L'utilisation de frameworks pour le développement avec Java EE

Module 7 – Spring MVC

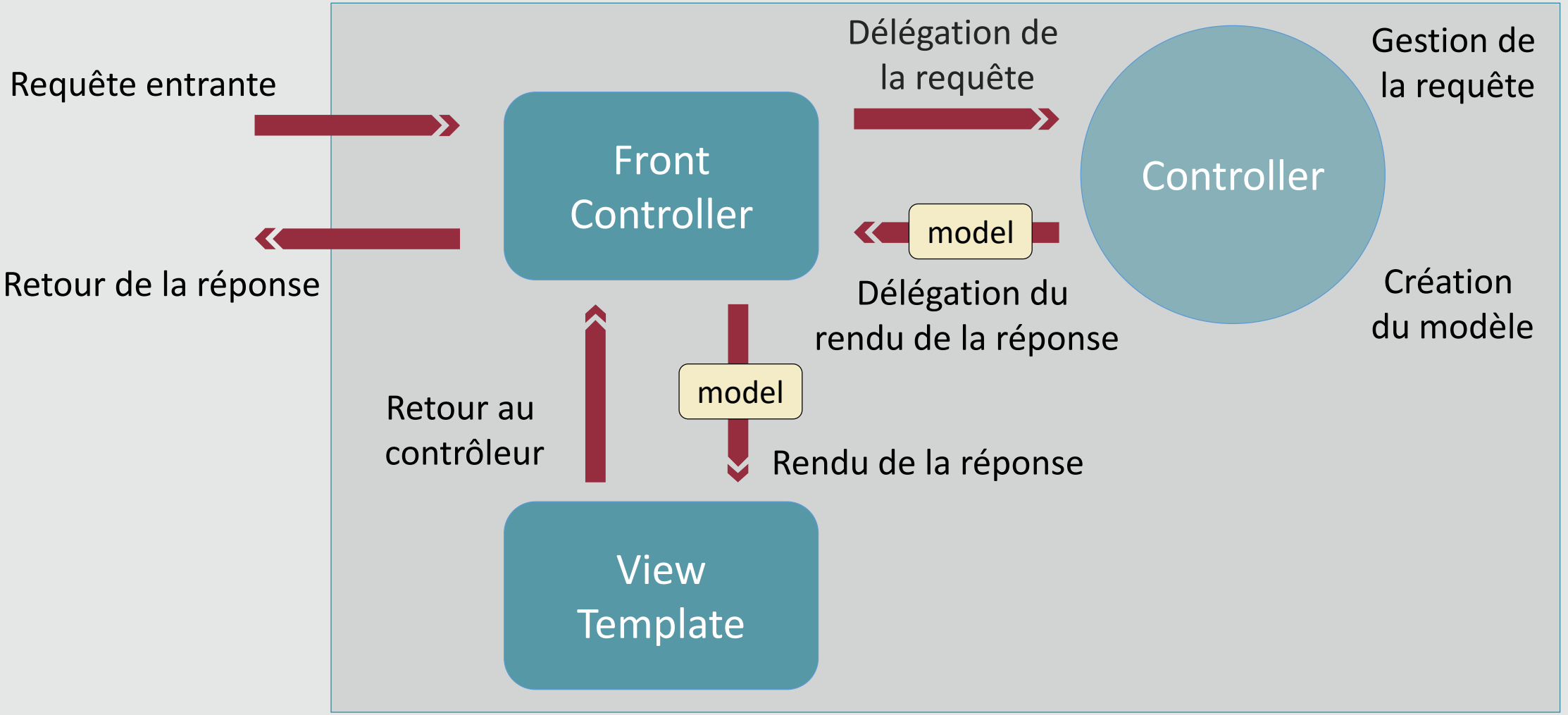
Objectifs

- Savoir créer un projet utilisant Spring MVC
- Comprendre le fonctionnement Contrôleur/Vue/Modèle
- Savoir traiter l'affichage ainsi que la validation d'un formulaire
- Construire une application internationalisée
- Permettre la validation des données
- Utiliser les Web Services REST dans un environnement Spring

Le Modèle MVC - Rappels

- MVC est un design pattern destiné aux interfaces graphiques
- Il est composé de trois types de modules
  - Le modèle qui contient les données à afficher
  - La vue qui contient la partie présentation
  - Le contrôleur qui contient la logique applicative et gère les actions utilisateurs

Spring MVC


Requête entrante
Retour de la réponse
Front Controller
Délégation de la requête
Controller
Gestion de la requête
model
Délégation du rendu de la réponse
Création du modèle
Retour au contrôleur
model
Rendu de la réponse
View Template

Spring MVC

Configuration du Front Controller

- Dans le fichier web.xml

```
<servlet>
  <servlet-name>dispatcher</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>dispatcher</servlet-name>
  <url-pattern>/app/*</url-pattern>
</servlet-mapping>
```

- Implique que le fichier de configuration Spring s'appelle dispatcher-servlet.xml

Spring MVC

Configuration du Front Controller

- Dans le fichier web.xml
  - Choix d'un autre nom de fichier

```
<servlet>
  <servlet-name>dispatcher</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>/WEB-INF/spring-configuration.xml</param-value>
  </init-param>
</servlet>

<servlet-mapping>
  <servlet-name>dispatcher</servlet-name>
  <url-pattern>/app/*</url-pattern>
</servlet-mapping>
```

VOIR DEMO

Spring MVC

Configuration Spring

- Dans le fichier dispatcher-servlet.xml

```
<context:annotation-config />

<context:component-scan
    base-package="fr.eni.mvc.controller" />

<bean id="viewResolver" class="org.springframework.web.servlet.view.UrlBasedViewResolver">
    <property name="viewClass" value="org.springframework.web.servlet.view.JstlView"/>
    <property name="prefix" value="/WEB-INF/jsp/"/>
    <property name="suffix" value=".jsp"/>
</bean>
```

Spring MVC

Le contrôleur

- C'est un Bean Spring
  - Avec l'annotation @Controller
- Sert à traiter une requête HTTP
- Il valide la requête, remplit le modèle et redirige vers la vue associée
- Les requêtes peuvent être mappées au niveau de la classe ou des méthodes

Spring MVC

Le contrôleur

- Mapping sur une méthode

```
@Controller
public class HelloController {

    @RequestMapping(method=RequestMethod.GET, path="/hello")
    public String hello(){
        return "hello";
    }
}
```

http://localhost:8080/01-SpringMVC-Base/app/hello

VOIR DEMO

Spring MVC

Le contrôleur

- Mapping sur une classe

```
@Controller
@RequestMapping(path="/bonjour")
public class BonjourController {

    @RequestMapping(method=RequestMethod.GET)
    public String bonjour(){
        return "bonjour";
    }

    @RequestMapping(method=RequestMethod.GET, path="/hello")
    public String hello(){
        return "hello";
    }
}
```

http://localhost:8080/01-SpringMVC-Base/app/bonjour

http://localhost:8080/01-SpringMVC-Base/app/bonjour/hello

VOIR DEMO

Spring MVC

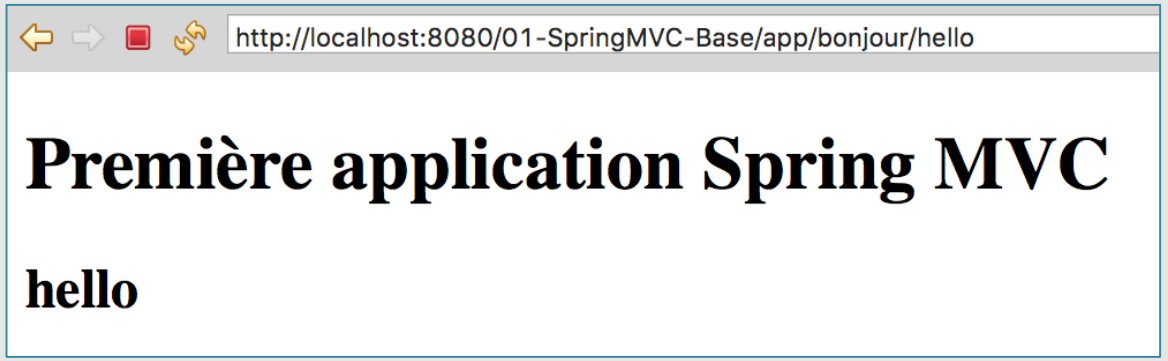
La vue

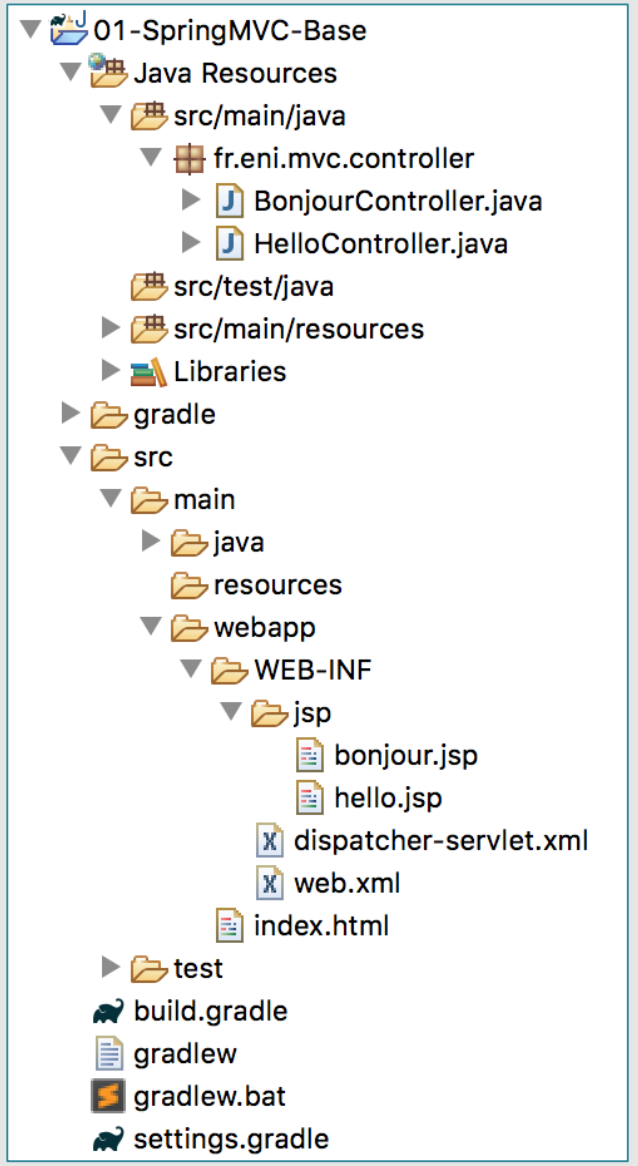
- Mapping entre le nom défini dans le contrôleur et le fichier réel
- Pas de nom de fichier dans le contrôleur
- Les vues sont généralement placées dans le répertoire WEB-INF

Spring MVC

La vue

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Insert title here</title>
    </head>
    <body>
        <h1>Première application Spring MVC</h1>
        <h2>hello</h2>
    </body>
</html>
```





VOIR DEMO

Spring MVC

Mise en place de la structure MVC

Démonstration

Spring MVC

ModelMap

- Le contrôleur peut transmettre des attributs à la vue
  - Utilisation de ModelMap

```
@Controller
@RequestMapping(path="/heure")
public class HeureController {

    @RequestMapping(method=RequestMethod.GET)
    public String bonjour(ModelMap map){

        SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
        String chaine = sdf.format(new Date());
        map.addAttribute("heureFormate", chaine);

        List<String> listeS = new ArrayList<>();
        listeS.add("A");
        listeS.add("B");
        listeS.add("C");
        map.addAttribute("liste", listeS);

        return "dateHeure";
    }
}
```

VOIR DEMO

Spring MVC

ModelMap

- Le contrôleur peut transmettre des attributs à la vue
  - Utilisation de ModelMap

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Insert title here</title>
    </head>
    <body>
        <h1>Application Spring MVC</h1>
        <h2>Heure générée par le serveur</h2>
        <p>${heureFormate}</p>

        <h2>Liste de chaines :</h2>
        <p>${liste }</p>
    </body>
</html>
```

VOIR DEMO

Spring MVC

ModelAndView

- Le contrôleur peut transmettre des attributs à la vue
  - Utilisation de ModelAndView

```
@RequestMapping(method=RequestMethod.GET, path="/hello")
public ModelAndView bonjour(){

    ModelAndView mav = new ModelAndView( "hello",
                                         "msg",
                                         "Message de la part du controleur HelloController");

    return mav;
}
```

VOIR DEMO

Spring MVC

Formulaire

- Passage d'un paramètre

```
<p><a href="app/bonjour?nom=dupond">Lien vers app/bonjour?nom=dupond</a></p>
```

```
@RequestMapping(method=RequestMethod.GET, path="/bonjour")
public ModelAndView bonjourNom(String nom){

    ModelAndView mav = new ModelAndView( "hello",
                                         "msg",
                                         "Bonjour " + nom + " !");

    return mav;
}
```

VOIR DEMO

Spring MVC

Formulaire

- Passage de plusieurs paramètres

```
<form action="app/ajout" method="POST">
    <div>
        <label for="nom">Nom : </label>
        <input type="text" id="nom" name="nom">
    </div>
    <div>
        <label for="prenom">Prénom : </label>
        <input type="text" id="prenom" name="prenom">
    </div>
    <button type="submit">Go !</button>
</form>
```

```
@RequestMapping(method=RequestMethod.POST, path="/ajout")
public ModelAndView ajout(String prenom, String nom){

    ModelAndView mav = new ModelAndView( "hello",
                                         "msg",
                                         "La personne (" + prenom + " " + nom + ") a bien ét uté.");
    return mav;
}
```

VOIR DEMO

Spring MVC

Passage de paramètres

Démonstration

Spring MVC

Formulaire

- Le modèle est une simple classe POJO

```
public class Adresse {

    private String codePostal;
    private String ville;

    public Adresse() {
        this("", "");
    }

    public Adresse(String codePostal, String ville) {
        this.codePostal = codePostal;
        this.ville = ville;
    }
    // [...}
```

```
public class Personne {

    private String nom;
    private String prenom;
    private Adresse adresse;

    public Personne() {
        this("", "", new Adresse());
    }

    public Personne(String nom, String prenom, Adresse adresse
        this.nom = nom;
        this.prenom = prenom;
        this.adresse = adresse;
    }
    // [...}
```

VOIR DEMO

Spring MVC

Formulaire

- Appel de la page d'ajout via le contrôleur
  - Fichier index.html :

```
<p><a href="app/ajout">Lien vers app/ajout</a></p>
```

  - Méthode ajout() du contrôleur

```
@RequestMapping(method=RequestMethod.GET, path="/ajout")
public ModelAndView ajout(){
    Personne current = new Personne("Legrand", "Lucie", new Adresse("44000", "Nantes"));
    ModelAndView mav = new ModelAndView("ajout", "command", current);

    return mav;
}
```

VOIR DEMO

Spring MVC

Formulaire

- Utilisation du taglib de Spring dans la JSP
  - <%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>

```
<form:form action="ajout" method="POST">
    <div>
        <form:label path="nom">Nom : </form:label>
        <form:input path="nom"/>
    </div>
    <div>
        <form:label path="prenom">Prénom : </form:label>
        <form:input path="prenom"/>
    </div>
    <div>
        <form:label path="adresse.codePostal">Code Postal : </form:label>
        <form:input path="adresse.codePostal"/>
    </div>
    <div>
        <form:label path="adresse.ville">Ville : </form:label>
        <form:input path="adresse.ville"/>
    </div>
    <div>
        <input type="submit" value= "Ajouter" />
    </div>
</form:form>
```

Spring MVC

Formulaire

- Validation du formulaire
  - Méthode ajoutValidation() du contrôleur

```
@RequestMapping(method=RequestMethod.POST, path="/ajout")
public ModelAndView ajoutValidation(Personne p){

    System.out.println("Traitement de " + p);
    ModelAndView mav = new ModelAndView("success", "nouveau", p);

    return mav;
}
```

VOIR DEMO

Spring MVC

Formulaire

- Validation du formulaire (autre méthode)

```
@RequestMapping(method=RequestMethod.POST, path="/ajoutAutre")
public String ajoutValidationAutre(@ModelAttribute("nouveau") Personne p){

    System.out.println("Traitement de " + p);

    return "success";
}
```

VOIR DEMO

Attributs du modèle dans la requête

- @ModelAttribute sur une méthode
- Cette méthode sera appelée avant chaque mapping
  - Attribut généré pour chaque requête

```
@ModelAttribute("heure")
public String getHeure(){
    System.out.println("Appel de getHeure");
    SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
    String dh = sdf.format(new Date());
    return dh;
}
```

```
Appel de getHeure
Appel de ajout()
Appel de getHeure
Traitement de Personne [nom=Legrand, prenom=Lucie, adresse=Adresse
[codePostal=44000, ville=Nantes]]
```

VOIR DEMO

Spring MVC

Attributs du modèle dans la session

- Utilisation de l'annotation @SessionAttributes sur la classe

```
@ModelAttribute("heureSession")
public String getHeureSession(){
    System.out.println("Appel de getHeureSession");
    SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
    String dh = sdf.format(new Date());
    return dh;
}
```

```
@Controller
@SessionAttributes({"heureSession"})
public class PersonneController {
```

```
Appel de getHeure
Appel de getHeureSession
Appel de ajout()
Appel de getHeure
Traitement de Personne [nom=Legrand, prenom=Lucie, adresse=Adr
[codePostal=44000, ville=Nantes]]
```

VOIR DEMO

Spring MVC

Attributs du modèle dans l'application

- Injection du contexte

```
@Autowired
ServletContext context;
```

- Annotation d'une méthode avec @PostConstruct

```
@PostConstruct
public void getHeureContext(){
    System.out.println("Appel de getHeureContext");
    SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
    String dh = sdf.format(new Date());
    context.setAttribute("heureContext", dh);
}
```

VOIR DEMO

Spring MVC

Injection d'objets prédéfinis

- Déclaration et injection d'un premier bean

```
<bean id = "pers1" class = "fr.eni.mvc.bean.Personne">
    <property name="nom" value="Leblond" />
    <property name="prenom" value="Margot"/>
    <property name="adresse">
        <bean class = "fr.eni.mvc.bean.Adresse">
            <property name="codePostal" value="44100"/>
            <property name="Ville" value="Nantes"/>
        </bean>
    </property>
</bean>
```

```
@Resource(name="pers1")
Personne personneInjectee1;
```

```
@RequestMapping(path = "/ajoutInjPers1", method = RequestMeth    ET)
public ModelAndView ajoutInjection1() {
    return new ModelAndView("ajout", "command", personneInjec    );
}
```

VOIR DEMO

Spring MVC

Injection d'objets prédéfinis

- Déclaration et injection d'un second bean

```
<bean id = "pers2" class = "fr.eni.mvc.bean.Personne">
    <property name="nom" value="Lebrun" />
    <property name="prenom" value="Marie"/>
    <property name="adresse">
        <bean class = "fr.eni.mvc.bean.Adresse">
            <property name="codePostal" value="44700"/>
            <property name="Ville" value="Orvault"/>
        </bean>
    </property>
</bean>
```

```
@Autowired
@Qualifier("pers2")
Personne personneInjectee2;
```

```
@RequestMapping(path = "/ajoutInjPers2", method = RequestMethod.
public ModelAndView ajoutInjection2() {
    return new ModelAndView("ajout", "command", personneInjectee
}
```

VOIR DEMO

Spring MVC

Formulaire

Démonstration

Spring MVC

Internationalisation (i18n)

- Externalisation des chaînes de caractères dans des fichiers properties
- Un fichier properties par langue gérée par l'application


```
# message_fr.properties
index.bonjour=Bonjour
index.bienvenue=Bienvenue sur ma page
hello= Bien le bonjour {0} {1} et bienvenue sur mon application
```

```
# message_en.properties
index.bonjour=Hello
index.bienvenue=Welcome to my page
hello= Hello {0} {1} and welcome to my application
```

Spring MVC

Internationalisation (i18n)

- Déclaration du localeResolver

```
<bean id="localeResolver"
    class="org.springframework.web.servlet.i18n.SessionLocaleResolver">
    <property name="defaultLocale" value="fr" />
</bean>
```

- Déclaration du ou des fichiers properties

```
<bean id="messageSource"
   class="org.springframework.context.support.ResourceBundleMessageSource">
    <property name="basename" value="messages" />
</bean>
```

```
<bean id="messageSource"
   class="org.springframework.context.support.ResourceBundleMessa      rce">
    <property name="basenames" >
        <list>
            <value>messages</value>
            <value>erreurs</value>
        </list>
    </property>
</bean>
```

VOIR DEMO

Spring MVC

Internationalisation (i18n)

- Traitement des messages dans le contrôleur
  - Injection des objets SessionLocaleResolver et MessageSource

```
@Autowired
 private  SessionLocaleResolver slr;

@Autowired
private MessageSource ms;
```

  - Modification de la locale par défaut

```
http://localhost:8080/04-SpringMVC-i18n/app/i18n?lg=fr
```

```
@RequestMapping(method=RequestMethod.GET, path="/i18n")
public String testInter(String lg, ModelMap map){

    Locale locale = new Locale(lg);
    slr.setDefaultLocale(locale);
```

VOIR DEMO

Spring MVC

Internationalisation (i18n)

- Traitement des messages dans le contrôleur
  - Récupération des messages

```
System.out.println("displayLanguage : " + locale.getDisplayLanguage());
System.out.println("language : " + locale.getLanguage());

String message1 = ms.getMessage("index.bonjour", null, locale);
String message2 = ms.getMessage("index.bienvenue", null, locale);
String message3 = ms.getMessage("hello", new String[]{"Fred", "Leblond"}, locale);
```

```
displayLanguage : français
language : fr
Récupération des messages dans le controleur :
message 1 : Bonjour
message 2 : Bienvenue sur ma page
message 3 : Bien le bonjour Fred Leblond et bienvenue sur mon application
```

```
displayLanguage : anglais
language : en
Récupération des messages dans le controleur :
message 1 : Hello
message 2 : Welcome to my page
message 3 : Hello Fred Leblond and welcome to my applica
```

VOIR DEMO

Spring MVC

Internationalisation (i18n)

- Traitement des messages dans la vue

```
<%@ taglib uri="http://www.springframework.org/tags" prefix="spring"%>

<spring:message code="index.bonjour" />

<spring:message code="index.bienvenue" />

<spring:message code="hello"
   arguments="${prenom};${nom}"
   htmlEscape="true"
   argumentSeparator=";"/>
```

VOIR DEMO

Spring MVC

Internationalisation (i18n)

Démonstration

Spring MVC

Validation

- Utilisation de Bean Validation (JSR 303)
  - C'est une spécification
- Utilisation de Hibernate Validator
  - C'est l'implémentation de référence

Spring MVC

Validation

- Liste des contraintes fournies par Bean Validation

Contrainte	Signification (sur l'élément)
@AssertTrue	Doit être à true
@AssertFalse	Doit être à false
@Min, @DecimalMin	Doit être supérieur à …
@Max, @DecimalMax	Doit être inférieur à …
@Digits	Définit le nombre de chiffres
@Size	Doit être entre deux tailles

Contrainte	Signification (sur l'élément)
@Null	Doit être nul
@NotNull	Doit être non nul
@Past	Doit être dans le passé
@Future	Doit être dans le futur
@Pattern	Doit respecter une RegExp

Spring MVC

Validation

- Quelques contraintes fournies par Hibernate Validator

Contrainte	Signification (sur l'élément)
@CreditCardNumber	Représente un numéro de carte de crédit
@Email	Possède le format email
@NotEmpty	Ne peut pas être nul ni vide
@NotBlank	Ne peut pas être nul ni vide après trim()
@URL	Représente une URL valide
@Range	Doit être dans l'intervalle

Spring MVC

Validation

- Création de la classe bean

```
public class Personne {

    @NotBlank
    private String nom;

    @Email
    private String email;

    @Pattern(regexp="^[0-9]{5}$")
    private String codePostal;

    @Min(value=0)
    @Max(value=100)
    private int age;

    // [...]
```

```
import javax.validation.constraints.Max;
import javax.validation.constraints.Min;
import javax.validation.constraints.Pattern;

import org.hibernate.validator.constraints.Email;
import org.hibernate.validator.constraints.NotBlank;
```

Spring MVC

Validation

- Création du formulaire

```
<form:form action="formulaire" method="POST" commandName="pers">
    <table>
        <tr>
            <td><form:label path="nom">Nom : </form:label></td>
            <td><form:input path="nom" /></td>
            <td><form:errors path="nom" cssClass="erreur"/> </td>
        </tr>
        <tr>
            <td><form:label path="email">Email : </form:label></td>
            <td><form:input path="email" /></td>
            <td><form:errors path="email" cssClass="erreur"/> </td>
        </tr>
        <tr>
            <td><form:label path="codePostal">Code Postal : </form:label></td>
            <td><form:input path="codePostal" /></td>
            <td><form:errors path="codePostal" cssClass="erreur"/> </td>
        </tr>
        <tr>
            <td><form:label path="age">Age : </form:label></td>
            <td><form:input path="age" /></td>
            <td><form:errors path="age" cssClass="erreur"/> </td>
        </tr>
    </table>
    <input type="submit" value = "Ajouer" />
</form:form>
```

Spring MVC

Validation

- Dans le contrôleur
  - Appel de méthode permettant l'affichage du formulaire

```
@RequestMapping(value="/formulaire", method=RequestMethod.GET)
public ModelAndView appelFormulaire(){
    Personne p = new Personne();
    return new ModelAndView("formulaire", "pers", p);
}
```

Spring MVC

Validation

- Dans le contrôleur
  - Appel de méthode validant le formulaire : @Valid

```
@RequestMapping(value="/formulaire", method=RequestMethod.POST)

public String validLogin(@Valid @ModelAttribute("pers") Personne p, BindingResult result){

    if (result.hasErrors())
        return "formulaire";
    else
        return "success";
}
```

VOIR DEMO

Spring MVC

Validation

- Configuration
  - Ajout de <mvc:annotation-driven /> permettant à l'annotation @Valid de fonctionner

```
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
    http://www.springframework.org/schema/mvc
    http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-4.0.xsd">


    <mvc:annotation-driven />
```

VOIR DEMO

Spring MVC

Validation

- Après validation du formulaire

VOIR DEMO

Spring MVC

Validation

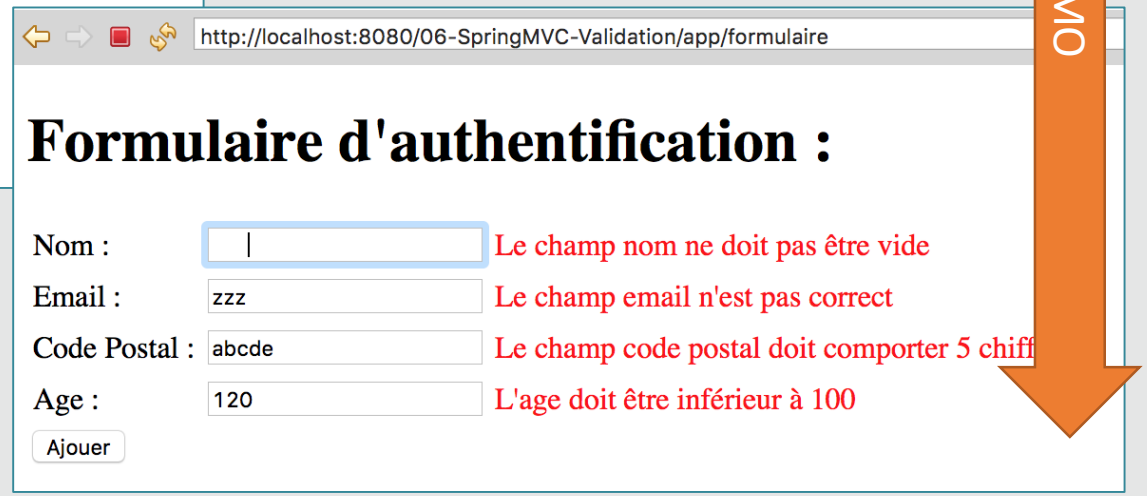
- Possibilité de personnaliser les messages d'erreur (1)

```
@NotBlank(message="Le champ nom ne doit pas être vide")
private String nom;

@Email(message="Le champ email n''est pas correct")
private String email;

@Pattern(regexp="^[0-9]{5}$", message="Le champ code postal doit comporter 5 chiffres")
private String codePostal;

@Min(value=0, message="L''age doit être supérieur à 0")
@Max(value=100, message="L''age doit être inférieur à 100")
private int age;
```



VOIR DEMO

Spring MVC

Validation

- Possibilité de personnaliser les messages d'erreur (2)
- Externalisation dans des fichiers properties
- Supporte l'internationalisation (i18n)
  - Modification du fichier de configuration

```
<bean id="messageSource"
    class="org.springframework.context.support.ReloadableResourceBundleMessageSource">
    <property name="basename" value="/WEB-INF/erreurs" />
    <property name="cacheSeconds" value="30" />
</bean>
```

Spring MVC

Validation

- Possibilité de personnaliser les messages d'erreur (2)

```
#/WEB-INF/erreurs.properties

NotBlank.pers.nom=Le champ nom ne doit pas être vide !!!
Email.pers.email=Le champ email n''est pas correct !!!
Pattern.pers.codePostal=Le champ code postal doit comporter 5 chiffres !!!
Min.pers.age=L''age doit être supérieur à 0 !!!
Max.pers.age=L''age doit être inférieur à 100 !!!
```

VOIR DEMO

Spring MVC

Validation

Démonstration

Spring MVC

Spring REST

- Spring utilise l'environnement MVC afin de proposer des services web de type REST

- Annotation de la classe contrôleur avec @RestController

```
@RestController
@RequestMapping("/test")
public class TestRestController {
```

Spring MVC

Spring REST

- Annotation des méthodes avec @RequestMapping
  - en indiquant le path (value)
  - en indiquant la méthode HTTP (method)

- Utilisation de @ResponseEntity

```
@RequestMapping(value = "/un", method=RequestMethod.GET)
public ResponseEntity<Integer> getInt() {
    return new ResponseEntity<Integer>(new Integer(123), HttpStatus.OK);
}
```

```
http://localhost:8080/07-SpringMVC-Rest/app/test/un
```

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Sat, 10 Nov 2017 15:54:00 GMT

123
```

Spring MVC

Spring REST

- Annotation des méthodes avec @RequestMapping
  - en indiquant le path (value)
  - en indiquant la méthode HTTP (method)

- Utilisation de @ResponseEntity

```
@RequestMapping(value = "/hello", method=RequestMethod.GET)
public ResponseEntity<String> hello() {
    return new ResponseEntity<String>("Hello World !", HttpStatus.OK);
}
```

```
http://localhost:8080/07-SpringMVC-Rest/app/test/hello
```

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/plain;charset=ISO-8859-1
Content-Length: 13
Date: Sat, 10 Nov 2017 15:54:00 GMT

Hello World !
```

Spring MVC

Spring REST

- Passage de paramètres

```
@RequestMapping(value = "/hello/{nom}", method=RequestMethod.GET)
public ResponseEntity<String> hello(@PathVariable("nom") String name) {

    String chaine = "Hello " + name;
    return new ResponseEntity<String>(chaine, HttpStatus.OK);
}
```

```
http://localhost:8080/07-SpringMVC-Rest/app/test/hello/Legrand
```

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/plain;charset=ISO-8859-1
Content-Length: 13
Date: Sat, 10 Nov 2017 15:54:00 GMT

Hello Legrand
```

Spring MVC

Spring REST

- Méthodes renvoyant un objet

```
@RequestMapping(value = "/{id}", method=RequestMethod.GET)
public ResponseEntity<Personne> getPersonne(@PathVariable("id") int id) {

    if (personnes.getListe().size() <= id )
        return new ResponseEntity<Personne>(HttpStatus.NOT_FOUND);
    else
        return new ResponseEntity<Personne>(personnes.getListe().get(id), HttpStatus.OK);
}
```

Spring MVC

Spring REST

- Méthodes renvoyant un objet

```
http://localhost:8080/07-SpringMVC-Rest/app/test/personnes/1
```

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Sat, 10 Nov 2017 15:54:00 GMT

{"nom":"Lemoyen","prenom":"Julie","adresse":{"codePostal":"75001","ville":"Paris"}}
```

```
http://localhost:8080/07-SpringMVC-Rest/app/test/personnes/111
```

```
HTTP/1.1 404 Introuvable
Server: Apache-Coyote/1.1
Content-Length: 0
Date: Sat, 10 Nov 2017 15:54:00 GMT
```

Spring MVC

Spring REST

- Méthodes renvoyant une liste d'objets

```
@RequestMapping(value = "/liste", method=RequestMethod.GET)
public ResponseEntity<List<Personne>> getPersonnes() {

    return new ResponseEntity<List<Personne>>(personnes.getListe(), HttpStatus.OK);
}
```

```
http://localhost:8080/07-SpringMVC-Rest/app/test/personnes/liste
```

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Sat, 10 Nov 2017 15:54:00 GMT

[{"nom":"Legrand","prenom":"Jules","adresse":{"codePostal":"59000","ville":"Lille"}},
{"nom":"Lemoyen","prenom":"Julie","adresse":{"codePostal":"75001","ville":"Paris"}},
{"nom":"Lepetit","prenom":"Julien","adresse":{"codePostal":"44000","ville":"Nantes"}}]
```

Spring MVC

Spring REST

- Utilisation des verbes HTTP
  - Méthode POST
  - Modification de l'en-tête de réponse HTTP Location
  - Retour du code 201 (créé)

```
@RequestMapping(method = RequestMethod.POST)
   public ResponseEntity<Void> createUser(@RequestBody Personne p, UriComponentsBuilder ucBuilder) {

       personnes.addPersonne(p);
       int nb = personnes.getListe().size();
       HttpHeaders headers = new HttpHeaders();
       headers.setLocation(ucBuilder.path("/{id}").buildAndExpand(nb).toUri());
       return new ResponseEntity<Void>(headers, HttpStatus.CREATED);
   }
```

Spring MVC

Spring REST

- Utilisation des verbes HTTP
  - Méthode POST
  - Modification de l'en-tête de réponse HTTP Location
  - Retour du code 201 (créé)

```
POST http://localhost:8080/07-SpringMVC-Rest/app/test/personnes/
```

```
HTTP/1.1 201 Crée
Server: Apache-Coyote/1.1
Location: http://localhost:8080/07-SpringMVC-Rest/app/6
Content-Length: 0
Date: Sat, 10 Nov 2017 15:54:00 GMT
```

```
{
"nom":"Lemoyen",
"prenom":"Julie",
"adresse":
      {"codePostal":"75001",
      "ville":"Paris"}
}
```

Spring MVC

Spring REST

- Utilisation des verbes HTTP
  - Méthode PUT

```
@RequestMapping(value = "/{id}", method = RequestMethod.PUT)
public ResponseEntity<Personne> updateUser(@PathVariable("id") int id, @RequestBody Personne p) {

    if (personnes.getListe().size() <= id )
        return new ResponseEntity<Personne>(HttpStatus.NOT_FOUND);
    else {
        personnes.getListe().set(id, p);
        return new ResponseEntity<Personne>(personnes.getListe().get(id), HttpStatus.OK);
    }
}
```

Spring MVC

Spring REST

- Utilisation des verbes HTTP
  - Méthode PUT

```
PUT http://localhost:8080/07-SpringMVC-Rest/app/test/personnes/1
```

```
{
"nom":"Leblond",
"prenom":"Aline",
"adresse":
      {"codePostal":"75002",
      "ville":"Paris"}
}
```

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Sat, 10 Nov 2017 15:54:00 GMT

{"nom":"Leblond","prenom":"Aline","adresse":{"codePostal":"75002","ville":"Paris"}}
```

Spring MVC

Spring REST

- Utilisation des verbes HTTP
  - Méthode PUT

```
PUT http://localhost:8080/07-SpringMVC-Rest/app/test/personnes/111
```

```
{
"nom":"Leblond",
"prenom":"Aline",
"adresse":
      {"codePostal":"75002",
      "ville":"Paris"}
}
```

```
HTTP/1.1 404 Introuvable
Server: Apache-Coyote/1.1
Content-Length: 0
Date: Sat, 10 Nov 2017 15:54:00 GMT
```

Spring MVC

Spring REST

- Utilisation des verbes HTTP
  - Méthode DELETE

```
@RequestMapping(value = "/{id}", method=RequestMethod.DELETE)
public ResponseEntity<Void> removePersonne(@PathVariable("id") int id) {
    try {
        personnes.removePersonne(id);
        return new ResponseEntity<Void>(HttpStatus.OK);
    } catch (Exception e) {
        return new ResponseEntity<Void>(HttpStatus.CONFLICT);
    }
}
```

Spring REST

- Utilisation des verbes HTTP
  - Méthode DELETE

```
DELETE http://localhost:8080/07-SpringMVC-Rest/app/test/personnes/1
```

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Length: 0
Date: Sat, 10 Nov 2017 15:54:00 GMT
```

```
DELETE http://localhost:8080/07-SpringMVC-Rest/app/test/personnes/111
```

```
HTTP/1.1 409 Conflit
Server: Apache-Coyote/1.1
Content-Length: 0
Date: Sat, 10 Nov 2017 15:54:00 GMT
```

Spring REST

Démonstration