Technologie Web

Un framework JEE: Struts

Alexandre Pauchet

INSA Rouen - Département ASI

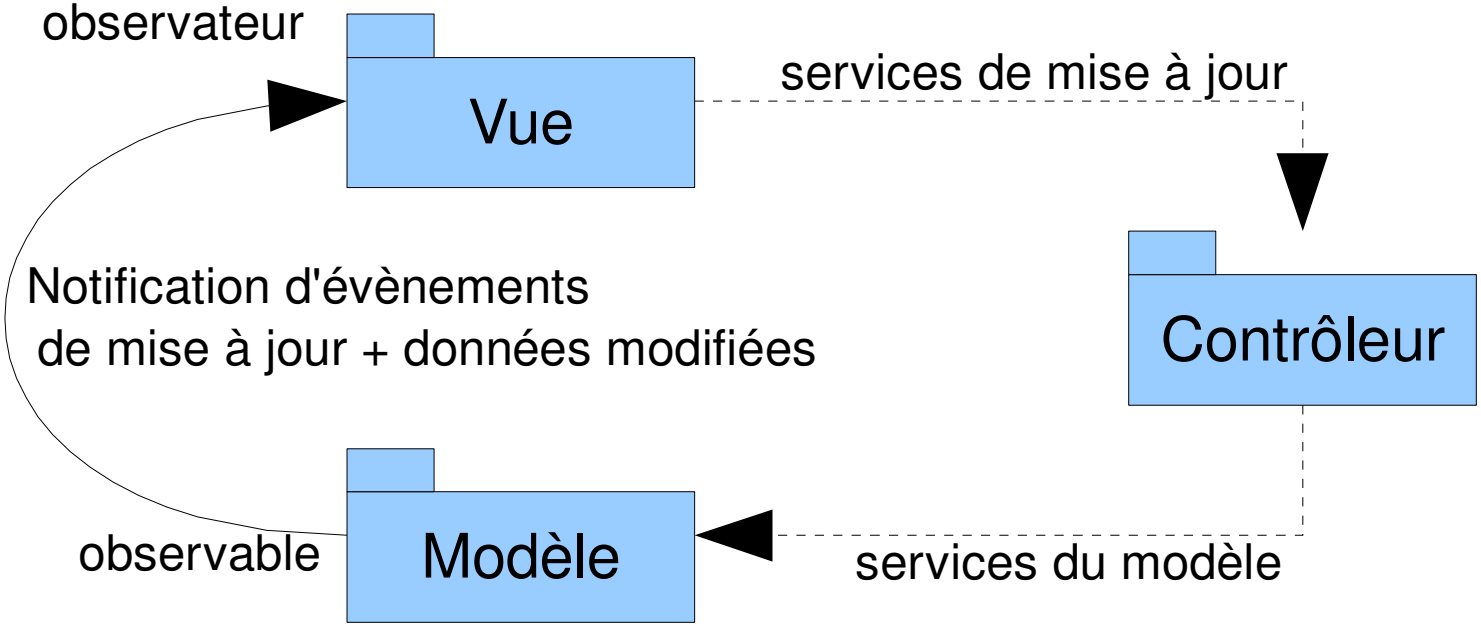
BO.B.RC.18, pauchet@insa-rouen.fr

Plan

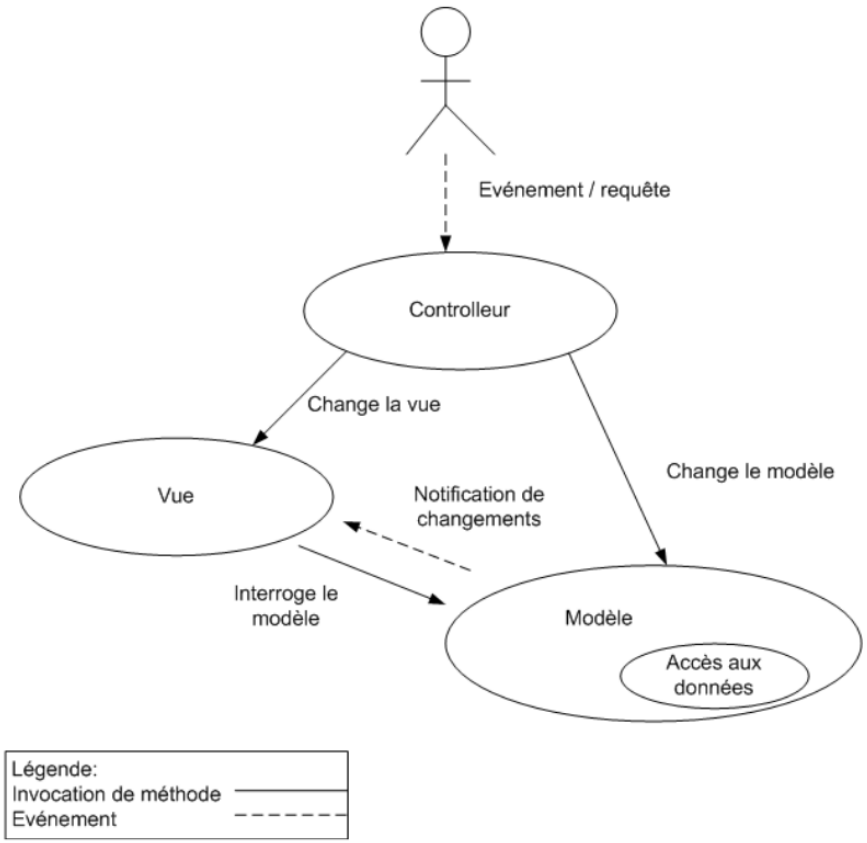
Introduction (1/8)
Le MVC (rappel)

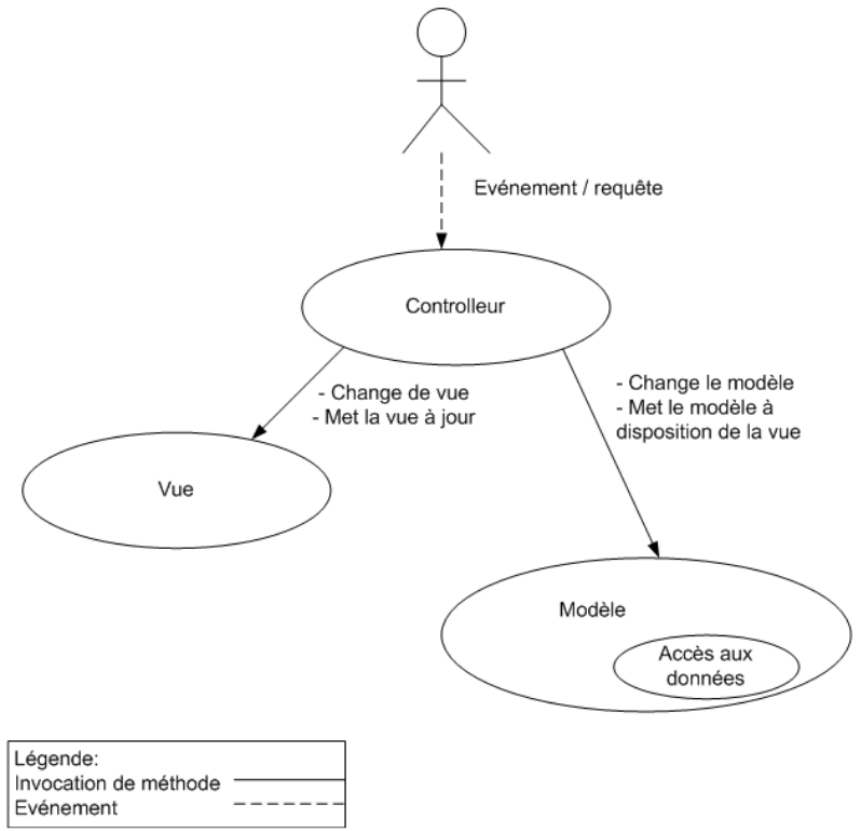
Le Modèle-Vue-Contrôleur (MVC) est une architecture et une méthode de conception pour le développement d'applications logicielles, basé sur :

- Modèle (données) : le comportement de l'application.
- Vue (présentation) : interface avec l'utilisateur.
- Contrôleur (traitement) : gestion des événements de synchronisation entre la vue et le modèle.

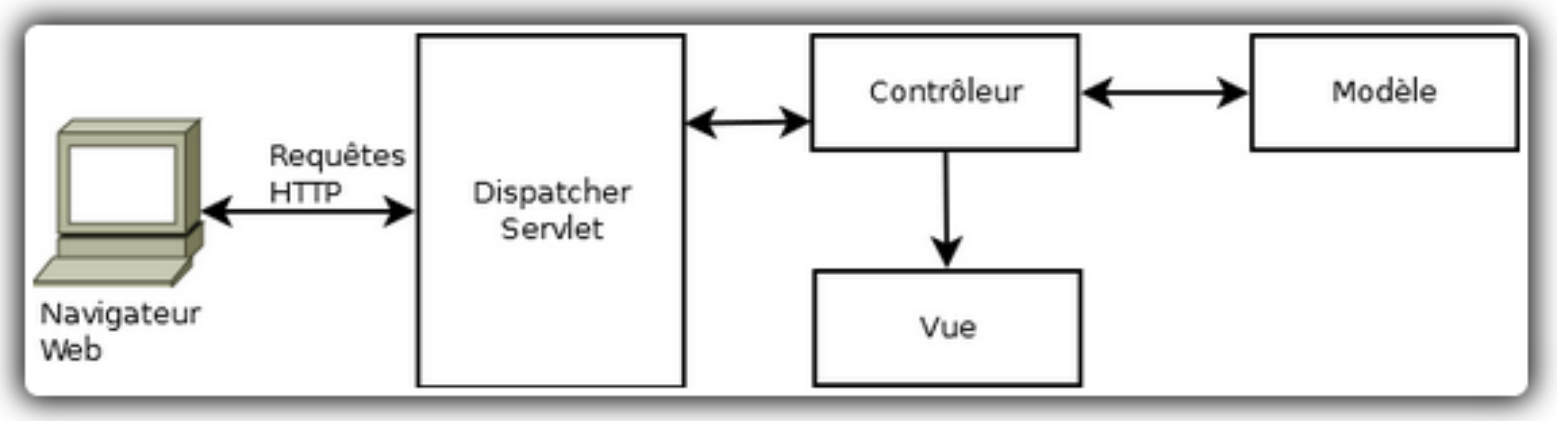


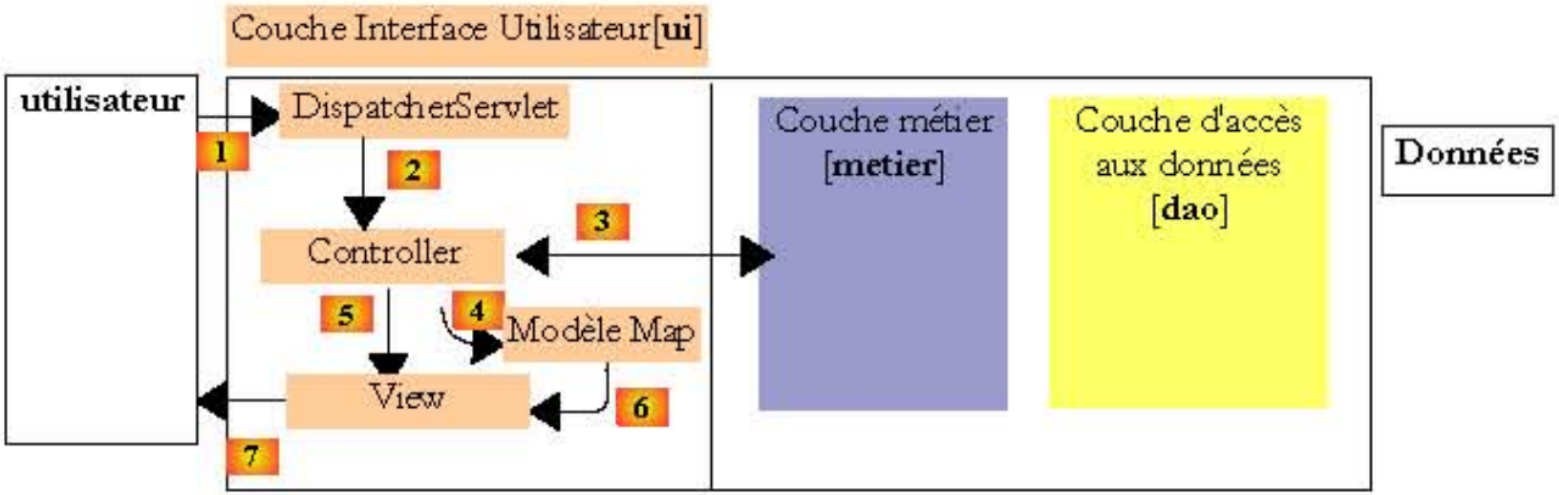
Introduction (2/8)
MVC1 vs MVC2





Introduction (3/8)
MVC in JEE frameworks (Struts et Spring MVC)





Introduction (4/8)
Description

- Framework développé par Apache (comme Tomcat)
- Basé sur JEE (JSP et servlets) :
  - Surcouche JEE,
  - Toute application Struts est une application web JEE
- Ajoute de nouvelles classes
- Ajoute de nouveaux tags pour JSP
- Basé sur l'architecture MVC (Modèle-Vue-Contrôleur)
- Très utilisé (conjointement Spring par exemple)

Introduction (5/8)
Le MVC sans Struts

- Des javabeans (et autres classes) qui gèrent l'aspect métier (modèle)
- Des JSP (éventuellement sans code Java) qui produisent l'affichage (les vues)
- Une ou plusieurs Servlets font le lien entre les JavaBeans et les JSP, *via* les mécanismes de Délégation/Inclusion (contrôleur(s))

Fonctionnement (6/8)
L'idée de Struts

- Un fichier de configuration (`struts.xml`) définissant les actions gérées par le contrôleur
- Une Servlet contrôleur déclarée dans le `web.xml` :

```
<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>
```

Avantages

⇒ Le code Java se limite désormais aux aspects métiers

⇒ Beaucoup plus de souplesse : pour changer le comportement de l'application, il suffit de changer le code XML

⇒ Moins de compilation

Introduction (7/8)
Installation

- Application "vide"
  1. Récupérer une archive `struts2-blank.war` ou `struts2-showcase.war` (comprise dans les exemples d'application) sur `http://struts.apache.org/`. Versions : **2.2.1** et 2.5.18
  2. La copier dans le répertoire de déploiement de JBOSS : elle se décompresse toute seule, comme tout fichier `.war`

- Création d'une nouvelle application
  1. On récupère une archive vide
  2. On la décompresse, renomme et effectue les modifications à l'intérieur
  3. Compilation : `ant clean/compile/init/build/archive` ou `mvn jetty:run`

Introduction (8/8)

Contenu (partiel) de struts-blank

```
struts-blank.war
   |_ index.html
   |_ error.jsp
   |_ example
   |    |_ Welcome.jsp, ...
   |_ WEB_INF
   |    |_ web.xml
   |    |_ lib
   |    |    |_ struts2-core-2.2.1.jar, ...
   |    |_ classes
   |         |_ struts.xml, ...
   |_ META-INF
        |_ MANIFEST.MF, ...
```

Fonctionnement (1/6)

Fonctionnement d'une application Struts

- L'utilisateur appelle une URL de la forme `traitement.action` (formulaire, lien, *etc.*), traitée selon le contenu de `struts.xml`
- On appelle la méthode `execute` d'une instance dérivée de la classe `ActionSupport` :

```
public String execute() throws Exception { ... }
```

- Une JSP est appelée en fonction du résultat de la méthode `execute` et du `struts.xml`

Fonctionnement (2/6)
Exemple : HelloWorld

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">

    <!-- If no class attribute is specified the framework will assume success and
     render the result index.jsp -->
    <!-- If no name value for the result node is specified the success value is the
     default -->
    <action name="index">
      <result>/index.jsp</result>
    </action>

    <!-- If the URL is hello.action the call the execute method of class
     HelloWorldAction.
    If the result returned by the execute method is success render the HelloWorld.jsp
     -->
    <action name="hello" class="helloworld.action.HelloWorldAction" method="execute">
      <result name="success">/HelloWorld.jsp</result>
    </action>
  </package>
</struts>
```

Fonctionnement (3/6)
Exemple : HelloWorld

index.jsp

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

Fonctionnement (4/6)
Exemple : HelloWorld

HelloWordAction.java

```
package helloworld.action;

import helloworld.model.StringMessage;
import com.opensymphony.xwork2.ActionSupport;

public class HelloWorldAction extends ActionSupport {

  private StringMessage message;

  public String execute() throws Exception {
    this.message = new StringMessage("Hello World !") ;
    return SUCCESS;
  }

  public StringMessage getMonMessage() {
    return this.message;
  }

  public void setMonMessage(StringMessage message) {
    this.message = message;
  }
}
```

Fonctionnement (5/6)
Exemple : HelloWorld

StringMessage.java

```
package helloworld.model;

public class StringMessage {

  private String message;

  public StringMessage() {
    this.setContenuMessage("");
  }

  public StringMessage(String message) {
    this.setContenuMessage(message);
  }

  public String getContenuMessage() {
    return this.message;
  }

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

Fonctionnement (6/6)
Exemple : HelloWorld

HelloWorld.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO
    -8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags" %>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
    <title>Hello World!</title>
  </head>
  <body>
    <h2><s:property value="monMessage.contenuMessage" /></h2>
  </body>
</html>
```

Données en provenance d'un formulaire / d'un lien (1/5)

Fonctionnement

Utilisation de javabeans

- L'utilisateur remplit (par exemple) un formulaire issu d'une JSP ou d'un document HTML statique
- On appelle lors de l'envoi du formulaire une URL de la forme `traitement.action`, qui est traitée en fonction du contenu de `struts.xml` ; la méthode `execute` d'une instance dérivée de la classe `ActionSupport` est alors utilisée
- **La classe appelée doit se présenter sous la forme d'un javabean dont les propriétés (getter et setter) vont correspondre aux champs du formulaire**
- Une JSP est appelée en fonction du résultat de la méthode et du `struts.xml`

Données en provenance d'un formulaire / d'un lien (2/5)

Exemple : HelloWorldPost

index.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML>
<%@taglib uri="/struts-tags" prefix="s" %>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Hello World Post</title>
    </head>
    <body>
        <s:form action="hello" >
            <s:textfield name="name" label="User Name" />
            <s:submit />
        </s:form>

        <s:url action="hello" var="helloLink">
          <s:param name="name">Bob Leponge</s:param>
        </s:url>
        <p><a href="${helloLink}">Hello Bob Leponge</a></p>
    </body>
</html>
```

Données en provenance d'un formulaire / d'un lien (3/5)

Exemple : HelloWorldPost

struts.xml

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

<struts>
  <package name="default" extends="struts-default">
    <action name="index">
      <result>/index.jsp</result>
    </action>

    <action name="hello" class="helloworldpost.action.HelloWorldPostAction" method="
     execute">
       <result name="success">/success.jsp</result>
    </action>
  </package>
</struts>
```

Données en provenance d'un formulaire / d'un lien (4/5)

Exemple : HelloWorldPost

HelloWordPostAction.java

```
package helloworldpost.action;

import com.opensymphony.xwork2.ActionSupport;

public class HelloWorldPostAction extends ActionSupport {
  private String userName;
  private String message;

  public String execute() throws Exception {
    this.message = "Hello " + this.userName + " !";
    return SUCCESS;
  }

  public String getMessage() {
    return this.message;
  }

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

  public String getName() {
    return this.userName;
  }

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

Données en provenance d'un formulaire / d'un lien (5/5)

Exemple : HelloWorldPost

success.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML>
<%@taglib uri="/struts-tags" prefix="s" %>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Hello World</title>
    </head>
    <body>
        <h1><s:property value="message" /></h1>
        <h1>Propriété "name" récupérée : <s:property value="name" /></h1>
    </body>
</html>
```

Couverture d'un formulaire par un Javabean (1/10)
Principe

Principe

- Les propriétés d'un Javabean sont utilisées pour encapsuler les valeurs d'un formulaire
- Des tags Struts 2 sont utilisés pour lier champs et propriétés
- Les tags Struts 2 sont interprétés pour générer du code HTML
  $\Rightarrow$ Dans `struts.xml`, une méthode doit être définie pour forcer le filtre et parser les tags
- Certains tags (listes, radio buttons, *etc.*) nécessitent une initialisation

Couverture d'un formulaire par un Javabean (2/10)

Exemple : Javabean

- Déclaration de 2 actions pour le contrôleur
  - 1 action (edit), 1 méthode (input → input) et une classe (Register dans l'exemple) pour déclarer l'édition de formulaire
  - 1 action (register), 1 méthode (execute → success) et une classe (Register dans l'exemple), pour son traitement

- Lier un Javabean avec les éléments d'un formulaire
  - textfield : name → setter correspondant
  - radio/select : key → setter correspondant + list → setter correspondant pour la liste d'éléments
  - checkbox : key → setter correspondant

Couverture d'un formulaire par un Javabean (3/10)

Exemple : Javabean

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">
    <!-- If no class attribute is specified the framework will assume success and
     render the result index.jsp -->
    <!-- If no name value for the result node is specified the success value is the
     default -->
    <action name="index">
      <result>/index.jsp</result>
    </action>
    <action name="edit" class="register.action.RegisterComponent" method="input">
      <result name="input">/edit.jsp</result>
    </action>
    <action name="register" class="register.action.RegisterComponent" method="execute">
      <result name="success">/thankyou.jsp</result>
    </action>
  </package>
</struts>
```

Couverture d'un formulaire par un Javabean (4/10)

Exemple : Javabean

index.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO
    -8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags" %>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
    <title>Struts 2 Form Tags - Welcome</title>
  </head>
  <body>
    <h1>Welcome To Struts 2!</h1>
    <p><a href='<s:url action="edit" />' >Edit your information</a></p>
  </body>
</html>
```

Couverture d'un formulaire par un Javabean (5/10)

Exemple : Javabean

edit.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO
    -8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags" %>
<!DOCTYPE html>
<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>Fill out and submit the following form:</p>
    <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:radio key="personBean.gender" label ="Gender" list="genders" />
      <s:select key="personBean.sport" label ="Sport" list="sports" />
      <s:checkbox key="personBean.over21" label="Over 21" />
      <s:submit value="Submit" />
    </s:form>
  </body>
</html>
```

Couverture d'un formulaire par un Javabean (6/10)

Exemple : Javabean

RegisterComponent.java

```
package register.action;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import register.model.Person;
import com.opensymphony.xwork2.ActionSupport;

public class RegisterComponent extends ActionSupport {
  private Person personBean;
  private String [] sports = {"football", "volleyball", "basketball"};
  private String [] genders = {"male", "female"};

  public String execute() throws Exception {
    return SUCCESS;
  }

  public String input() throws Exception {
    return INPUT;
  }

  public Person getPersonBean() {
    return personBean;
  }


...
```

Couverture d'un formulaire par un Javabean (7/10)

Exemple : Javabean

RegisterComponent.java

```
...
  public void setPersonBean(Person person) {
    personBean = person;
  }

  public List<String> getSports() {
    return Arrays.asList(sports);
  }

  public List<String> getGenders() {
    return Arrays.asList(genders);
  }
}
```

Couverture d'un formulaire par un Javabean (8/10)

Exemple : Javabean

Person.java

```
package register.model;

public class Person
{
  private String firstName;
  private String lastName;
  private String email;
  private String gender;
  private boolean over21;
  private String sport;

  public String getFirstName() { return firstName; }
  public void setFirstName(String firstName) { this.firstName = firstName; }

  public String getLastName() { return lastName; }
  public void setLastName(String lastName) { this.lastName = lastName; }

  public String getEmail() { return email; }
  public void setEmail(String email) { this.email = email; }


...
```

Couverture d'un formulaire par un Javabean (9/10)

Exemple : Javabean

Person.java

```
...
  public void setGender(String gender) { this.gender = gender; }
  public String getGender() { return gender; }

  public void setOver21(boolean over21) { this.over21 = over21; }
  public boolean isOver21() { return over21; }

  public String getSport() { return sport; }
  public void setSport(String sport) { this.sport = sport; }

  public String toString() {
    String desc = getFirstName() + " " + getLastName() +  ", " + getEmail() + ", " +
     getGender() + ", joueur de " + getSport();
    if(over21)
      desc += ", majeur";
    else
      desc += ", mineur";
    return desc;
  }
}
```

Couverture d'un formulaire par un Javabean (10/10)

Exemple : Javabean

thankyou.jsp

```
<%@ taglib prefix="s" uri="/struts-tags" %>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO
    -8859-1"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <title>Registration Successful</title>
  </head>
  <body>
    <h3>Thank you for registering !</h3>
    <p>Your registration information: <s:property value="personBean" /> </p>
    <p><a href="<s:url action='edit' />" >Return to home page</a>.</p>
  </body>
</html>
```

Validation d'un formulaire (1/6)
Principe

Validation de formulaire en utilisant Struts 2

- Utiliser les résultats de la méthode de traitement d'un formulaire pour choisir la vue :
  - Le contrôle est défini dans `struts.xml`
  - Les tests sont effectués dans le contrôleur
- Faciliter les messages d'erreurs en utilisant les tags Struts 2
- **La validation se fait en AJAX (voir cours correspondant) : utilisation de Javascript pour interroger de manière asynchrone le serveur sur la validité des champs**

Validation d'un formulaire (2/6)

Exemple : Validation

index.jsp

```
<%@ taglib prefix="s" uri="/struts-tags" %>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO
    -8859-1"%>
<!DOCTYPE html PUBLIC>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <title>Register</title>
    <s:head /><!-- Nécessaire pour afficher les erreurs !-->
  </head>
  <body>
    <h3>Please register</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>
```

Validation d'un formulaire (3/6)

Exemple : Validation

Person.java

```
package register.model;

public class Person
{
  private String firstName;
  private String lastName;
  private String email;
  private int age;

  public String getFirstName() { return firstName; }
  public void setFirstName(String firstName) { this.firstName = firstName; }

  public String getLastName() { return lastName; }
  public void setLastName(String lastName) { this.lastName = lastName; }

  public String getEmail() { return email; }
  public void setEmail(String email) { this.email = email; }

  public int getAge() { return age; }
  public void setAge(int age) { this.age = age; }

  public String toString() {
    return getFirstName() + " " + getLastName() +  ", " + getEmail() + ", " + getAge()
     + " ans";
  }
}
```

Validation d'un formulaire (4/6)

Exemple : Validation

RegisterComponent.java

```
package register.action;

import register.model.Person;
import com.opensymphony.xwork2.ActionSupport;

public class RegisterComponent extends ActionSupport {
  private Person personBean;

  public String execute() throws Exception { return SUCCESS; }

  public String input() throws Exception { return INPUT; }

  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" );
    }
    }

  public Person getPersonBean() { return this.personBean; }
  public void setPersonBean(Person person) { this.personBean = person; }
}
```

Validation d'un formulaire (5/6)

Exemple : Validation

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="edit" class="register.action.RegisterComponent" method="input">
      <result name="input">/index.jsp</result>
    </action>

    <action name="register" class="register.action.RegisterComponent" method="execute">
      <result name="success">/thankyou.jsp</result>
      <result name="input">/index.jsp</result>
    </action>
  </package>
</struts>
```

La présence d'une méthode `validate` implique la recherche d'un résultat `input` retourné par le contrôleur Struts

Validation d'un formulaire (6/6)

Exemple : Validation

thankyou.jsp

```
<%@ taglib prefix="s" uri="/struts-tags" %>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO
    -8859-1"%>
<!DOCTYPE html>
  <html xmlns="http://www.w3.org/1999/xhtml">
    <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
  <title>Registration Successful</title>
  </head>
  <body>
    <p>Registration information: <s:property value="personBean" /> </p>
    <p><a href="<s:url action='index' />" >Return to home page</a>.</p>
  </body>
</html>
```

Conclusion (1/3)
Struts ? Et le reste ?

Struts n'est pas le seul outil de haut niveau disponible pour les applications web JEE :

- Java Server Faces (JSF),
- Tapestry,
- Spring (Spring MVC),
- Hibernate (qui peut être combiné avec Struts ou Spring),
- ...

Ces outils ne sont pas mutuellement exclusifs :

- Struts est plutôt spécialisé "contrôleur"
- JSF propose des composants graphiques : "vue"
- Hibernate, les EJB et autres s'intéressent à la sérialisation des objets : "modèle"

Conclusion (2/3)
Faut-il utiliser Struts ?

- Oui si l'application est conséquente ;
- Non s'il s'agit d'une application simple (exemple limite, le blog) : même le site de Struts déconseille d'utiliser le Framework dans ces cas-là
- On peut utiliser des JSP seules, du PHP ou un framework dit agile (Ruby on Rails, Django, ...) dans les cas plus simples

Conclusion (3/3)
Sources

http://struts.apache.org/

http://litis.univ-lehavre.fr/~duvallet/enseignements/Cours/JEE/COURS-SPRING.pdf

http://www-lih.univ-lehavre.fr/~coletta/dess0607/struts/une_appli.html
http://www-lih.univ-lehavre.fr/~coletta/dess0607/struts/erreurs.html

http://www.labo-sun.com/resource-fr-essentiels-859-5-java-j2ee-struts-un-framework-mvc-pour-vos-applications-j2ee.htm\#h1n3

http://www.jmdoudoux.fr/java/dej/chap052.htm

https://rpouiller.developpez.com/tutoriels/spring/application-web-spring-hibernate/

http://www.lifl.fr/~dumoulin/enseign/2013-2014/ipint/cours/8.spring/Spring-2013.pdf