EISTI

Spring MVC

Manuel d'installation et d'utilisation

VILLETTE Charles, BEQUET Pierre, DE SAINT MARTIN Cédric, CALAY-ROCHE Vincent, NAUTRE Francois

20/05/2010

Logiciels Prérequis :

Eclipse (http://www.eclipse.org/)

Tomcat (http://tomcat.apache.org/)

Fichiers utilisés par/dans le tutorial :

http://www.springsource.org/download

Vous pourrez via cette page, et en vous inscrivant au préalable, télécharger les jar nécessaires à l'utilisation de Spring.

Etape 1

Dans un premier temps, vous aurez besoin de java 1.5 ou supérieur, ainsi que de tomcat et Eclipse.

Tout d'abord, créez un nouveau dossier « springmvc ». A l'intérieur de celui-ci, créez les dossiers « classes », « jsp » et « WEB-INF ».

Dans le dossier jsp, ajoutez la nouvelle jsp « index.jsp » :

```
<html>
  <body>
    <p>Hi</p>
  </body>
</html>
```

Dans le dossier WEB-INF, ajoutez le fichier « web.xml » et créer le dossier « src » :

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app 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">

  <welcome-file-list>
    <welcome-file>
      jsp/index.jsp
    </welcome-file>
  </welcome-file-list>

</web-app>
```

Exécutez ensuite votre application à l'aide de Tomcat. Vous devriez voir s'afficher un magnifique « Hi » signifiant que l'affichage de la jsp s'est déroulé correctement.

Etape 2

Dans le dossier WEB-INF, ajoutez maintenant le fichier « build.properties » :

```
appserver.home=/usr/share/tomcat5.5
appserver.lib=${appserver.home}/common/lib
```

Puis, toujours dans le même dossier, ajoutez le fichier « build.xml » :

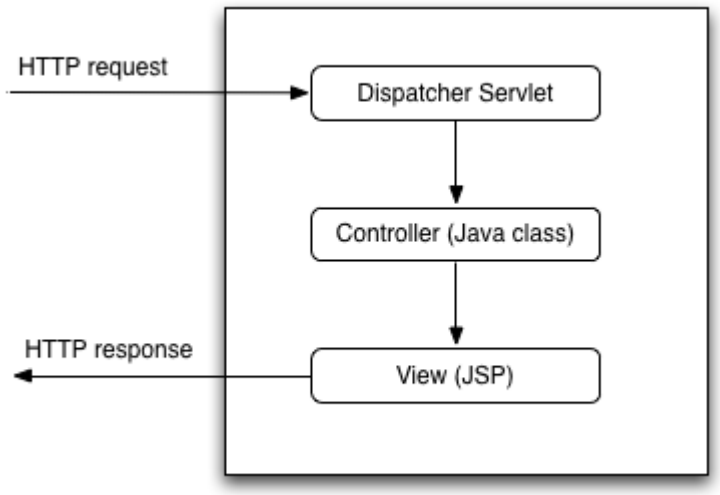
```
<?xml version="1.0" encoding="UTF-8"?>
<project basedir="." default="build">

  <property file="build.properties"/>
  <property name="src.dir" value="src"/>
  <property name="build.dir" value="classes"/>

  <path id="build.classpath">
      <fileset dir="lib">
          <include name="*.jar"/>
      </fileset>
      <fileset dir="${appserver.lib}"> <!-- servlet API classes: -->
          <include name="servlet*.jar"/>
      </fileset>
      <pathelement path="${build.dir}"/>
  </path>

  <target name="build">
      <mkdir dir="${build.dir}"/>
      <javac destdir="${build.dir}" source="1.5" target="1.5" debug="true" deprecation="false" optimize="false" failonerror="true">
          <src path="${src.dir}"/>
          <classpath refid="build.classpath"/>
      </javac>
  </target>

  <target name="clean" description="Clean output directories">
      <delete>
          <fileset dir="${build.dir}">
              <include name="**/*.class"/>
          </fileset>
      </delete>
  </target>

</project
```

Dans le dossier WEB-INF, créez le dossier « lib » et ajoutez-y les jar suivants :

- spring-framework/dist/spring.jar
- spring-framework/dist/modules/spring-webmvc.jar
- spring-framework/lib/jakarta-taglibs/standard.jar
- spring-framework/lib/jakarta-commons/commons-logging.jar
- spring-framework/lib/j2ee/servlet-api.jar
- spring-framework/lib/j2ee/jstl.jar

Etape 3

→ Comment marche Spring MVC ? Basiquement, de la même façon que Struts :



En se basant sur l'url de la requête HTTP, le DispatcherServlet appelle le contrôleur correspondant.

Une vue est ensuite rendu et envoyé en tant que réponse HTTP.

Dans le fichier web.xml, nous allons déclarer le dispatcher de Servlet et mapper les *.html.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app 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" >

  <servlet>
    <servlet-name>springmvc</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>springmvc</servlet-name>
    <url-pattern>*.html</url-pattern>
  </servlet-mapping>

  <welcome-file-list>
    <welcome-file>
      jsp/index.jsp
    </welcome-file>
  </welcome-file-list>

</web-app>
```

Nous allons maintenant créer le fichier de configuration « WEB-INF/springmvc-servlet.xml » (nom basé sur la servlet) :

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">

  <bean name="/hello_world.html" class="springmvc.web.HelloWorldController"/>

  <bean id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver">
      <property name="viewClass" value="org.springframework.web.servlet.view.JstlView"/>
      <property name="prefix" value="/jsp/"/>
      <property name="suffix" value=".jsp"/>
  </bean>

</beans>
```

Dans ce fichier, nous avons ainsi :

- mappé l'url /hello_world.html au contrôleur HelloWorldController.
- Déclaré un viewResolver : quand la vue view_name est appelé par le contrôleur, le fichier /jsp/view_name.jsp sera utilisé.

Nous allons maintenant créer le contrôleur correspondant à notre vue, dans le dossier :

WEB-INF/src/springmvc/web/HelloWorldController.java

```java
package springmvc.web;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.Controller;

public class HelloWorldController implements Controller {

    public ModelAndView handleRequest(HttpServletRequest request,
            HttpServletResponse response) throws ServletException, IOException {

        String aMessage = "Hello World MVC!";

        ModelAndView modelAndView = new ModelAndView("hello_world");
        modelAndView.addObject("message", aMessage);

        return modelAndView;
    }
}
```

Le contrôleur appelle la vue hello_world, affichant « message ».

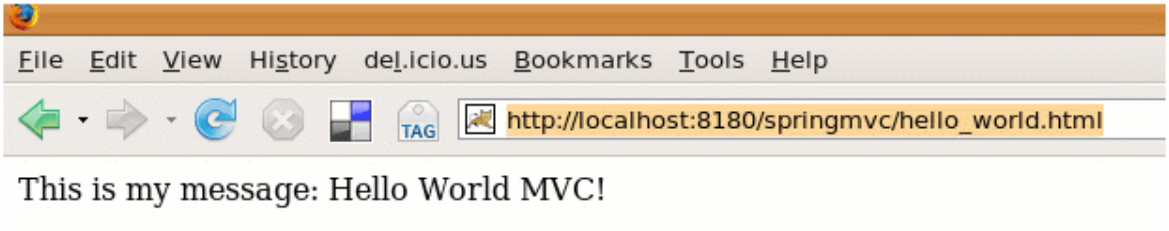
Nous allons maintenant nous occuper de la vue : jsp/hello_world.jsp

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<body>
        <p>This is my message: ${message}</p>
</body>
</html>
```

Cette jsp affichera l'attribut précédemment déclaré dans le contrôleur.

Vous pouvez enfin afficher le message dans votre navigateur préféré (IE n'est pas votre navigateur préféré…).

Si votre résultat est le suivant, ça marche !



Etape 4

Nous allons maintenant utiliser le modèle MVC et le mettre en place.

Nous allons créer une page affichant une liste de voitures et leurs caractéristiques.

Nous avons ainsi besoin :

- D'un modèle de classes : Car et Brand
- Un manager de classe : afin d'obtenir la liste de voitures
- Un contrôleur de class : va utiliser les méthodes du manager
- Une vue (jsp) : affiche la liste des voitures

→ Modèles (Brand.java et Car.java) :

```java
package springmvc.model;

public class Brand {
    private Long id;
    private String name;
    private String country;

    public String getCountry() {
        return country;
    }
    public void setCountry(String country) {
        this.country = country;
    }
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

```java
package springmvc.model;

import java.math.BigDecimal;

public class Car {
    private Long id;
    private Brand brand;
    private String model;
    private BigDecimal price;

    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public Brand getBrand() {
        return brand;
    }
    public void setBrand(Brand brand) {
        this.brand = brand;
    }
    public String getModel() {
        return model;
    }
    public void setModel(String model) {
        this.model = model;
    }
    public BigDecimal getPrice() {
        return price;
    }
    public void setPrice(BigDecimal price) {
        this.price = price;
    }
}
```

→ Manager Class

A créer dans WEB-INF/src/springmvc/service/CarManager.java

```
package springmvc.service;

import java.math.BigDecimal;
import java.util.LinkedList;
import java.util.List;

import springmvc.model.Brand;
import springmvc.model.Car;

public class CarManager {

    private static List<Car> carList;

    static {
        Brand brand1 = new Brand();
        brand1.setId((long)1);
        brand1.setName("Mercedes");
        brand1.setCountry("Germany");

        Brand brand2 = new Brand();
        brand2.setId((long)2);
        brand2.setName("Peugeot");
        brand2.setCountry("France");

        Car car1 = new Car();
        car1.setId((long)1);
        car1.setBrand(brand1);
        car1.setModel("SL 500");
        car1.setPrice(new BigDecimal(40000));

        Car car2 = new Car();
        car2.setId((long)2);
        car2.setBrand(brand2);
        car2.setModel("607");
        car2.setPrice(new BigDecimal(35000));

        carList = new LinkedList<Car>();
        carList.add(car1);
        carList.add(car2);
    }

    public List<Car> getCarList() {
        return carList;
    }
}
```

→ Controller

Dans le fichier springmvc-servlet.xml, il nous faut déclarer l'URL :

```
<bean name="/list_cars.html" class="springmvc.web.CarListController"/>
```

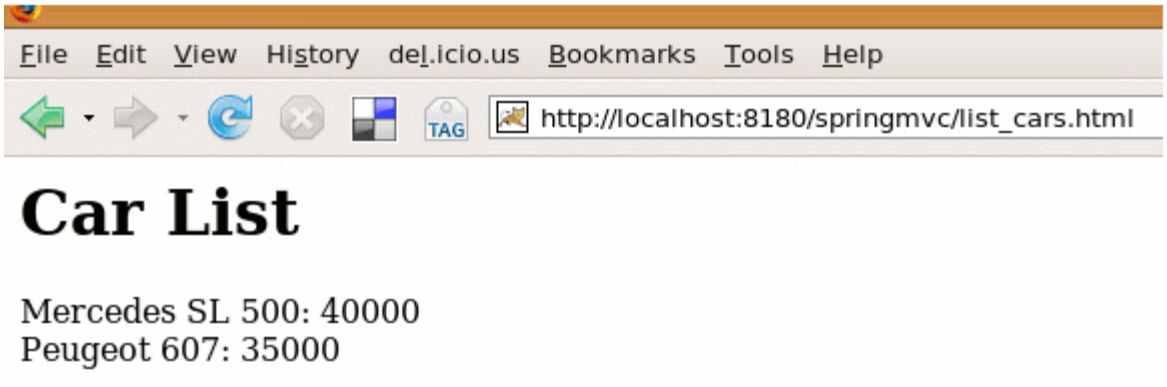
Puis nous créons le contrôleur WEB-INF/src/springmvc/web/CarListController.java :

```
package springmvc.web;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.Controller;

import springmvc.service.CarManager;

public class CarListController implements Controller {

    public ModelAndView handleRequest(HttpServletRequest arg0,
            HttpServletResponse arg1) throws Exception {

        CarManager carManager = new CarManager();

        ModelAndView modelAndView = new ModelAndView("carList");
        modelAndView.addObject("carList", carManager.getCarList());

        return modelAndView;
    }
}
```

Enfin, nous créons la vue jsp/carList.jsp :

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<body>
    <h1>Car List</h1>

    <c:forEach items="${carList}" var="car">
        ${car.brand.name} ${car.model}: ${car.price}
        <br />
    </c:forEach>

 </body>
</html>
```

Vous pouvez maintenant compiler votre application et l'exécuter sur un serveur Tomcat. Si tout fonctionne correctement, vous verrez s'afficher une superbe vue d'un garage ma fois assez réduit :



Et voila !

Bonne utilisation de Spring MVC !