Principles and Techniques of DBMS 9 Struts

Haopeng Chen

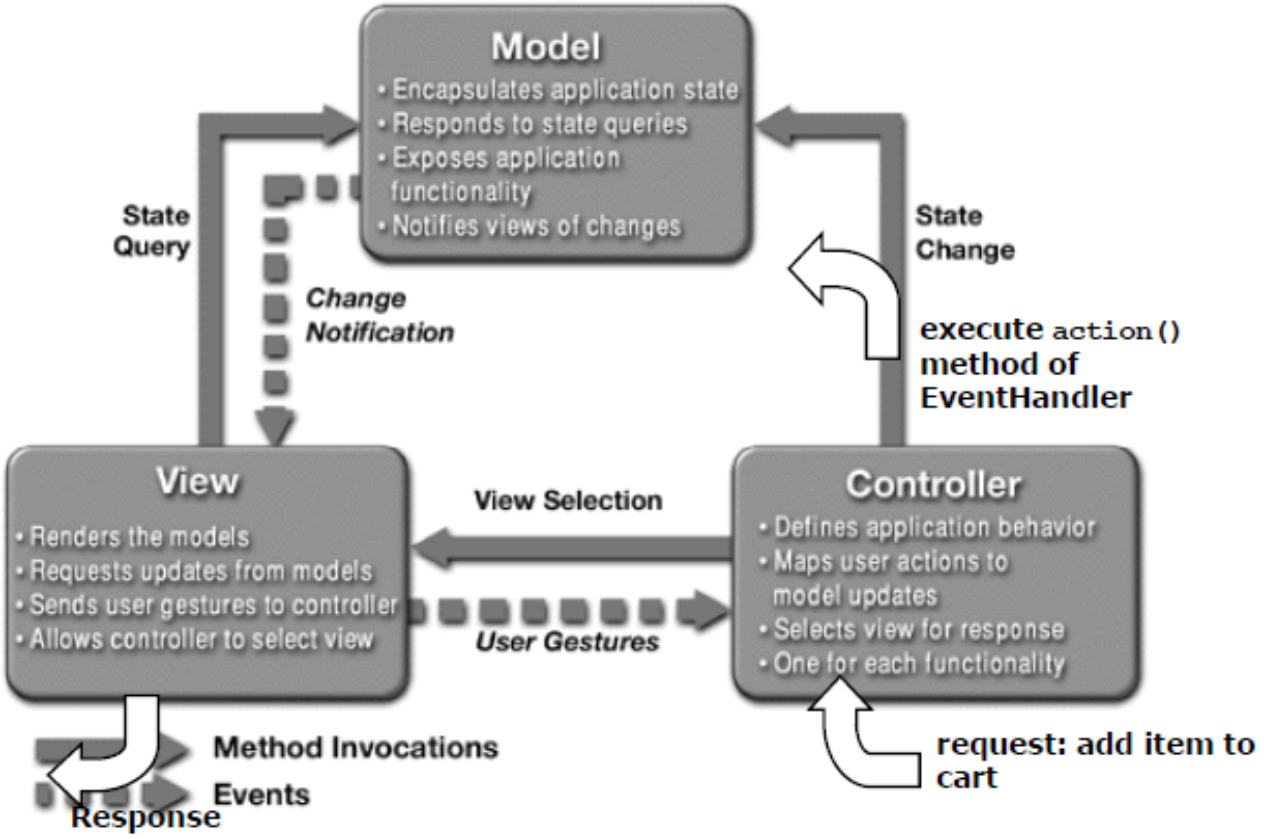
***RE**liable, **IN**telligent and **S**calable Systems Group (**REINS**)*
Shanghai Jiao Tong University
Shanghai, China
http://reins.se.sjtu.edu.cn/~chenhp
e-mail: chen-hp@sjtu.edu.cn

Overview

- Struts
  - MVC architecture

Model-View-Controller Architectures

- Model-View-Controller (MVC) is an old design pattern for user applications from Smalltalk

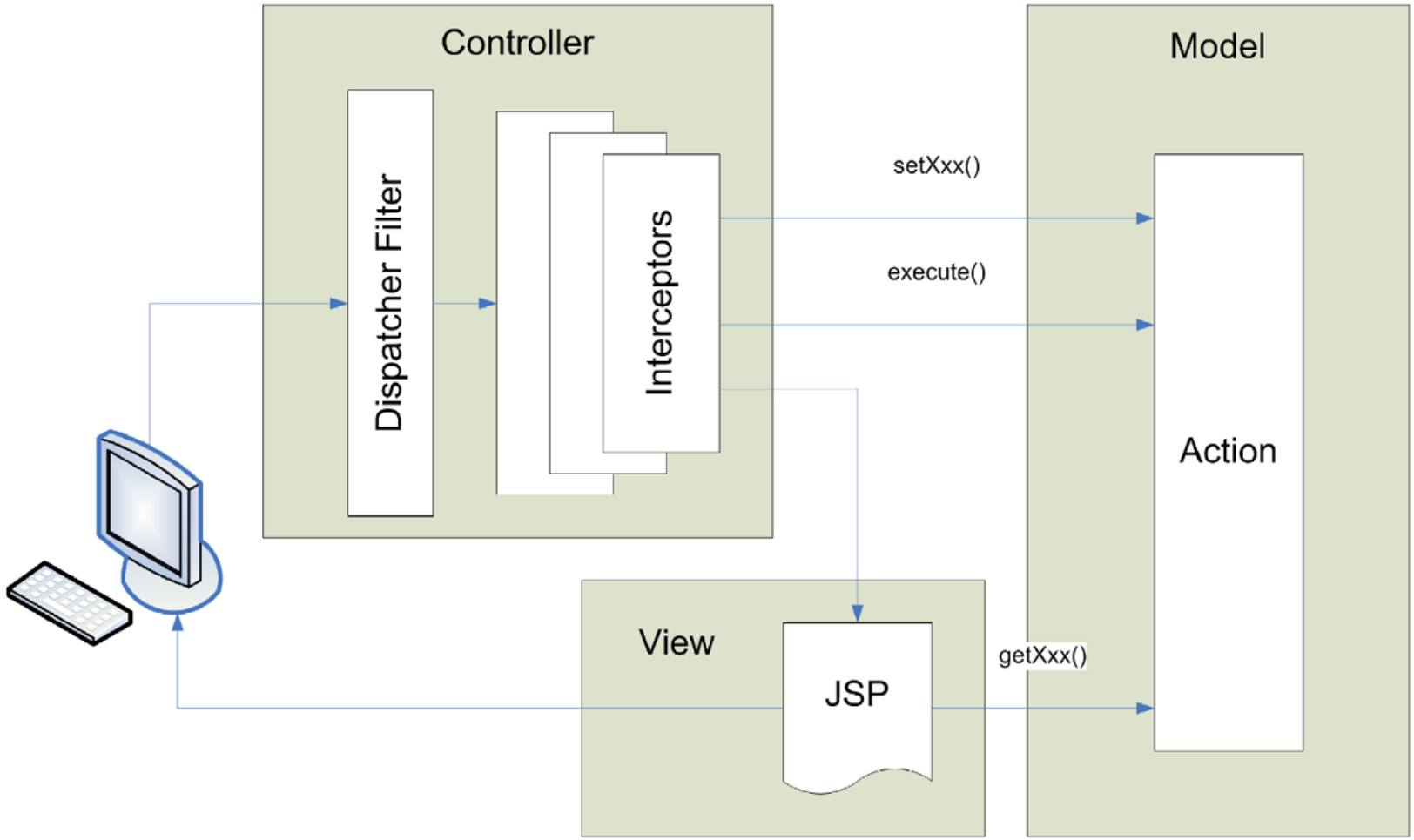


Struts framework

- Struts was originally developed by Craig McClanahan and donated to the Apache Foundation in May 2000.

- Struts has been a de facto framework with a strong and vibrant user community.

- Struts uses and extends the Java Servlet API to adopt the "Model 2" approach, a variation of the classic Model-View-Controller (MVC) design pattern.

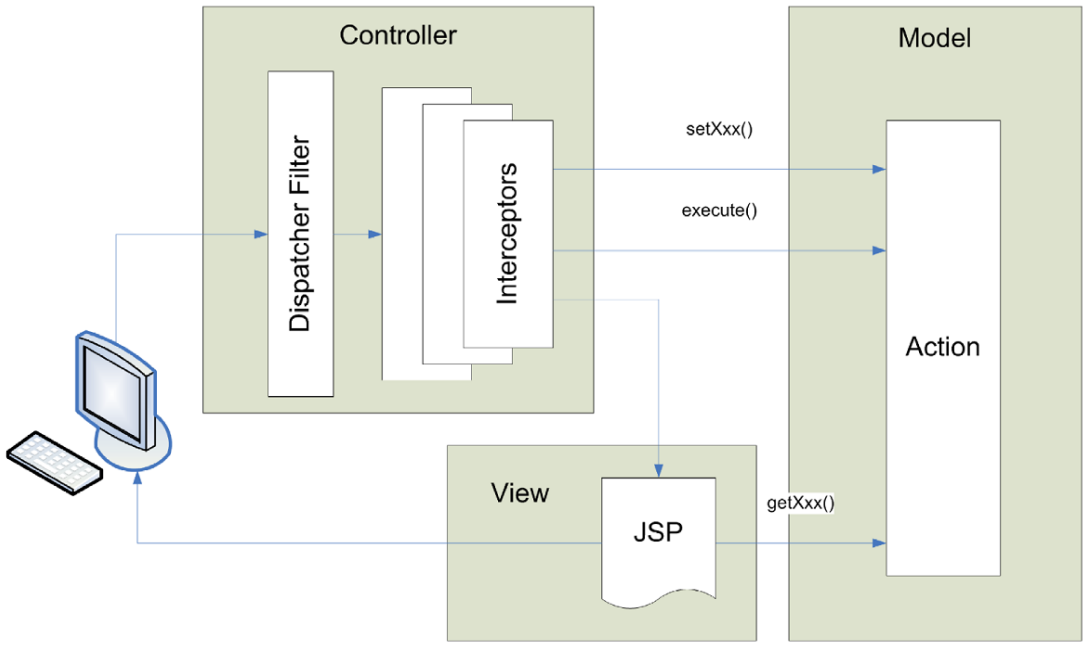

Controller
Dispatcher Filter
Interceptors
Model
setXxx()
execute()
Action
View
JSP
getXxx()

- Here are some of the features that may lead you to consider Struts2:
  - Action based MVC web framework
  - Mature with a vibrant developer and user community
  - Annotation and XML configuration options
  - POJO-based actions that are easy to test
  - Spring, SiteMesh and Tiles integration
  - OGNL expression language integration
  - Themes based tag libraries and Ajax tags
  - Multiple view options (JSP, Freemarker, Velocity and XSLT)
  - Plug-ins to extend and modify framework features

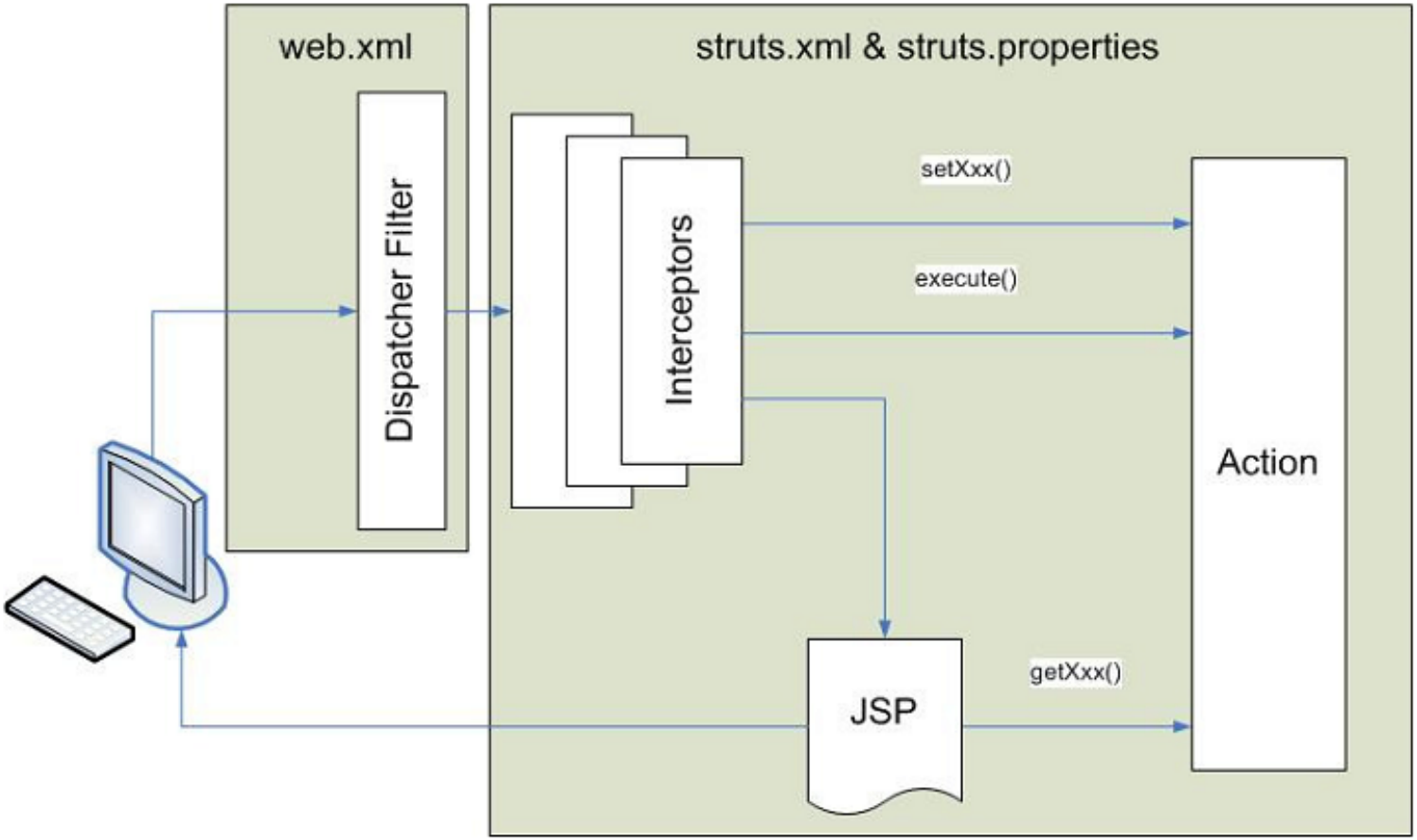
Core Components

REliable, INtelligent & Scalable Systems

- The Model-View-Controller pattern in Struts2 is realized with five core components
  - Actions: The model is implemented with actions
  - Interceptors: The controller is implemented with a Struts2 dispatch servlet filter as well as interceptors
  - Value stack / OGNL: provide common thread, linking and enabling integration between the other components.
  - Result types
  - Results / view technologies.



Configuration



Configuration

- The web application configuration for the `FilterDispatcher` servlet filter needs to be configured in your "`web.xml`" file:

```
……
<filter>
    <filter-name>action2</filter-name>
    <filter-class>
        org.apache.struts2.dispatcher.FilterDispatcher
    </filter-class>
</filter>

<filter-mapping>
    <filter-name>action2</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
……
```

Configuration

REliable, INtelligent & Scalable Systems

- The `struts.properties` File
  - This configuration file provides a mechanism to change the default behavior of the framework.
  - In a development environment, there are a couple of properties that you might consider changing:
    - `struts.i18n.reload = true` – enables reloading of internationalization files
    - `struts.devMode = true` – enables development mode that provides more comprehensive debugging
    - `struts.configuration.xml.reload = true` – enables reloading of XML configuration files (for the action) when a change is made without reloading the entire web application in the servlet container
    - `struts.url.http.port = 8080` – sets the port that the server is run on (so that generated URLs are created correctly)

Configuration

- The `struts.xml` File
  - This configuration file contains the configuration information that you will be modifying as actions are developed.

```
<?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.devMode" value="true" />
  <package name="basicstruts2" extends="struts-default">
    <action name="index">
      <result>/index.jsp</result>
    </action>
    ……
  </package>
</struts>
```

Configuration

REliable, INtelligent & Scalable Systems

- The struts.xml File
  - The Include Tag.
    - is used to modularize a Struts2 application by including other configuration files and is always a child of the <struts> tag.

```
<struts>
      <include file="billing-config.xml" />
      <include file="admin-config.xml" />
      <include file="reports-config.xml" />
    …
</struts>
```

Configuration

REliable, INtelligent & Scalable Systems

- The `struts.xml` File
  - The `Package` Tag.
    - is used to group together configurations that share common attributes such as interceptor stacks or URL namespaces.
    - The attributes for this tag are:
      - name – a developer provided unique name for this package
      - extends – the name of a package that this package will extend
      - namespace – the namespace provides a mapping from the URL to the package.
      - abstract – if this attribute value is "true" the package is truly configuration grouping, and actions configured will not be accessible via the package name

Actions

- Actions are a fundamental concept in most web application frameworks, and they are the most basic unit of work that can be associated with a HTTP request coming from a user.
- Single result

```
class MyAction {
     public void String execute() throws Exception {
        return "success";
    }
}
```

```
<action name="my" class="com.fdar.infoq.MyAction" >
    <result>view.jsp</result>
</action>
```

Actions

REliable, INtelligent & Scalable Systems

- Multiple results

```
class MyAction {
     public void String execute() throws Exception {
        if( myLogicWorked() ) {
            return "success";
        } else {
            return "error";
        }
    }
}
<action name="my" class="com.fdar.infoq.MyAction" >
    <result>view.jsp</result>
    <result name="error">error.jsp</result>
</action>
```

Interceptors

REliable, INtelligent & Scalable Systems

- Interceptors are conceptually the same as servlet filters or the JDKs Proxy class.
  - They provide a way to supply pre-processing and post-processing around the action.

```
<interceptors>
    …
    <interceptor name="autowiring"
        class="interceptor.ActionAutowiringInterceptor"/>
</interceptors>

<action name="my" class="com.fdar.infoq.MyAction" >
    <result>view.jsp</result>
    <interceptor-ref name="autowiring"/>
</action>
```

Interceptors

REliable, INtelligent & Scalable Systems

- Interceptors are conceptually the same as servlet filters or the JDKs Proxy class.
  - They provide a way to supply pre-processing and post-processing around the action.

```
<interceptor-stack name="basicStack">
    <interceptor-ref name="exception"/>
    <interceptor-ref name="servlet-config"/>
    <interceptor-ref name="prepare"/>
    <interceptor-ref name="checkbox"/>
    <interceptor-ref name="params"/>
    <interceptor-ref name="conversionError"/>
</interceptor-stack>

<action name="my" class="com.fdar.infoq.MyAction" >
    <result>view.jsp</result>
    <interceptor-ref name="basicStack"/>
</action>
```

Interceptors

- Implementing Interceptors

```
public interface Interceptor extends Serializable {
    void destroy();
    void init();
    String intercept(ActionInvocation invocation)
        throws Exception;
}
```

Value Stack / OGNL

REliable, INtelligent & Scalable Systems

- The value stack is exactly what it says it is – a stack of objects.
- OGNL stands for Object Graph Navigational Language, and provides the unified way to access objects within the value stack.
  - Temporary Objects
  - The Model Object
  - The Action Object
  - Named Objects
- Accessing the value stack can be achieved in many different ways.

Return Types

REliable, INtelligent & Scalable Systems

```
<action name="my" class="com.fdar.infoq.MyAction" >
    <result type="dispatcher">view.jsp</result>
</action>

<result-types>
    <result-type name="dispatcher" default="true"
        class="….dispatcher.ServletDispatcherResult"/>
    <result-type name="redirect"
        class="….dispatcher.ServletRedirectResult"/>
    …
</result-types>
```

- To create a new result type, implement the Result interface.

```
public interface Result extends Serializable {
     public void execute(ActionInvocation invocation)
        throws Exception;
}
```

Results / View Technologies

REliable, INtelligent & Scalable Systems

- There are three other technologies that can replace JSPs in a Struts2 application:
  - Velocity Templates
  - Freemarker Templates
  - XSLT Transformations

```
<action name="my" class="com.fdar.infoq.MyAction" >
    <result type="freemarker">view.ftl</result>
</action>

<result type="xslt">
  <param name="stylesheetLocation">render.xslt</param>
  <param name="exposedValue">model.address</param>
</result>
```

Hello World Using Struts 2

REliable, INtelligent & Scalable Systems

- To create a "Hello World" example, you need to do four things:
  - Create a class to store the welcome message (the model)
  - Create a server page to present the message (the view)
  - Create an Action class to control the interaction between the user, the model, and the view (the controller)
  - Create a mapping (struts.xml) to couple the Action class and view

Hello World Using Struts 2

- Step1: Create a class to store the welcome message (the model)

- Message.java

```
package helloworld.model;

public class MessageStore {

        private String message;

        public MessageStore() {
                setMessage("Hello Struts User");
        }

        public String getMessage() {
                return message;
        }

        public void setMessage(String message) {
                this.message = message;
        }

}
```

Hello World Using Struts 2

- Step 2 - Create The Action Class HelloWorldAction.java

- HelloWorld.java

```
package helloworld.action;

public class HelloWorldAction extends ActionSupport {

        private static final long serialVersionUID = 1L;

        private MessageStore messageStore;

        public String execute() throws Exception {
                messageStore = new MessageStore() ;
                return SUCCESS;
        }

        public MessageStore getMessageStore() {
                return messageStore;
        }

        public void setMessageStore(MessageStore messageStore) {
                this.messageStore = messageStore;
        }

}
```

Hello World Using Struts 2

- Step 3 - Create The View HelloWorld.jsp

- HelloWorld.jsp

```
<%@ page language="java" contentType="text/html;
         charset=ISO-8859-1"  pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags" %>
<!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=ISO-8859-1">
          <title>Hello World!</title>
     </head>
     <body>
          <h2><s:property value="messageStore.message" /></h2>
     </body>
</html>
```

Hello World Using Struts 2

- Step 4 - Add The Struts Configuration In struts.xml

- 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.devMode" value="true" />

  <package name="basicstruts2" extends="struts-default">

  <action name="index">
    <result>/index.jsp</result>
  </action>

  <action name="hello"
          class="helloworld.action.HelloWorldAction"
          method="execute">
    <result name="success">/HelloWorld.jsp</result>
  </action>

</package>

</struts>
```

Hello World Using Struts 2

REliable, INtelligent & Scalable Systems

- Step 5 - Create The URL Action

- index.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
         pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags" %>
<!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=ISO-8859-1">
    <title>Basic Struts 2 Application - Welcome</title>
   </head>
   <body>
     <h1>Welcome To Struts 2!</h1>
     <p>
         <a href="<s:url action='hello'/>">Hello World</a>
     </p>
   </body>
</html>
```

Hello World Using Struts 2

- Step 6 - Create The web.xml

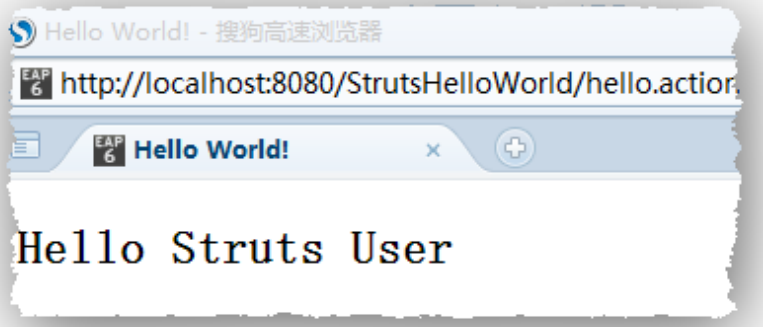
- web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_9" version="2.4"
         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">
    <display-name>Struts Blank</display-name>
    <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>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
</web-app>
```

Hello World Using Struts 2

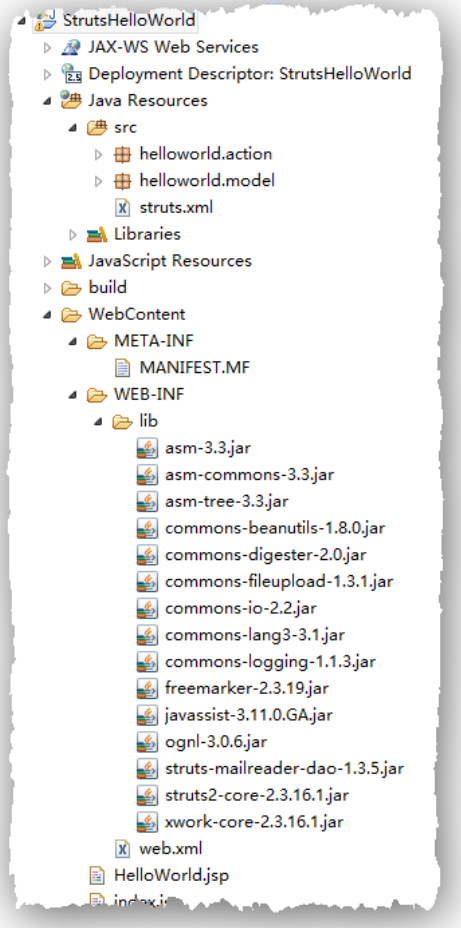
- Step 7 - Build the WAR File and Run The Application

- http://localhost:8080/StrutsHelloWorld/



Hello World Using Struts 2

- You need to add the necessary .jar to `/WEB-INF/lib/`
- The `struts.xml` is in the root of `src`
- The `web.xml` is in `/WEB-INF/`



Using Struts 2 Tags

REliable, INtelligent & Scalable Systems

- index.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
         pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
        "http://www.w3.org/TR/html4/loose.dtd">
<html>
   <head>……</head>
   <body>
     <h1>Welcome To Struts 2!</h1>
     <p><a href="<s:url action='hello'/>">Hello World</a></p>

     <s:url action="hello" var="helloLink">
       <s:param name="userName">Bruce Phillips</s:param>
     </s:url>
     <p><a href="${helloLink}">Hello Bruce Phillips</a></p>

     <p>Get your own personal hello by filling out and submitting this form.</p>
     <s:form action="hello">
       <s:textfield name="userName" label="Your name" />
       <s:submit value="Submit" />
     </s:form>
   </body>
</html>
```

Using Struts 2 Tags

Coding Struts 2 Actions

REliable, INtelligent & Scalable Systems

- HelloWorld.java(Solution 1)

```
public class HelloWorldAction extends ActionSupport {

        private static int helloCount = 0;
        private String userName;

        public String execute() throws Exception {
             messageStore = new MessageStore();
              if (userName != null) {
                messageStore.setMessage(messageStore.getMessage() +
                                        " " + userName);
             }
             return SUCCESS;
        }

        public String getUserName() { return userName; }

        public void setUserName(String userName) {
             this.userName = userName;
        }
}
```

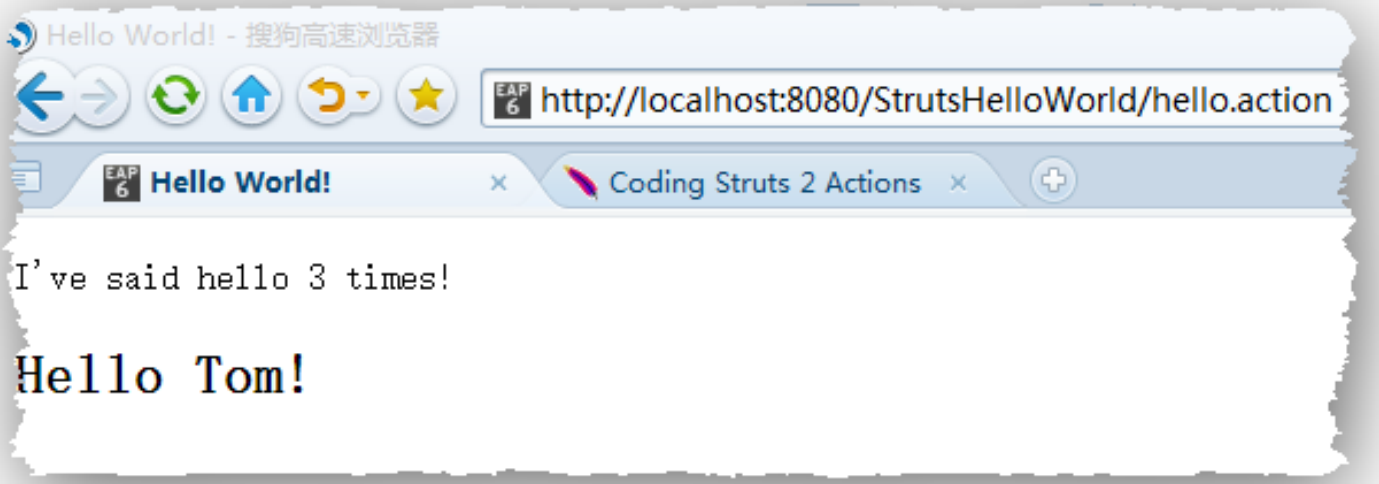
Coding Struts 2 Actions

REliable, INtelligent & Scalable Systems

- HelloWorld.java(Solution 2)

```
public class HelloWorldAction extends ActionSupport {

        private static int helloCount = 0;

        public String execute() throws Exception {
             messageStore = new MessageStore();

             ActionContext context = ActionContext.getContext();
             Map params = context.getParameters();
             String[] users = (String[]) params.get("userName");
             if (null != users){
                String userName = users[0];
                messageStore.setMessage(messageStore.getMessage() +
                                        " " + userName);
             }

             helloCount++;

             return SUCCESS;
        }
}
```

Coding Struts 2 Actions



Processing Forms

REliable, INtelligent & Scalable Systems

- index.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-
8859-1"
          pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
        "http://www.w3.org/TR/html4/loose.dtd">
<html>
   <head>……</head>
   <body>
    ……
    <p>
     <a href="register.jsp">
       Please register
     </a> for our prize drawing.
    </p>
    ……
   </body>
</html>
```

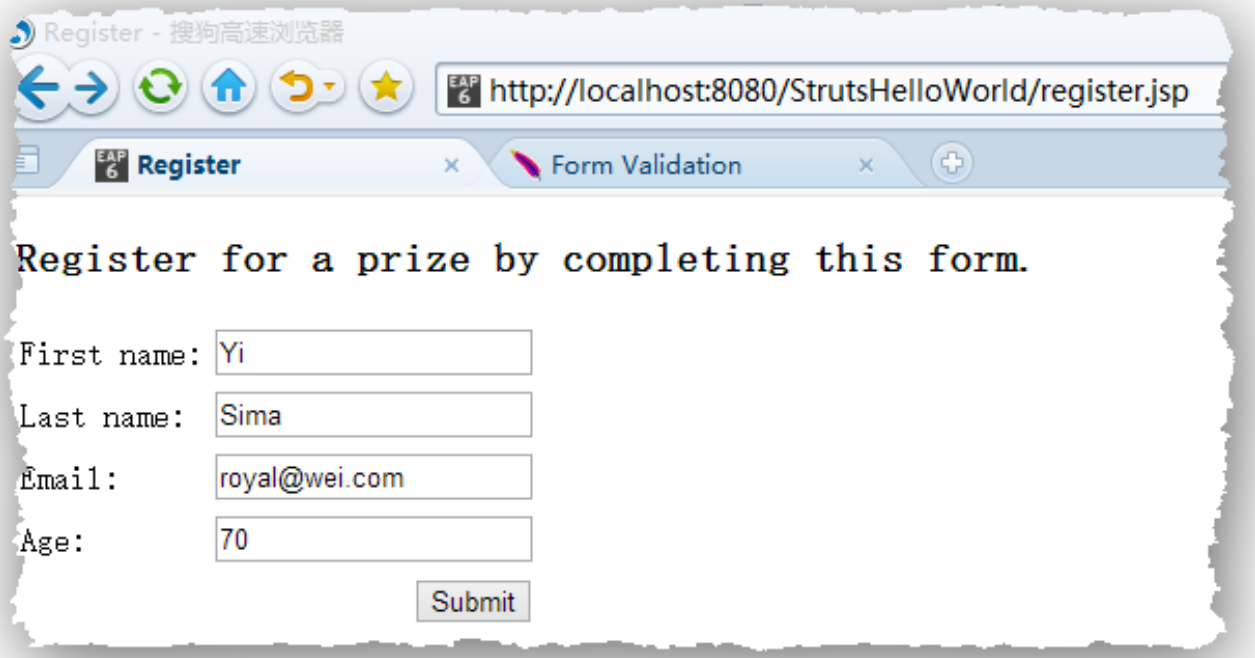
Processing Forms

REliable, INtelligent & Scalable Systems

- register.jsp

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<%@ taglib prefix="s" uri="/struts-tags"%>
……
 <head>……</head>
 <body>
  <h3>Register for a prize by completing this form.</h3>
  <s:form action="register">
   <s:textfield name="personBean.firstName" label="First name" />
   <s:textfield name="personBean.lastName" label="Last name" />
   <s:textfield name="personBean.email" label="Email" />
   <s:textfield name="personBean.age" label="Age" />
   <s:submit />
  </s:form>
 </body>
</html>
```

Processing Forms



Processing Forms

- register.java

```
public class Register extends ActionSupport {
    private User personBean;

    @Override
    public String execute() throws Exception {
       Session session = HibernateUtil.getSessionFactory().
                              getCurrentSession();
       session.beginTransaction();

       personBean.getEmailAddresses().add(personBean.getEmail());
       session.save(personBean);
       session.getTransaction().commit();

       return SUCCESS;
    }
```

Processing Forms

- User.java

```
public class User
{
    private Long id;
    private String firstName;
    private String lastName;
    private String email;
    private int age;
    private Set emailAddresses = new HashSet();

    //getters and setters
}
```

Processing Forms

REliable, INtelligent & Scalable Systems

- User.hbm.xml

```
<hibernate-mapping package="helloworld.model">
  <class name="User" table="persons">
    <id name="id" column="PERSON_ID">
        <generator class="native"/>
    </id>
    <property name="age"/>
    <property name="firstName"/>
    <property name="lastName"/>
    <set name="emailAddresses" table="PERSON_EMAIL">
        <key column="PERSON_ID"/>
        <element type="string" column="EMAIL_ADDRESS"/>
    </set>
  </class>
</hibernate-mapping>
```

Processing Forms

REliable, INtelligent & Scalable Systems

- struts.xml

```
<struts>
  ……
 <action name="register"
         class="helloworld.action.Register"
         method="execute">
   <result name="success">/thankyou.jsp</result>
 </action>
</struts>
```

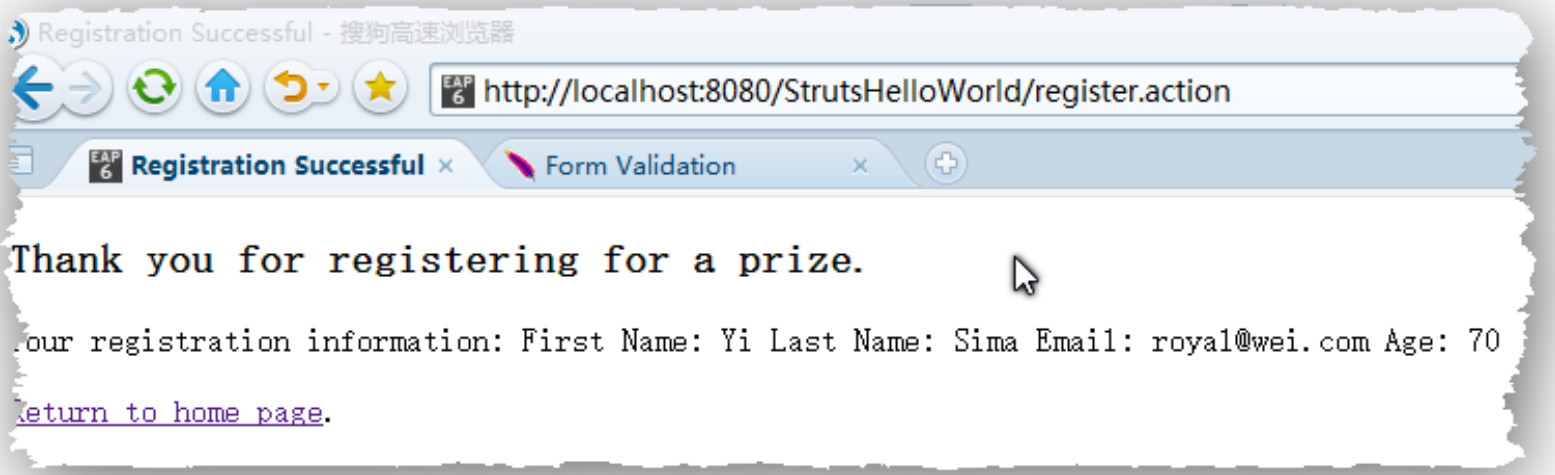
Processing Forms

REliable, INtelligent & Scalable Systems

- thankyou.jsp

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<%@ taglib prefix="s" uri="/struts-tags"%>
……
<head>……</head>
<body>
 <h3>Thank you for registering for a prize.</h3>
 <p> Your registration information:
   <s:property value="personBean" />
 </p>
 <p>
   <a href="<s:url action='index' />">Return to home page</a>.
 </p>
</body>
</html>
```

Processing Forms



Form Validation

REliable, INtelligent & Scalable Systems

- Constraints
  - User must provide a first name
  - User must provide an email address
  - User younger then 18 cannot register

- Register.java

```
public void validate(){
    if ( personBean.getFirstName().length() == 0 ){
        addFieldError( "personBean.firstName",
                       "First name is required." );
    }
    if ( personBean.getEmail().length() == 0 ){
        addFieldError( "personBean.email",
                       "Email is required." );
    }
    if ( personBean.getAge() < 18 ){
        addFieldError( "personBean.age",
                       "Age is required and must be 18 or older" );
    }
}
```

- If any errors have been added then Struts 2 will not proceed to call the execute method.
  - Rather the Struts 2 framework will return "input" as the result of calling the action.

Form Validation

REliable, INtelligent & Scalable Systems

- struts.xml

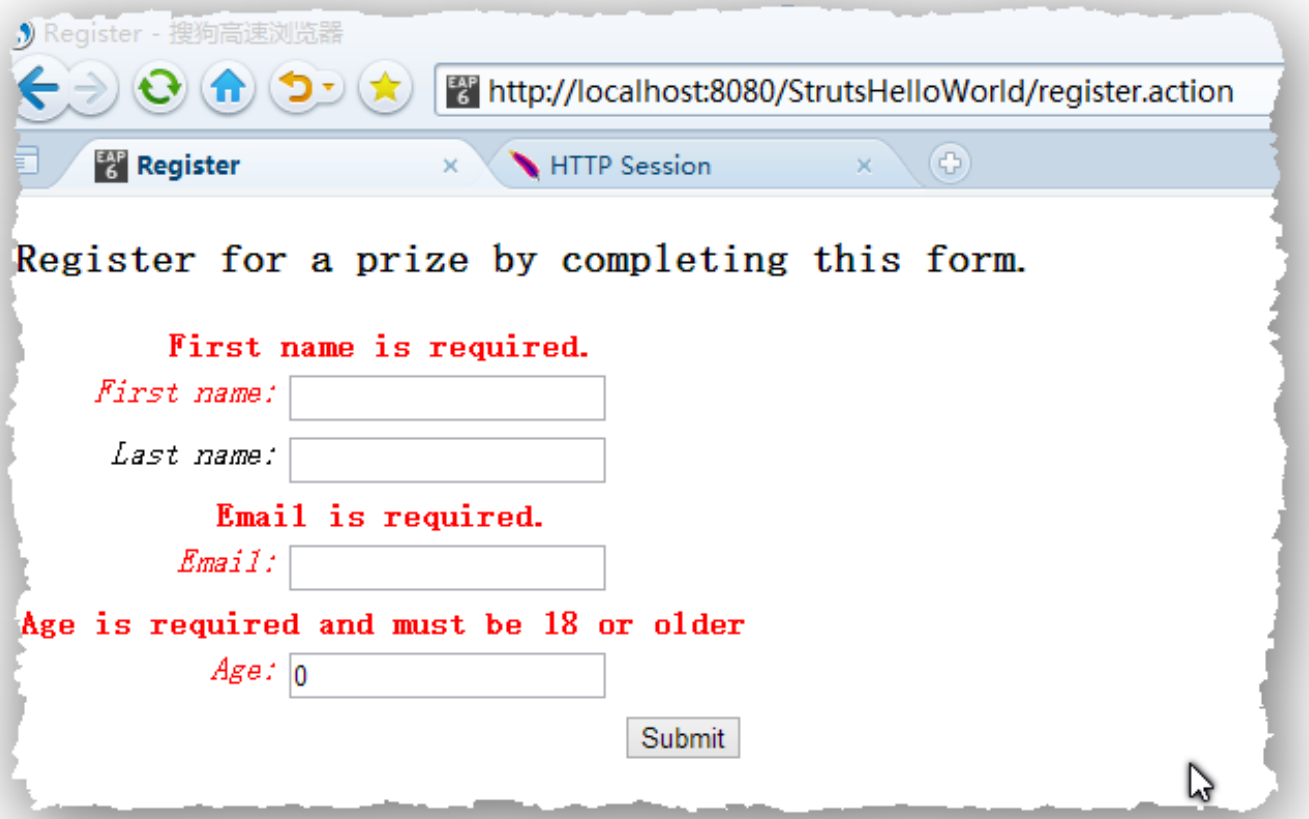
```
<struts>
  ……
 <action name="register"
         class="helloworld.action.Register"
         method="execute">
   <result name="success">/thankyou.jsp</result>
   <result name="input">/register.jsp</result>
 </action>
</struts>
```

Form Validation

- Register.jsp

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
……
<head>……</head>
<body>
  <h3>Register for a prize by completing this form.</h3>
  <s:form action="register">
   <s:textfield name="personBean.firstName" label="First name" />
   <s:textfield name="personBean.lastName" label="Last name" />
   <s:textfield name="personBean.email" label="Email" />
   <s:textfield name="personBean.age" label="Age" />
   <s:submit />
  </s:form>
 </body>
 <s:head />
</html>
```

Form Validation



Http Session

- Your Struts 2 application may need to access the HTTP session object.
  - Struts 2 provides an interface, `SessionAware`, that your Action class should implement to obtain a reference to the HTTP session object.
  - The `SessionAware` interface has one method, `setSession`, that your Action class will need to override.

- HelloWorldAction.java

```
private Map<String, Object> userSession ;

public void setSession(Map<String, Object) session) {

   userSession = session ;

}
```

Http Session

- HelloWorldAction.java

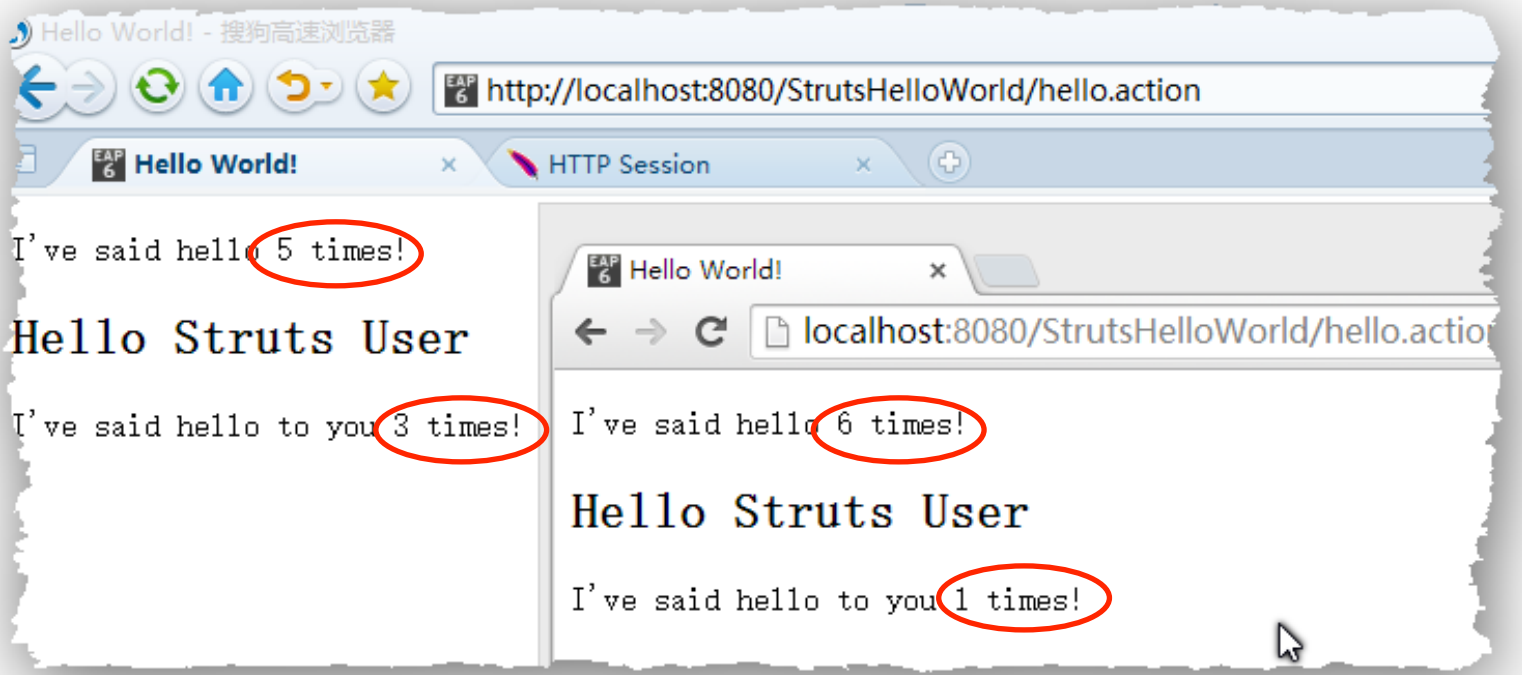
```
public String execute() throws Exception {
    ……
    increaseHelloCount();
    ……
}

private void increaseHelloCount() {

    Integer helloCount = (Integer)userSession.get("helloCount");

    if (helloCount == null ) {
      helloCount = 1;
    } else {
      helloCount++;
    }

    userSession.put("helloCount", helloCount);

}
```

Http Session

- HelloWorld.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags" %>

<!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=ISO-8859-1">
<title>Hello World!</title>
</head>
<body>
    <p>I've said hello <s:property value="helloCount" /> times!</p>
    <h2><s:property value="messageStore.message" /></h2>
    <p>I've said hello to you
     <s:property value="#session.helloCount" /> times!</p>
</body>
</html>
```

Http Session



Project

- Requirement:
  - To develop a web application: Book store.
  - User management: registration, login/logout, remove, query
  - Book selling: the registered users can browse the books, add them to the shopping cart, and pay for them.
  - Sales statistics: by user, by day/month/year, by categories

  - Tomcat server
  - Servlet + JSP
  - Points: shopping cart & statistics

  - Deadline: 11th, May 2014

  - Upload project, include source codes, necessary doc, and video if necessary, but NOT include the jar your project depends on.

References

- Apache Struts 2 Documentation: Getting Started,
  - http://struts.apache.org/release/2.3.x/docs/getting-started.html

Thank You!