Java EE

Cours 8

Cours de 2e année ingénieur
Spécialisation « Génie Informatique »

JEE Approfondi

- A partir de ce cours, nous verrons comment simplifier l'utilisation de JEE en se servant de frameworks développés dans ce but.

- A ce jour, il existe un très grand nombre de frameworks pour JEE. Certains sont sous licence libre et donc gratuits tandis que d'autres sont payants.

- De plus **certains frameworks** ne sont pas limités à JEE mais sont plus génériques à Java et peuvent donc s'appliquer **à n'importe quelle application JSE** (c'est surtout le cas des frameworks de persistance).

- Ces frameworks peuvent être regroupés en 2 familles :
  - Les frameworks dit **MVC** que nous illustrerons en approfondissant le framework : **Struts**
  - Les frameworks dit **de Persistance** illustrés par **Hibernate**

Frameworks MVC

- Struts 1 (http://struts.apache.org/1.x/)
- Struts 2 (http://struts.apache.org/2.x/)
- Stripes(http://www.stripesframework.org)
- Spring (http://www.springsource.org/)
- JavaServer Faces (http://jcp.org/en/jsr/detail?id=127)
- Tapestry (http://tapestry.apache.org/)
- Seam (http://www.seamframework.org/)

- LifeRay (http://www.liferay.com/)
- JetSpeed (http://portals.apache.org/jetspeed-2/)
- SiteMesh(http://www.opensymphony.com/sitemesh/)

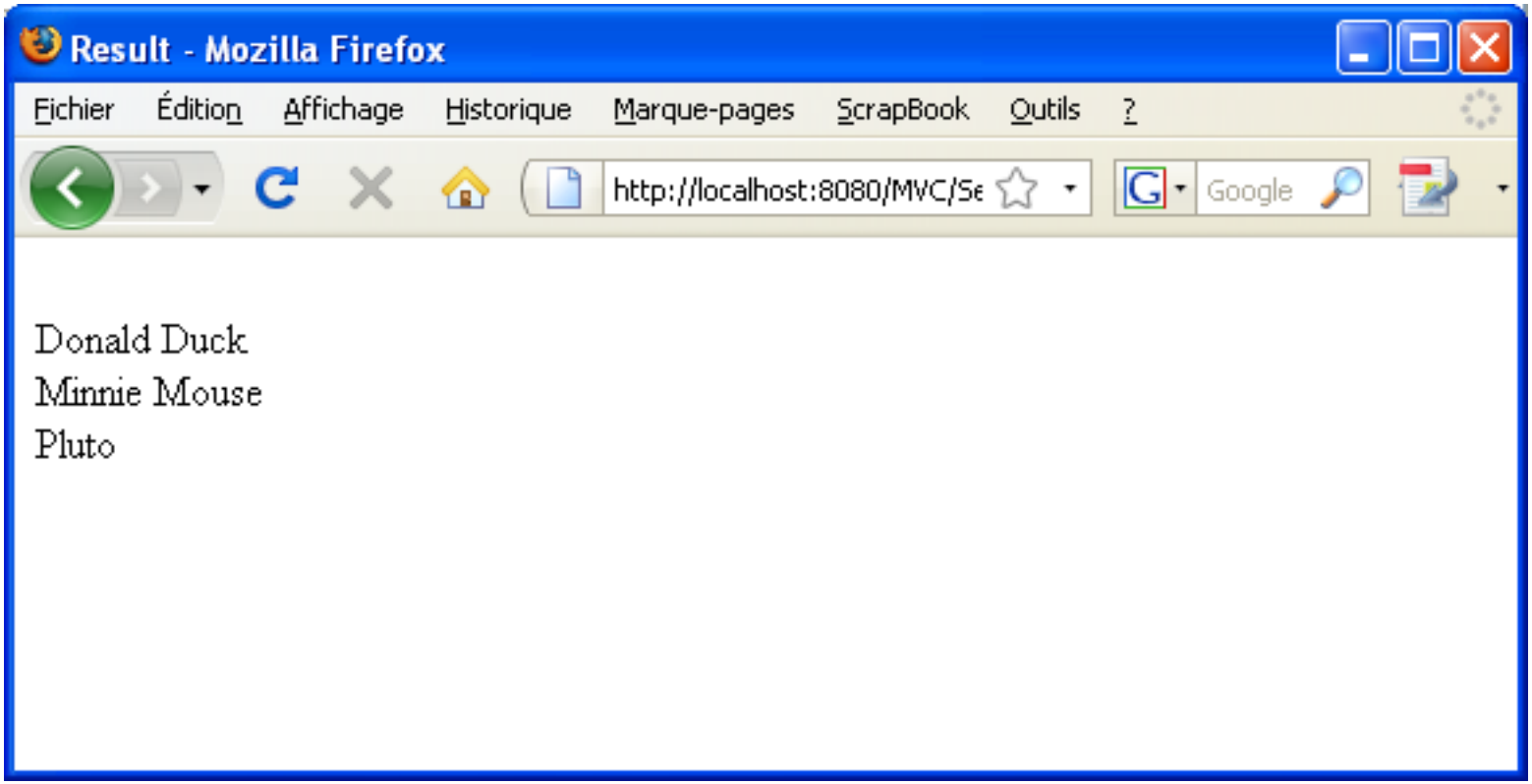
Frameworks persistance/XML

- Hibernate (https://www.hibernate.org/)
- JAXB(https://jaxb.dev.java.net/) **(XML)**
- iBATIS (http://ibatis.apache.org/)
- TopLink (http://www.oracle.com/technology/products/ias/toplink/index.html)
- EclipseLink (http://www.eclipse.org/eclipselink/)
- Cayenne (http://cayenne.apache.org/)
- XStream(http://xstream.codehaus.org/) **(XML)**

Rappel MVC

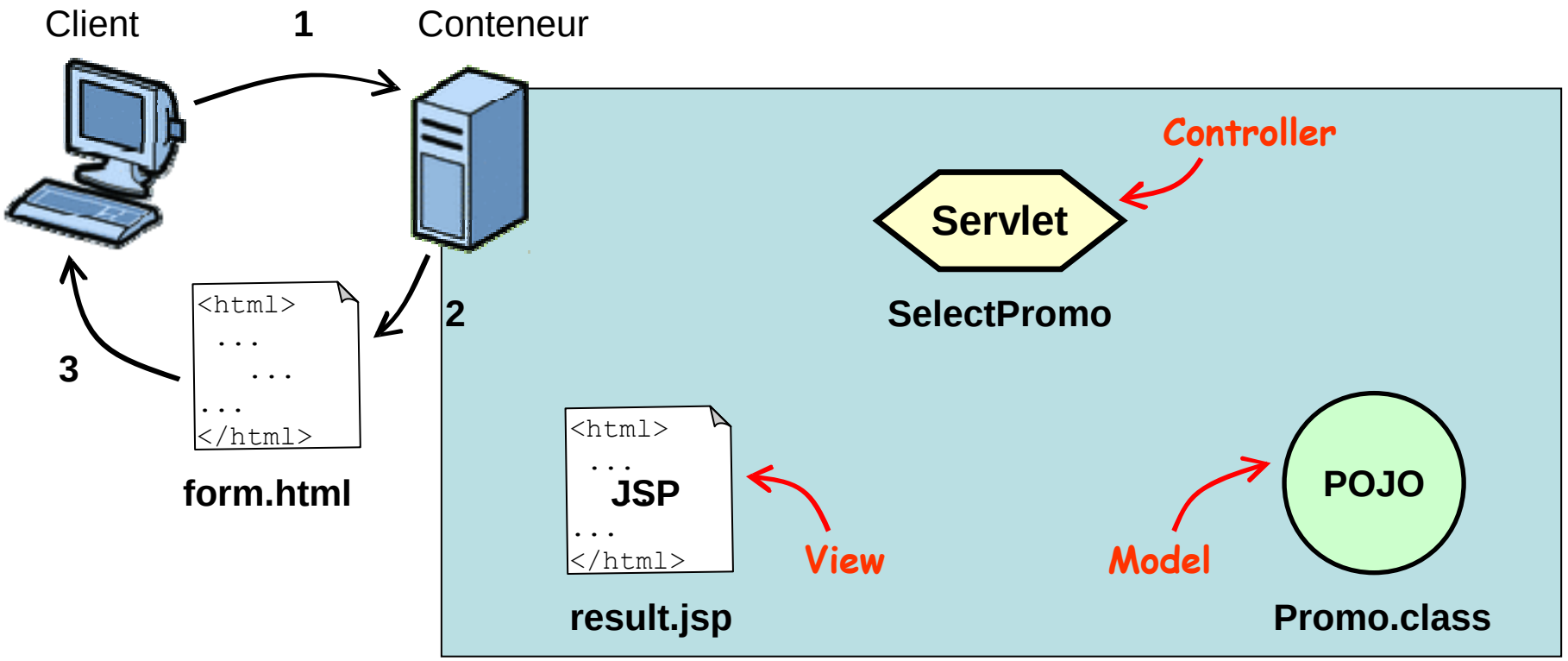
Ex : AREL V6 – liste des promos

Ex : AREL V6 – liste des promos



MVC : étape 1

Le client récupère un formulaire (form.html)
pour passer une requête avec paramètres (1, 2, puis 3)

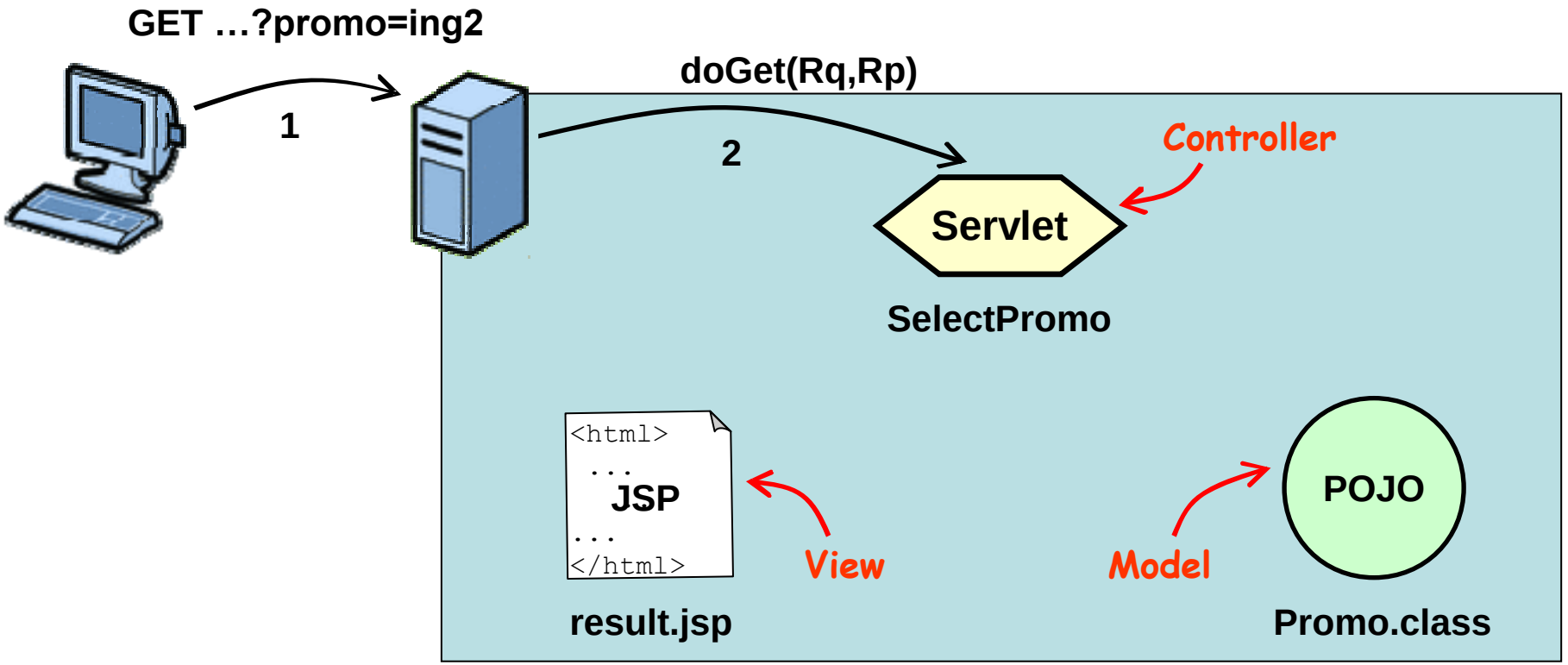


Formulaire : form.html

```
<!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>AREL V6.0</title>
</head>
<body>
  <h1align="center">AREL: L'école virtuelle de l'EISTI</ h1>

<form method="GET" action="http://localhost:8080/MVC/SelectPromo">
Sélectionner la promo à afficher:

 <select name="promo" size="1">
  <option>ing1</option>
  <option>ing2</option>
 </select><input type="SUBMIT" />

</form>
</body>
</html>
```

MVC : étape 2

1. Le client envoie son formulaire (GET/POST avec paramètres)
2. Le conteneur transmet au servlet correspondant (le *controller*)



Controller : SelectPromo.java

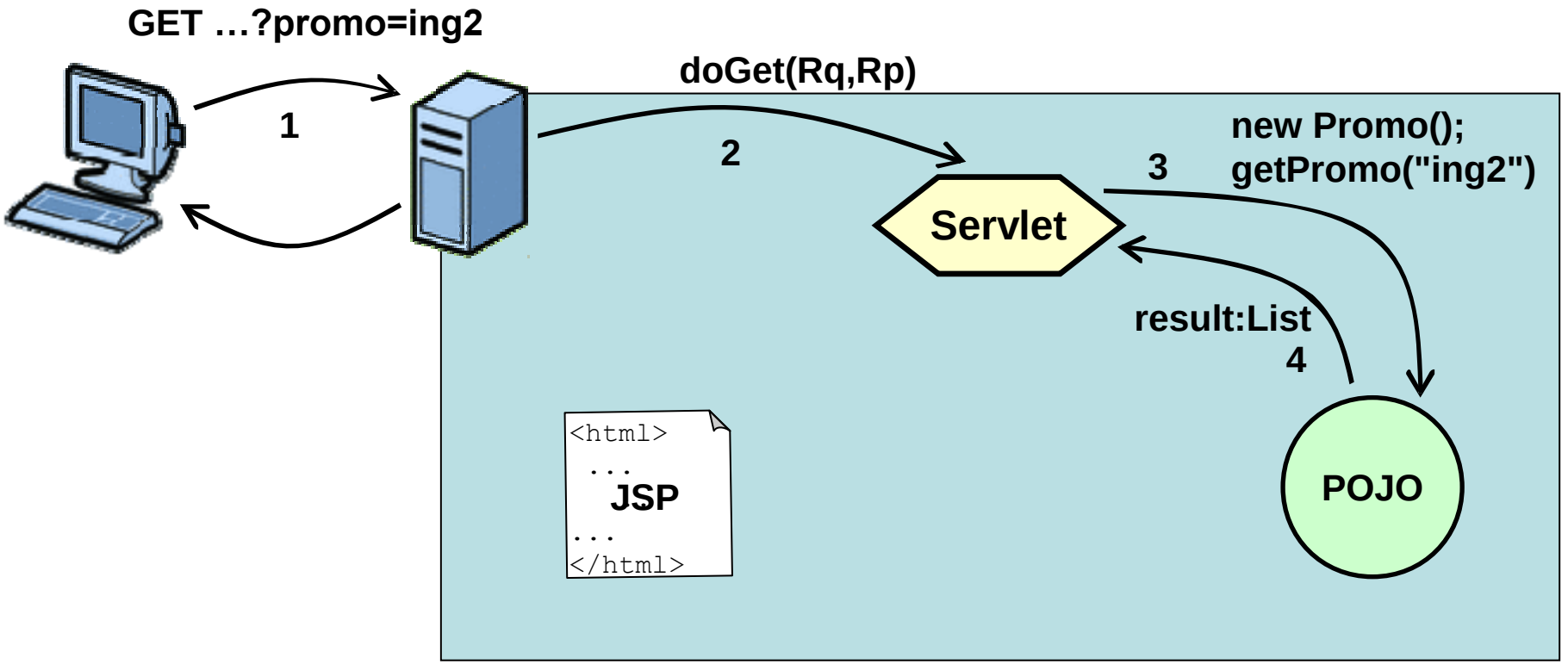
```
package arel;

import ...;

public class SelectPromo extends javax.servlet.http.HttpServlet
                         implements javax.servlet.Servlet{
   //...
   protected void doGet(HttpServletRequest request,
                        HttpServletResponse response)
                 throws ServletException, IOException{

      String promoName = request.getParameter( "promo" );

      //...
   }
}
```

Configuration : web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
 http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
         id="WebApp_ID" version="2.5">
  <display-name>MVC</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
</welcome-file-list>
  <servlet>
    <description></description>
    <display-name>SelectPromo</display-name>
    <servlet-name>SelectPromo</servlet-name>
    <servlet-class>arel.SelectPromo</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>SelectPromo</servlet-name>
    <url-pattern>/SelectPromo</url-pattern>
  </servlet-mapping>
</web-app>
```

MVC : étape3

3. Le servlet *controller* interroge le *model* sur « ing2 »
4. Le *model* retourne au *controller* le résultat correspondant

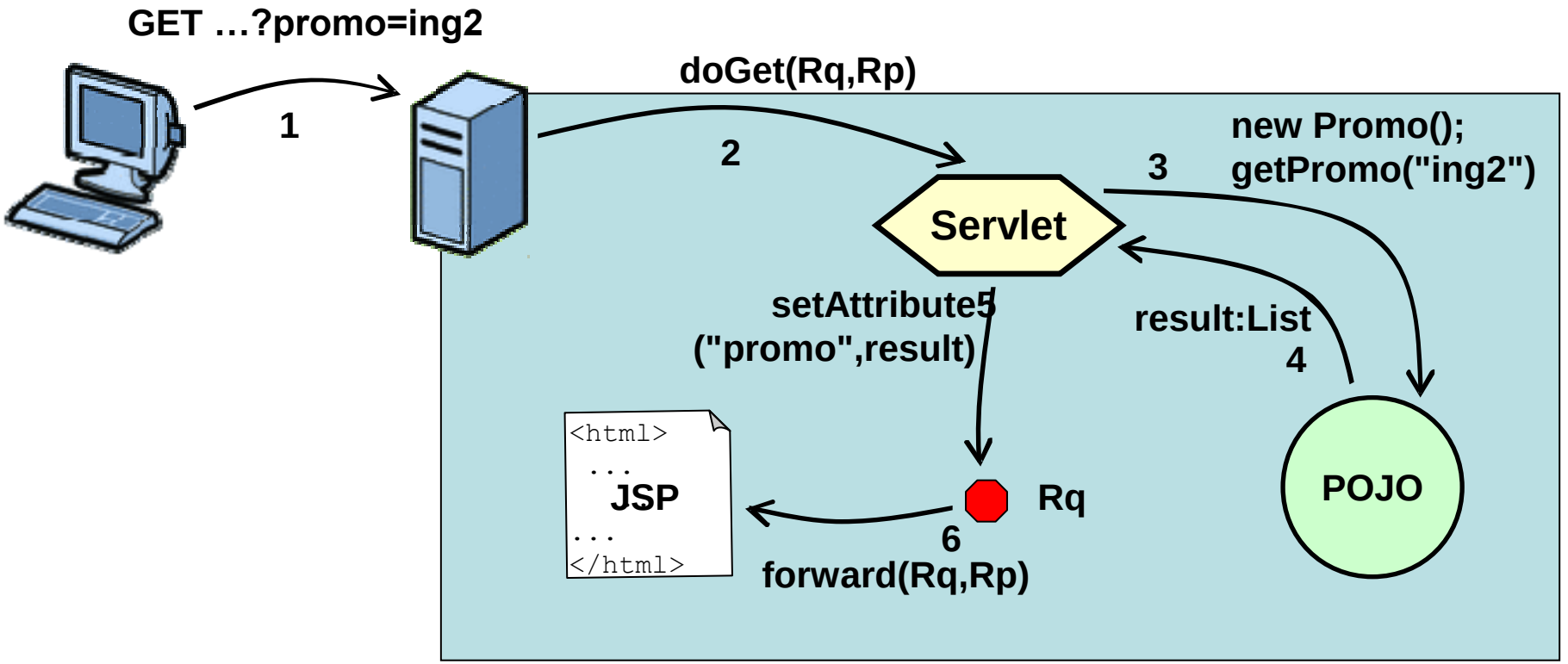


Model : Promo.java

```
package arel;
import ...;

public class Promo{

   public List<String> getPromo(String promo){
     List<String> promoList = new ArrayList<String>();
     if(promo.equals("ing1")){
       promoList.add("Donald Duck");
       promoList.add("Minnie Mouse");
       promoList.add("Pluto");  //...
     } else if (promo.equals("ing2")){
       promoList.add("Mickey Mouse");
       promoList.add("Daisy Duck");
       promoList.add("Goofy");  //...
     } else{ return null;}

     return promoList;
   }
}
```

MVC : étape 4

5. Le *controller* utilise les données du *model* pour sa réponse
6. Le *controller* transmet sa réponse à la *view* (JSP)

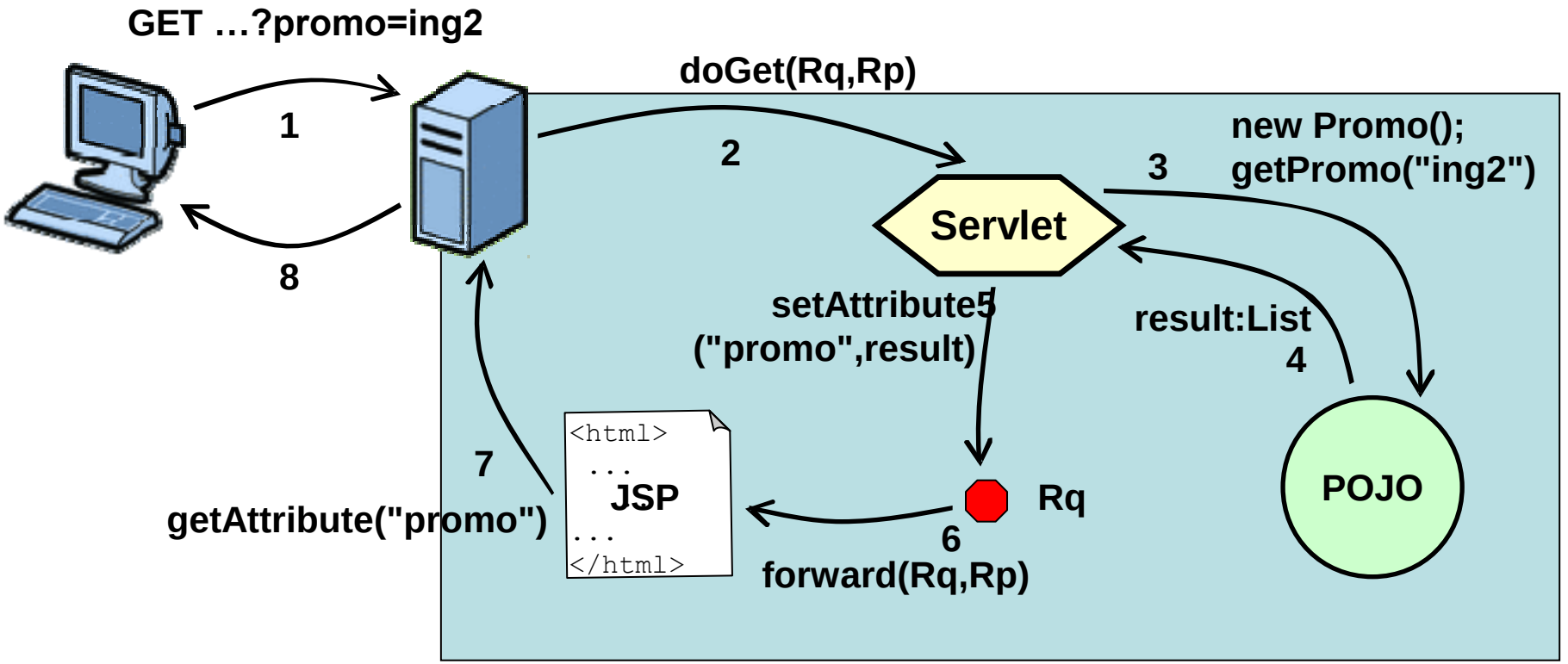


Controller : SelectPromo.java

```
package arel;

import ...;

public class SelectPromo extends javax.servlet.http.HttpServlet
                         implements javax.servlet.Servlet{
  //...
  protected void doGet(HttpServletRequest request,
                       HttpServletResponse response)
                  throws ServletException, IOException{
    String promoName = request.getParameter("promo");
    Promo promo = new Promo();
    List<String> result = promo.getPromo(promoName);
    request.setAttribute("promo", result);
    RequestDispatcher view=
      request.getRequestDispatcher("result.jsp");
    view.forward(request, response);
  }
}
```

MVC : étape5

7. La JSP (view) traite la réponse transmise par le *controller*
8. La page HTML résultante est reçue par le client



View : result.jsp

```
<%@ page import="java.util.*" %>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!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>Result</title>
</head>
<body>
<%
  List<String> promoList=(List<String>)request.getAttribute("promo");
  Iteratorit= promoList.iterator();
  while(it.hasNext()){
    out.print("<br />"+it.next());
  }
%>
</body>
</html>
```

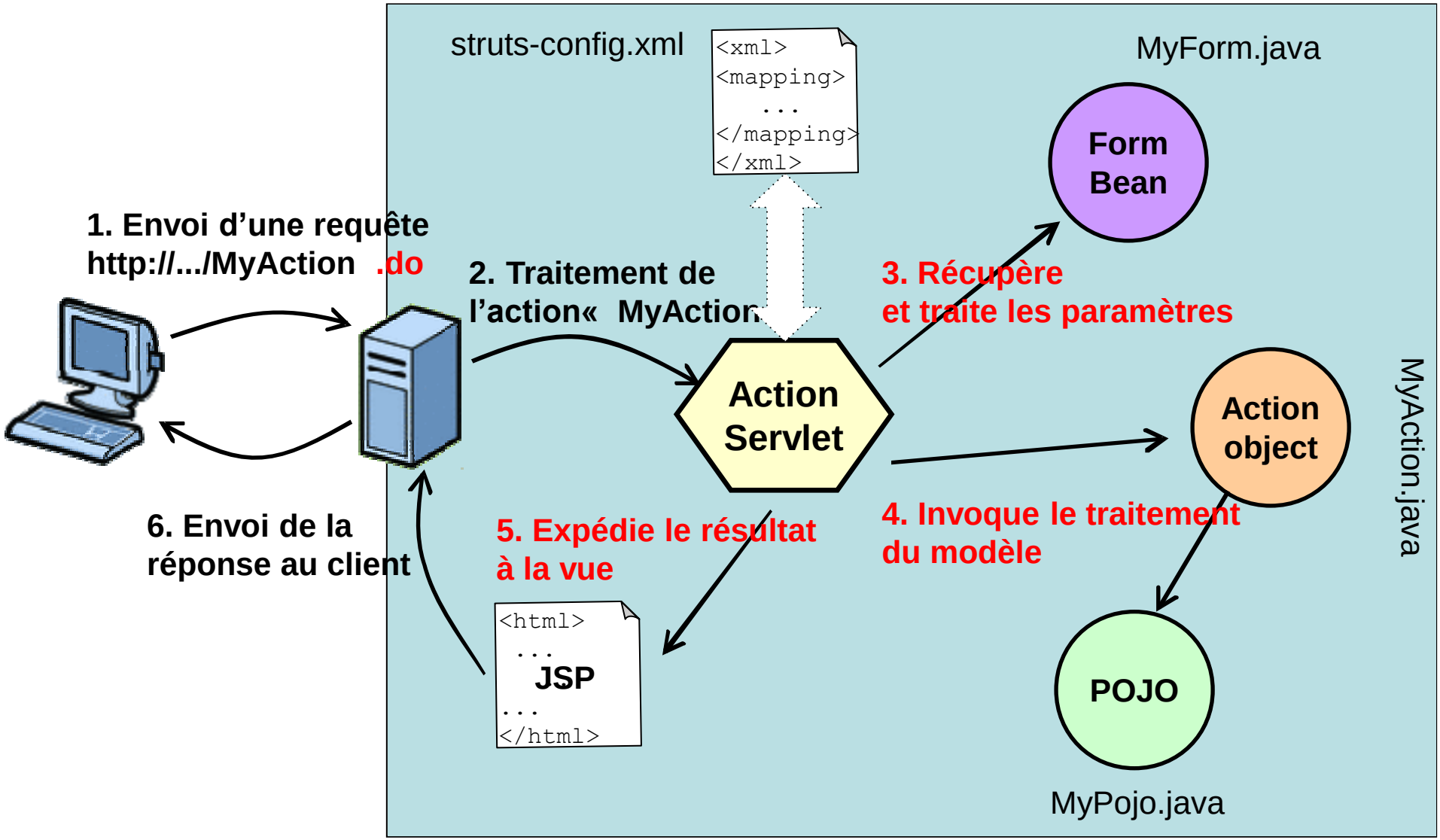
Struts

http://struts.apache.org/1.3.10/index.html

Une composition de patterns

- Mise en œuvre d'un design pattern pour Java EE : «Front Controller»
  - Un seul servlet pour traiter toutes les requêtes
  - Cf. http://java.sun.com/blueprints/patterns/catalog.html
- Le Front Controller, appelé ActionServlet, permet de mettre en œuvre le pattern MVC de façon générique

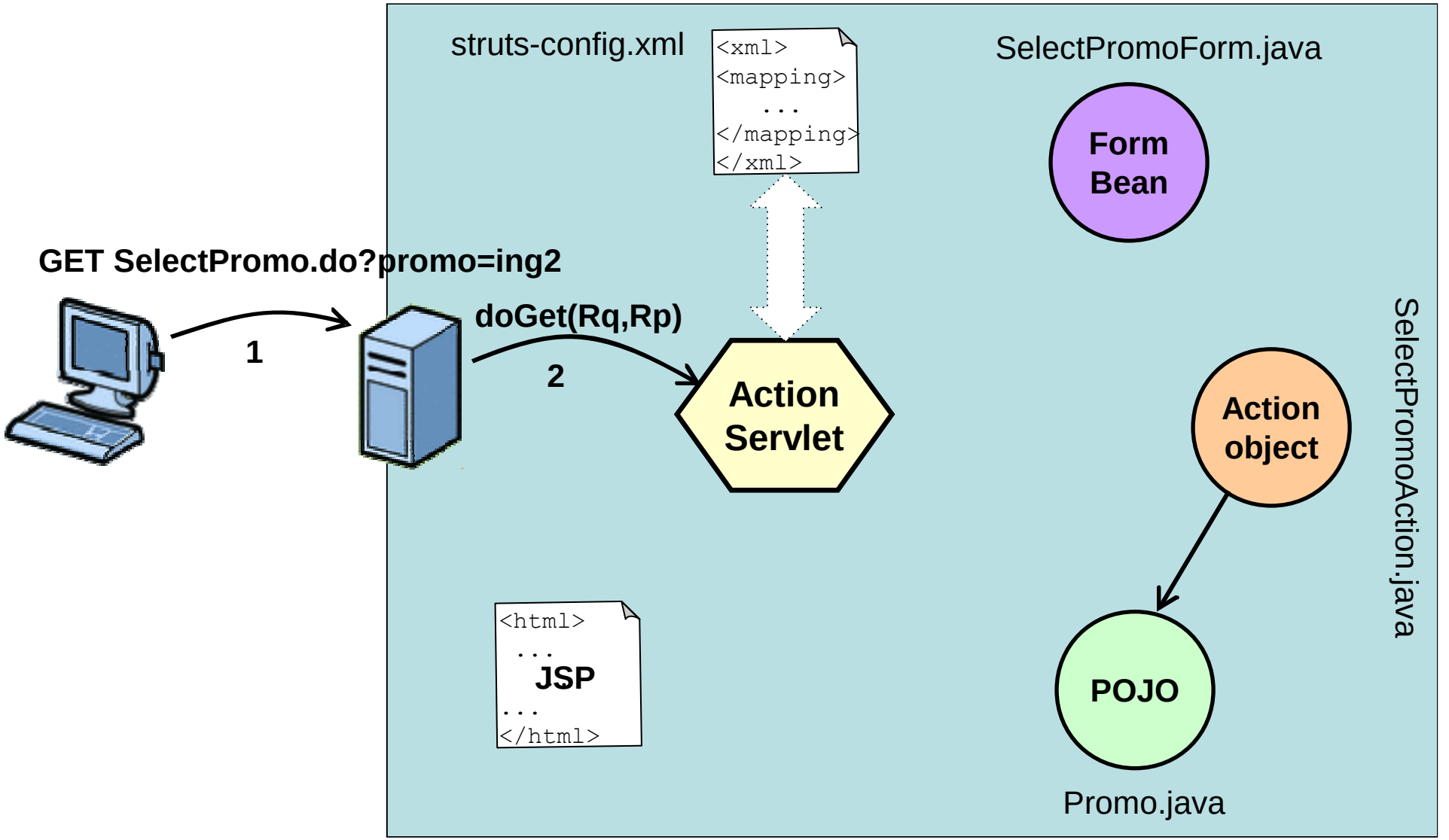
Architecture Struts



Configuration : web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC…>

<web-app>
  <display-name>Struts Blank Application</display-name>

  <!-- Standard Action Servlet Configuration -->
  <servlet>
    <servlet-name>action</servlet-name>
    <servlet-class>org.apache.struts.action.ActionServlet
    </servlet-class>
    <init-param>
      <param-name>config</param-name>
      <param-value>/WEB-INF/struts-config.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
 </servlet>

  <!-- Standard Action Servlet Mapping -->
  <servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern> *.do </url-pattern>
  </servlet-mapping>
</web-app>
```

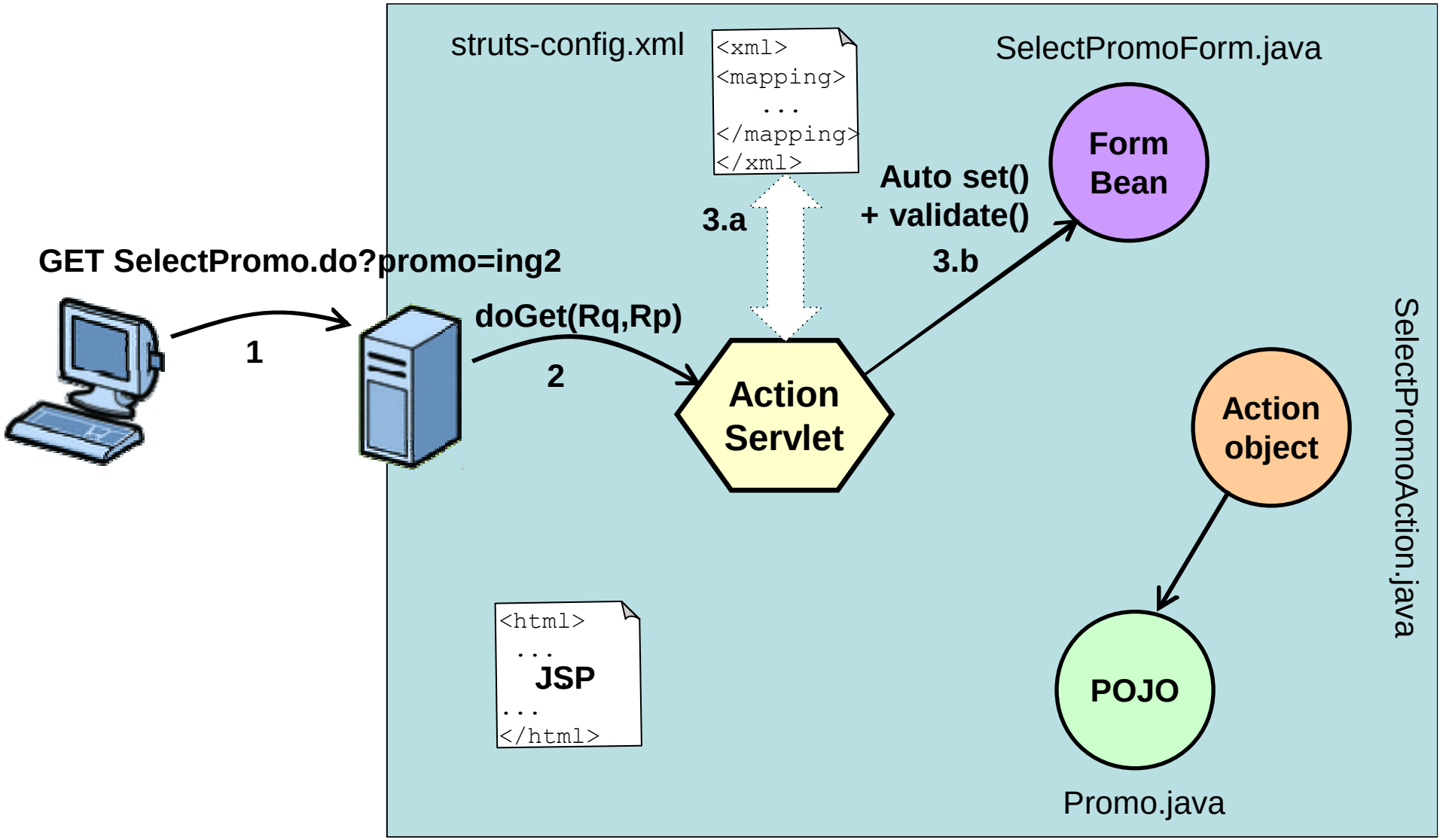
Étapes 1 & 2


struts-config.xml
<xml>
<mapping>
...
</mapping>
</xml>
SelectPromoForm.java
Form Bean
GET SelectPromo.do?promo=ing2
1
doGet(Rq,Rp)
2
Action Servlet
Action object
SelectPromoAction.java
<html>
...
JSP
...
</html>
POJO
Promo.java

Configuration : struts-config.xml

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE struts-config PUBLIC…  struts-config_1_3.dtd">

<struts-config>
<!-- Form Bean Definitions -->
 <form-beans>
   <form-bean
    name="selectPromoForm"
    type="arel.SelectPromoForm" />
 </form-beans>

<!-- Action Mapping Definitions -->
 <action-mappings>
   <action path="/SelectPromo"
    type="arel.SelectPromoAction" name="selectPromoForm"
    scope="request" validate="true" input="/form.jsp">
     < forward name="show_results" path="/pages/result.jsp" />
   </action>
 </action-mappings>

<!-- Other Definitions… -->
</struts-config>
```

Étape 3


struts-config.xml
<xml>
<mapping>
...
</mapping>
</xml>
SelectPromoForm.java
Form Bean
Auto set()
+ validate()
3.b
3.a
GET SelectPromo.do?promo=ing2
1
doGet(Rq,Rp)
2
Action Servlet
Action object
SelectPromoAction.java
<html>
...
JSP
...
</html>
POJO
Promo.java

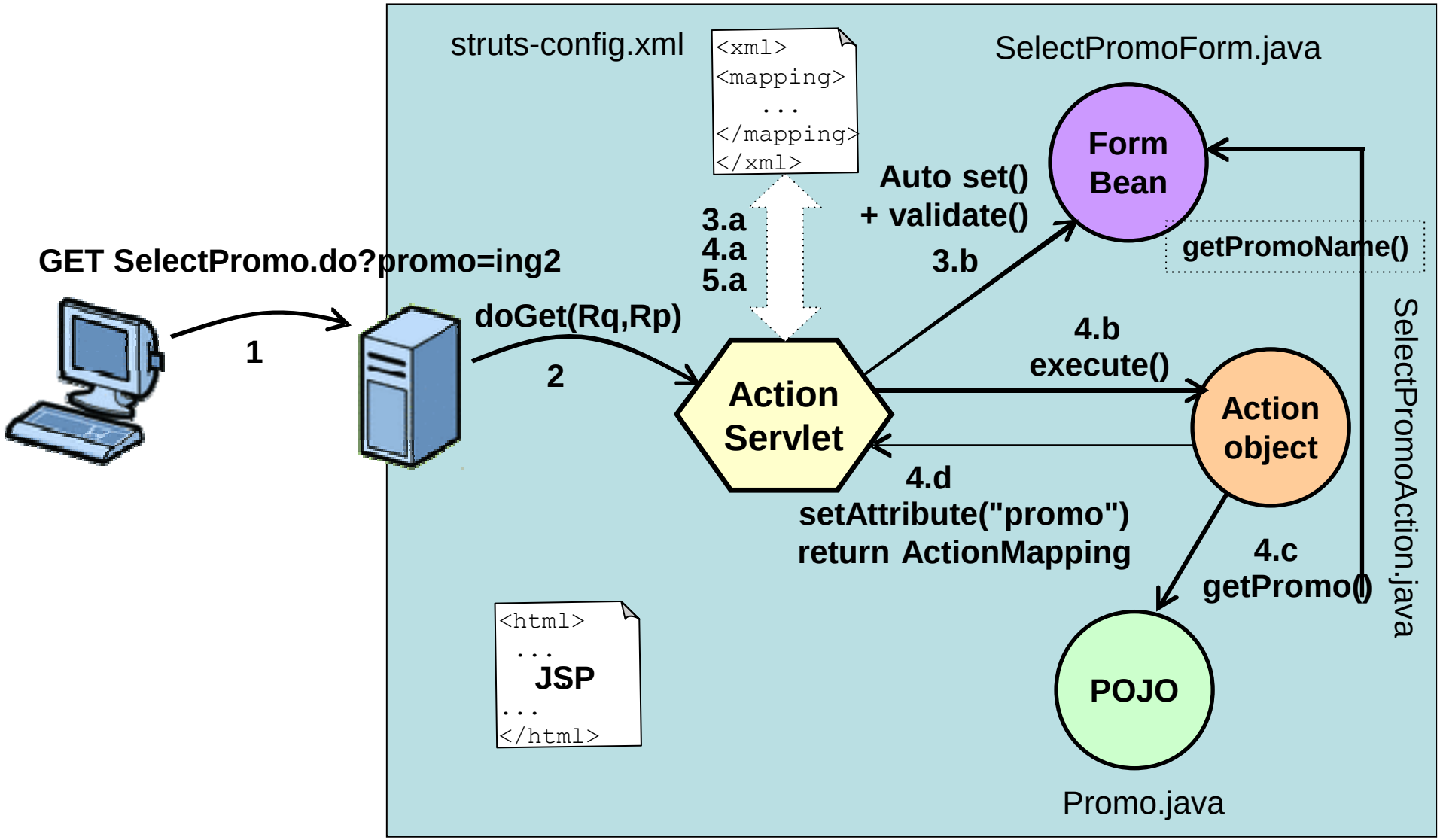
Form bean : SelectPromoForm.java

```
package arel; import ...;

public class SelectPromoForm extends ActionForm{

   private String promoName;
   public void setPromoName(String pN) {promoName=pN;}
   public String getPromoName (){ return promoName;}

   private static final String VALID_PROMOS = "ing1,ing2";

   public ActionErrors validate (ActionMapping mapping,
                                 HttpServletRequest request){
     ActionErrors errors = new ActionErrors();
     if(VALID_PROMOS.indexOf(promoName) == -1){
       errors.add("promo",
                new ActionMessage("error.promoField.notValid"));
     }
     return errors;
   }
}
```

Formulaire : form.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<!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>ARELV6.0</title>
</head>
<body>
<h1 align="center">AREL:L'école virtuelle de l'EISTI</h1>

<html:errors/>

<form method="GET" action="SelectPromo.do">Sélectionner la promo à
afficher:<select name="promoName" size="1">
<option>ing1</option>
<option>ing2</option>
</select><input type="SUBMIT"/></form>
</body>
</html>
```

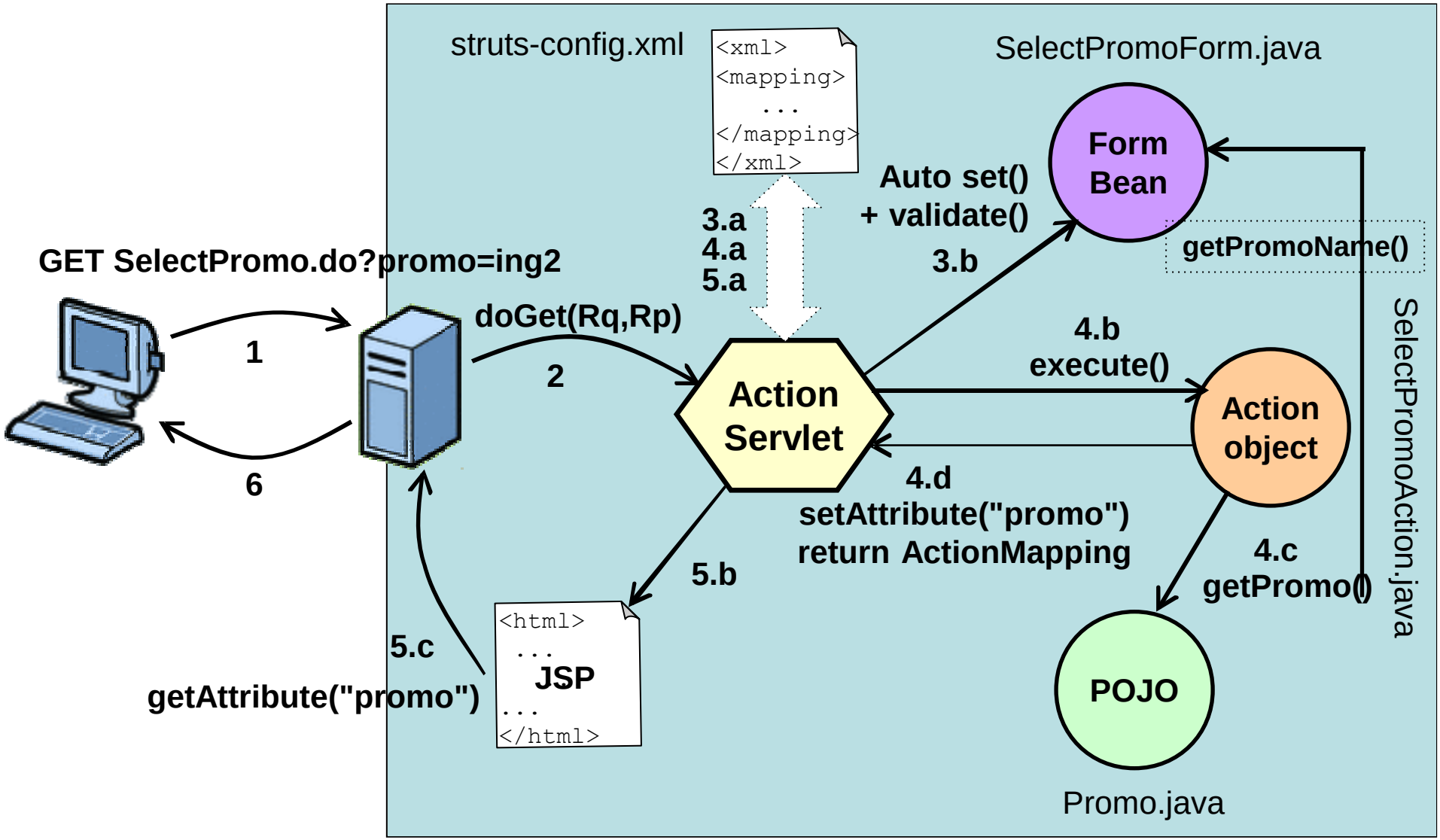
Étape 4


struts-config.xml
<xml>
<mapping>
...
</mapping>
</xml>
SelectPromoForm.java
Form Bean
Auto set()
+ validate()
3.b
getPromoName()
3.a
4.a
5.a
GET SelectPromo.do?promo=ing2
1
doGet(Rq,Rp)
2
Action Servlet
4.b
execute()
Action object
4.d
setAttribute("promo")
return ActionMapping
4.c
getPromo()
SelectPromoAction.java
<html>
...
JSP
...
</html>
POJO
Promo.java

Action object: SelectPromoAction.java

```
package arel;
import ...;

public class SelectPromoAction extends Action{

 public ActionForward execute( ActionMapping mapping,
                               ActionForm form,
                               HttpServletRequest request,
                               HttpServletResponse response){

  SelectPromoForm myForm=(SelectPromoForm)form;

  Promo promo = new Promo();
  List<String> result=promo.getPromo( myForm.getPromoName() );
  request.setAttribute("promo", result);

  return mapping.findForward("show_results");

 }

}
```

Étapes 5 & 6


struts-config.xml
<xml>
<mapping>
...
</mapping>
</xml>
SelectPromoForm.java
Form Bean
Auto set()
+ validate()
3.b
getPromoName()
SelectPromoAction.java
GET SelectPromo.do?promo=ing2
1
doGet(Rq,Rp)
2
3.a
4.a
5.a
Action Servlet
4.b
execute()
Action object
4.d
setAttribute("promo")
return ActionMapping
5.b
4.c
getPromo()
6
5.c
getAttribute("promo")
<html>
...
JSP
...
</html>
POJO
Promo.java

Notion supplémentaire

Les fichiers properties

Localisation : les fichiers properties

• Struts supporte nativement le mécanisme d'internationalisation et de localisation offert par la plateforme Java : Le **ResourceBundle** et la **Locale**.
•Un fichier .properties permet de stocker des informations sous la forme

```
Parametre = Valeur
```

• Avec Struts, ils sont chargés **automatiquement** en fonction de la langue préférée définie dans le browser (elles sont définies par ordre de préférence => cf. Mozilla ou IE).
• Si cette locale (le fichier *.properties* associé à cette langue) n'est pas présente, c'est la suivante (dans l'ordre des préférences) qui est choisie, etc… Si aucune n'est présente, on utilise le fichier *.properties* par défaut.
• Exemple :
MessageResources.properties (fichier par défaut)

```
error.promoField.notValid=Invalid promo entered.
```

MessageResources_fr.properties (fichier pour la locale _fr)

```
error.promoField.notValid=La promo entrée est invalide.
```

Localisation : les fichiers properties

• Il en sera de même pour toutes les langues et vous aurez autant de fichiers .*properties* que de langues que vous supporterez.

• Il vous faudra ensuite déclarer ces ressources dans le descripteur de configuration de votre application (*struts-config.xml*) :

<message-resources parameter="MessageResources" />

• Pour l'utiliser dans une JSP :

<bean:message key="error.promoField.notValid"/>