">Login.jsp</result>
      </action>
   </package>
</struts>
```

Access the bundle

Define the controlling logic

ApplicationResources.properties

```
label.username= Username
label.password= Password
label.login= Login
error.login= Invalid Username/Password. Please try again.

name= Name
age= Age
email= Email
telephone= Telephone
label.add.customer=Add Customer

errors.invalid=${getText(fieldName)} is invalid.
errors.required=${getText(fieldName)} is required.
errors.number=${getText(fieldName)} must be a number.
errors.range=${getText(fieldName)} is not in the range ${min} and ${max}
```

Startup.jsp

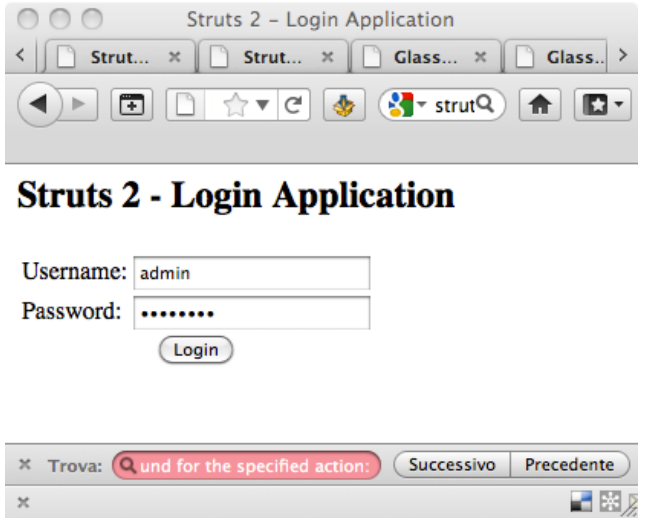

```
<%@page contentType="text/html"
    pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Hello. Please go to the <a href="Login.jsp">registration</a></h1>
    </body>
</html>
```

Login.jsp

```
<%@ page contentType="text/html; charset=UTF-8"%>
<%@ taglib prefix="s" uri="/struts-tags"%>
<html>
<head>
<title>Struts 2 - Login Application </title>
</head>

<body>
<h2>Struts 2 - Login Application</h2>
<s:actionerror />
<s:form action="login" namespace="/" method="get">
   <s:textfield name="username" key="label.username" size="20" />
   <s:password name="password" key="label.password" size="20" />
   <s:submit method="execute" key="label.login" align="center" />
</s:form>
</body>
</html>
```



Data from the bundle

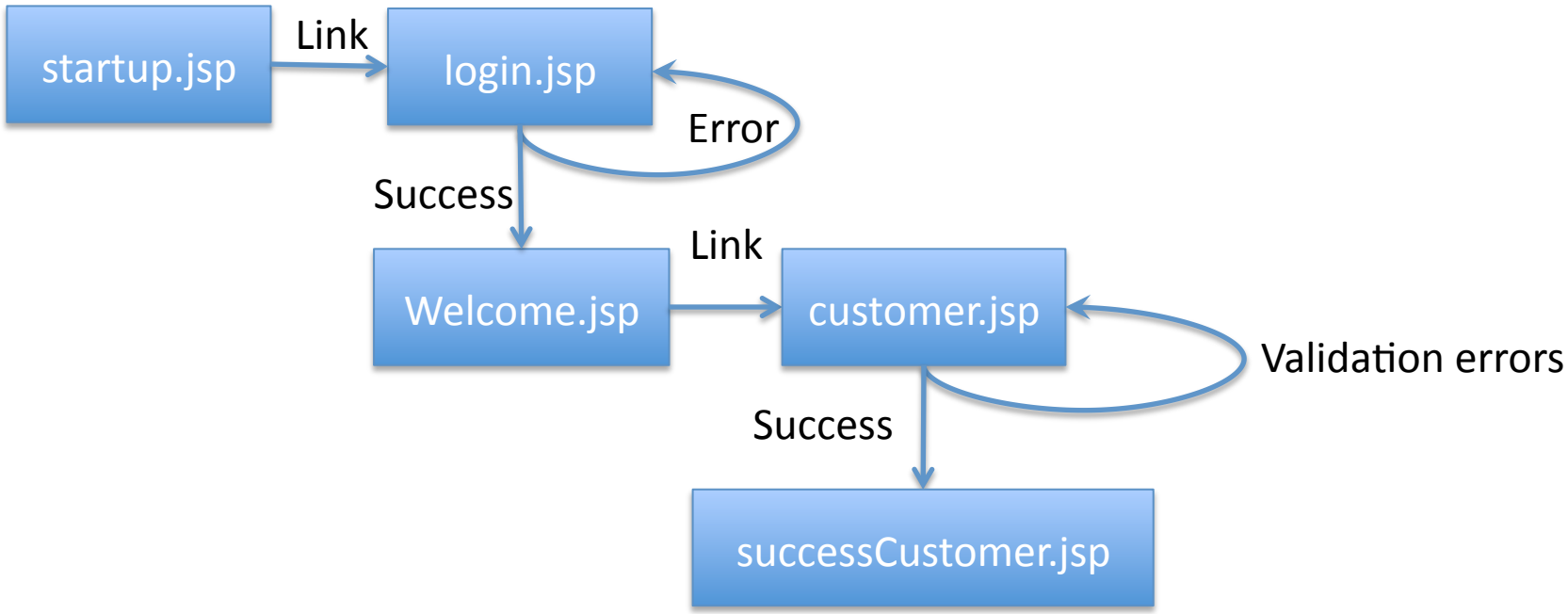
ApplicationResources.properties

```
label.username= Username
label.password= Password
label.login= Login
error.login= Invalid Username/Password. Please try again.

name= Name
age= Age
email= Email
telephone= Telephone
label.add.customer=Add Customer

errors.invalid=${getText(fieldName)} is invalid.
errors.required=${getText(fieldName)} is required.
errors.number=${getText(fieldName)} must be a number.
errors.range=${getText(fieldName)} is not in the range ${min} and ${max}
```

A miniapp


startup.jsp
Link
login.jsp
Error
Success
Welcome.jsp
Link
customer.jsp
Validation errors
Success
successCustomer.jsp

struts.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
   "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
   "http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>
   <constant name="struts.enable.DynamicMethodInvocation"
      value="false" />
   <constant name="struts.devMode" value="false" />
   <constant name="struts.custom.i18n.resources"
      value="ApplicationResources" />
   <package name="Demo" namespace="/" extends="struts-default" >
   <action name="login"
         class="Demo.LoginAction">
         <interceptor-ref name="loggingStack"></interceptor-ref>
         <result name="success">Welcome.jsp</result>
         <result name="error">Login.jsp</result>
      </action>
</package >
</struts>
```

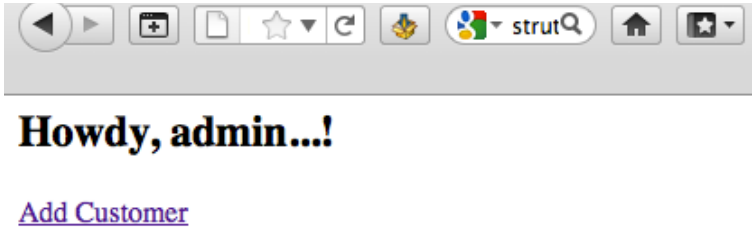
LoginAction.java

```
package Demo;
import com.opensymphony.xwork2.ActionSupport;
public class LoginAction extends ActionSupport {
   private String username;
   private String password;
   public String execute() {
      if (this.username.equals("admin")
            && this.password.equals("admin123")) {
         return "success";
      } else {
         addActionError(getText("error.login"));
         return "error";
      }
   }
   public String getUsername() {return username;}
   public void setUsername(String username) {this.username = username; }
   public String getPassword() {return password; }
   public void setPassword(String password) {this.password = password; }
}
```

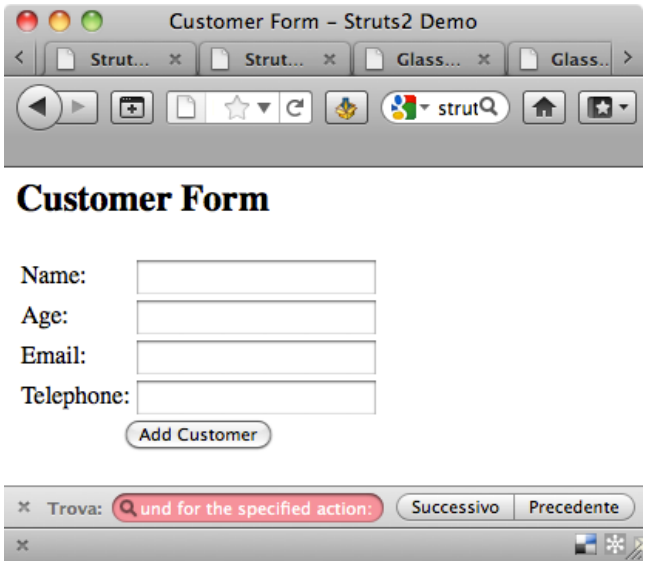
Welcome.jsp

```
<%@ page contentType="text/html;
charset=UTF-8"%>
<%@ taglib prefix="s" uri="/struts-tags"%>
<html>
<head>
<title>Welcome</title>
</head>

<body>
   <h2>Howdy, <s:property value="username" />...!
</h2>
   <s:a href="Customer.jsp">Add Customer</s:a>
</body>
</html>
```



Customer.jsp




```
<%@ page contentType="text/html; charset=UTF-8"%>
<%@ taglib prefix="s" uri="/struts-tags"%>
<html>
<head>
<title>Customer Form - Struts2 Demo </title>
</head>

<body>
<h2>Customer Form</h2>

<s:form action="customer" namespace="/" method="post" validate="false">
   <s:textfield name="name" key="name" size="20" />
   <s:textfield name="age" key="age" size="20" />
   <s:textfield name="email" key="email" size="20" />
   <s:textfield name="telephone" key="telephone" size="20" />
   <s:submit method="addCustomer" key="label.add.customer" align="center" />
</s:form>
</body>
</html>
```

struts.xml

```
<package name="Demo" namespace="/" extends="struts-default" >
    …
     <action name="customer"
        class="Demo.CustomerAction">
        <result name="success">SuccessCustomer.jsp</result>
        <result name="input">Customer.jsp</result>
     </action>
   </package >
```

Demo.CustomerAction

```
package Demo;
import com.opensymphony.xwork2.ActionSupport;

public class CustomerAction extends ActionSupport{
   private String name;
   private Integer age;
   private String email;
   private String telephone;

   public String addCustomer() {return SUCCESS;}

   public String getName() { return name;}
   public void setName(String name) {this.name = name; }
   public Integer getAge() {return age;}
   public void setAge(Integer age) {this.age = age;
   public String getEmail() {return email;}
   public void setEmail(String email) {this.email = email;}
   public String getTelephone() {return telephone; }
   public void setTelephone(String telephone) {this.telephone = telephone; }
}
```

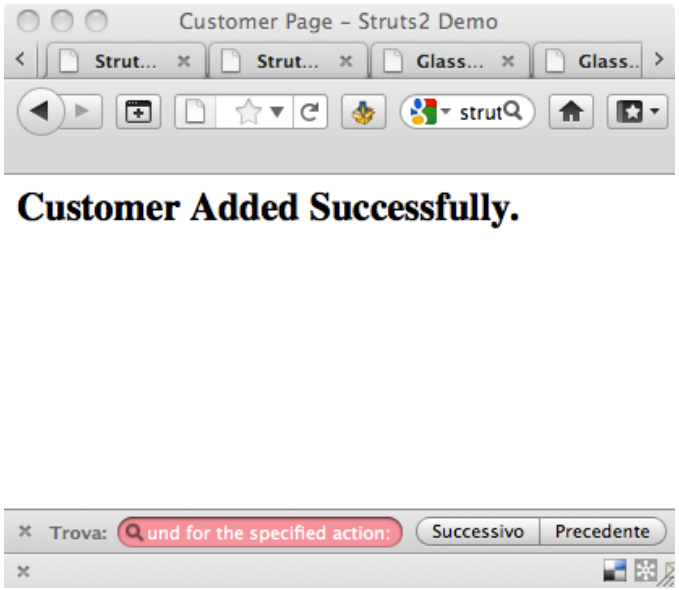
CustomerAction-validation.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE validators PUBLIC
      "-//OpenSymphony Group//XWork Validator 1.0.2//EN"
      "http://www.opensymphony.com/xwork/xwork-validator-1.0.2.dtd">
<validators>
   <field name="name">
      <field-validator type="requiredstring">
         <param name="trim">true</param>
         <message key="errors.required" />
      </field-validator>
   </field>
   <field name="age">
      <field-validator type="required">
         <message key="errors.required" />
      </field-validator>
      <field-validator type="int">
         <param name="min">1</param>
         <param name="max">100</param>
         <message key="errors.range"/>
      </field-validator>
   </field>
   <field name="email">
      <field-validator type="requiredstring">
         <message key="errors.required" />
      </field-validator>
      <field-validator type="email">
         <message key="errors.invalid" />
      </field-validator>
   </field>
   <field name="telephone">
      <field-validator type="requiredstring">
         <message key="errors.required" />
      </field-validator>
   </field>
</validators>
```

CustomerSuccess.jsp

```
<%@ page contentType="text/html; charset=UTF-8"%>
<%@ taglib prefix="s" uri="/struts-tags"%>
<html>
<head>
<title>Customer Page - Struts2 Demo </title>
</head>

<body>
   <h2>Customer Added Successfully.</h2>
</body>
</html>
```



Adding interceptors

```
<package name="Demo" namespace="/" extends="struts-default" >
      <interceptors>
        <interceptor name="mylogging"
          class="Demo.Interceptors.MyLoggingInterceptor">
        </interceptor>
        <interceptor-stack name="loggingStack">
          <interceptor-ref name="mylogging" />
          <interceptor-ref name="defaultStack" />
        </interceptor-stack>
      </interceptors>
      <action name="login"
        class="Demo.LoginAction">
        <interceptor-ref name="loggingStack"></interceptor-ref>
        <result name="success">Welcome.jsp</result>
        <result name="error">Login.jsp</result>
      </action>
...
</package>
```

MyLoggingInterceptor.java

```
package Demo.Interceptors;
import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.interceptor.Interceptor;
public class MyLoggingInterceptor implements Interceptor{
   private static final long serialVersionUID = 1L;
   public String intercept(ActionInvocation invocation) throws Exception {
      String className = invocation.getAction().getClass().getName();
      long startTime = System.currentTimeMillis();
      System.out.println("Before calling action: " + className);
      String result = invocation.invoke();
      long endTime = System.currentTimeMillis();
      System.out.println("After calling action: " + className
            + " Time taken: " + (endTime - startTime) + " ms");
      return result;
   }
   public void destroy() { System.out.println("Destroying MyLoggingInterceptor..."); }
   public void init() {System.out.println("Initializing MyLoggingInterceptor..."); }
}
```

BEFORE

AFTER