

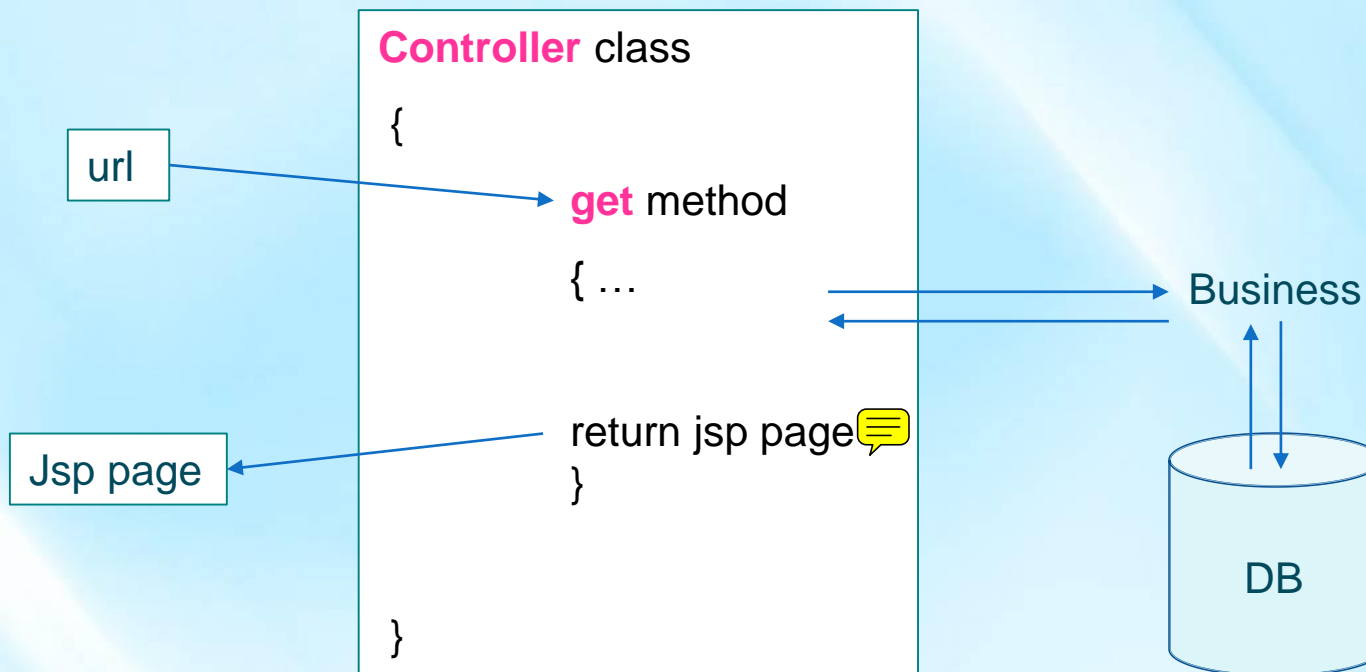
MODULE 6

CONTROLLER

TABLE OF CONTENT

- Navigation through Controller
- Creating a Controller
- Parameter in Url
- Model Dictionary
- Form Controller
- Redirection
- Unified Expression Language
- JSTL
- Form Validation

Navigation through Controller



Navigation through Controller

- ▶ Url \Rightarrow call of the get method of the corresponding controller
- ▶ Creation of a Controller class
- ▶ Get method
 - Possible call of business methods (cf Services)
 - \Rightarrow Possible access to database
 - Returns a jsp page
 - Data passed to the jsp page (through use of the Model class)

Creating a Controller

► Import

- `org.springframework.stereotype.Controller;`
- `import org.springframework.ui.Model;`
- `import org.springframework.validation.BindingResult;`
- `import org.springframework.web.bind.annotation.*;`

► Class annotation

- **`@Controller`**

► Get Method

- **`@RequestMapping (method = RequestMethod.GET)`**
- Returns the name of the jsp page

Creating a Controller

- ▶ Request path
 - Path to call the controller
 - Through annotation on controller class and/or on get method
 - **`@RequestMapping (...value = ...)`**

↓
path

Creating a Controller

- ▶ E.g.,
 - Controller

```
@Controller
@RequestMapping(value="/hello")
public class HelloController {

    @RequestMapping(value="/say", method=RequestMethod.GET)
    public String sayHello() {

        return "helloPage";
    }
}
```


Name of the jsp page

- Browser

http://localhost:8080/first/hello/say

Root of the project (see *application.yml*)

Query Parameter in Url

- ▶ To take input through query parameter
- ▶ For each parameter (as input of the get method)
 - **@RequestParam**
 - **required** : true/false 
 - **defaultValue** : if no value given for the parameter in the url

Query Parameter in Url

- ▶ E.g,
 - Controller

```
@Controller
@RequestMapping(value="/hello")
public class HelloController {

    @RequestMapping (value="/say", method=RequestMethod.GET)
    public String sayHello (@RequestParam(required=false, defaultValue="World!") String name)
    {
        ...
    }
}
```

- Browser

http://localhost:8080/first/hello/say

http://localhost:8080/first/hello/say?name=john

Query Parameter in Url

- ▶ If different methods to call according to the presence or absence of parameters
 - In get method declaration
 - **`@RequestMapping (... params = {...})`**
 - List of parameter names
 - Helps the DispatcherServlet to find the right method to call

Query Parameter in Url

- ▶ E.g,
 - Controller

```
@Controller
@RequestMapping(value="/hello")
public class HelloController {

    @RequestMapping (value="/say", method=RequestMethod.GET) ①
    public String sayHello ()
    { ... }

    @RequestMapping ( value="/say",                               ②
                      params={"name"},
                      method=RequestMethod.GET)
    public String sayHello (@RequestParam(required=false, defaultValue="World!") String name)
    { ... }
}
```

- Browser

http://localhost:8080/first/hello/say ①

http://localhost:8080/first/hi/say?name=John ②

Model Dictionary

- ▶ To pass model data from controller to the view (jsp page)
- ▶ By default, scope = request
 - For session scope (see module 8 Session Attribute)

Model Dictionary

- ▶ Through an object of the Model class
 - **`org.springframework.ui.Model`** class
 - Key-value pairs dictionary
 - Key (Id) : String
 - Value : object
 - To add an entry: `addAttribute("modelAttributeID",value)`
 - Input of the get method
- ▶ Use of the data
 - Data added to the model dictionary by the controller
 - Data accessed (through its id) in the jsp page
 - Syntax : **`${modelAttributeID}`**
 - See *Unified Expression Language*

Model Dictionary

- ▶ E.g,
 - Controller

```
@Controller
@RequestMapping (value="/hi")
public class HelloWorldController {

    @RequestMapping (value="/say",
                    params={"name"},
                    method=RequestMethod.GET)
    public String sayHi (@RequestParam(required=false, defaultValue="World!") final String name,
                        final Model model) {
        model.addAttribute("nameToDisplay", name);
        return "hello";
    }
}
```

Model Dictionary

- hello.jsp page

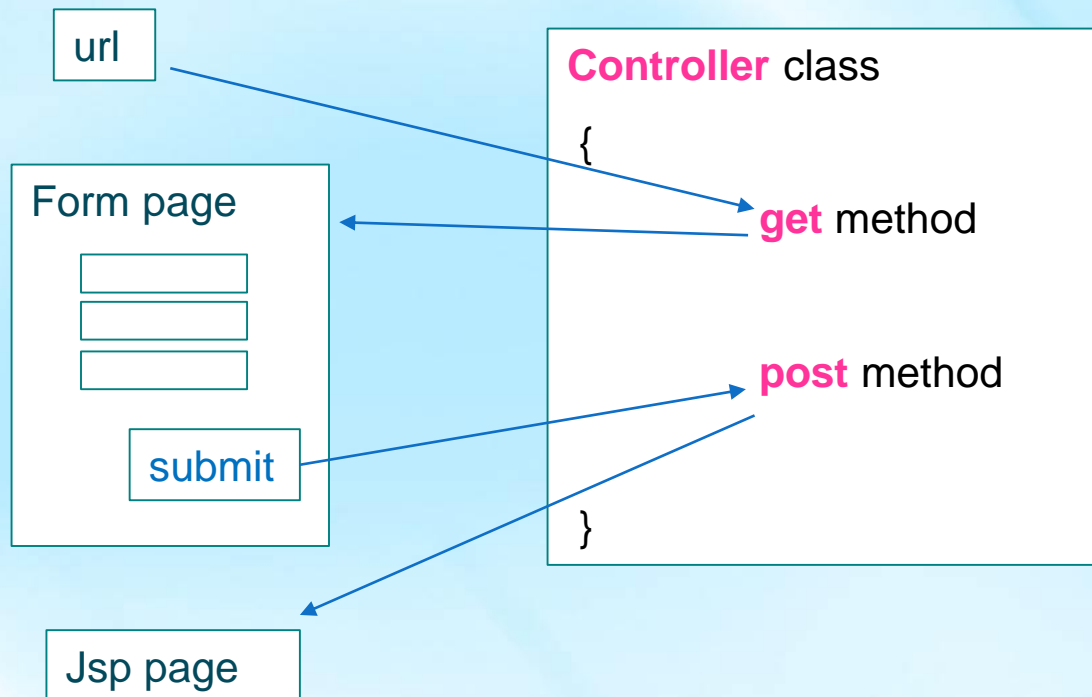
Hello \${nameToDisplay}



ID of the attribute in the model dictionary



Form Controller



Form Controller

- ▶ Form
 - Fields
 - Submit button
- ▶ Submit button \Rightarrow calls the post method of the controller class
- ▶ E.g,

Name	<input type="text"/>	Zip Code	<input type="text"/>	<input type="submit" value="Submit"/>
------	----------------------	----------	----------------------	---------------------------------------

Form Controller – *Model Class*

- ▶ Create a model class corresponding to the form
 - Each form field = a property variable (instance variable) in the class
 - Gettors/settors
- ▶ E.g,

```
public class UserForm {  
  
    private String name;  
    private Integer zipCode;  
  
    //gettors/settors  
}
```

Form Controller – *Controller Class*

- ▶ Create a controller class
- ▶ Create an instance of the form model class
 - In the get method
 - Add it to the Model dictionary

Form Controller – Controller Class

- ▶ Create a post method
 - **@RequestMapping (method = RequestMethod.POST)**
 - Pass this form model object as input for **Data Binding**
 - Use **@ModelAttribute** annotation on this input
 - Returns
 - Either a jsp page
 - Or redirection (see further)

Form Controller – Controller Class

► E.g,

```
@Controller
@RequestMapping(value="/form")
public class UserFormController {

    @RequestMapping(method=RequestMethod.GET)
    public String home(Model model) {
        model.addAttribute("userForm", new UserForm());
        return "userFormPage";
    }

    @RequestMapping(value="/userInscription", method=RequestMethod.POST)
    public String getFormdata(Model model, @ModelAttribute(value="userForm") UserForm inscriptionForm) {
        String welcomeMessage = "Welcome, " + inscriptionForm.getName() + " !";
        model.addAttribute("message", welcomeMessage);
        return "userFormPage";
    }
}
```

→ Add a form model object to the model dictionary

↓
Property values of inscriptionForm can be automatically filled with values entered by the user through the form thanks to Data Binding

Form Controller – Jsp Page

- ▶ Create a form jsp page
- ▶ Tag library

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

- ▶ Tag : **form**
 - **method**
 - Post
 - **action**
 - Path to call the post method (cf value of the **@RequestMapping**)
 - **modelAttribute**
 - Id of the model object corresponding to the form in the Model dictionary

Form Controller – Jsp Page

► Data binding

- To automatically link value from fields (of jsp form) to properties of the model
- In field tag (<form:label>, <form:input> ...)
 - **path** : corresponding property name (instance variable) of the model class

Form Controller – Jsp Page

► E.g,

```
<form:form id="inscription"
  method="POST"
  action="/first/form/userInscription"
  modelAttribute="userForm">
  <form:label path="name"> Name </form:label>
  <form:input path="name"/>

  <form:label path="zipCode"> Zip Code </form:label>
  <form:input path="zipCode"/>

  <form:button>Submit</form:button>

  <c:if test="${not empty message}"> ${message} </c:if>
</form:form>
```

If "/first" is the root of the project

Cf **name** property of UserForm

Cf **zipCode** property of UserForm

⇒ Data Binding :

The *name* and *zipCode* properties of the *inscriptionForm* object model are automatically filled by values entered by the user

Form Controller

- ▶ If only one get/post method in a controller class
 - ⇒ In Controller
 - No need of value attribute for `@RequestMapping` in post/get method
 - ⇒ In jsp page
 - No need of action attribute in form tag

Redirection

- ▶ Output of the get/post method
 - Either a jsp pag
 - Or redirection
- ▶ Redirection
 - To an url
 - ⇒ calls the get method of a **controller**
- ▶ Syntax
 - **redirect** : path

Redirection

► E.g,

```
@Controller
@RequestMapping(value="/login")
public class LoginController {

    @RequestMapping(method=RequestMethod.GET)
    public String home(Model model) {

        ...
        return "loginPage";
    }

    @RequestMapping(method=RequestMethod.POST)
    public String verifyLogin(Model model , ...)
    { ...
        if (...) // if login OK
        { ...
            return "redirect:/index";
        }
        else return "LoginPage";
    }
}
```

Jsp page ←

Call of a controller ←

Jsp page ←

Unified Expression Language

- ▶ Unified Expression Language (UEL)
 - Syntax to interact with beans
- ▶ In Web pages :
 - **`${scope.expression}`**
 - Where scope is optional

Unified Expression Language

► Scopes in UEL

- **applicationScope**
 - A Map of the application scope attribute values, keyed by attribute name
- **requestScope**
 - A Map of the request attributes for this request, keyed by attribute name
- **sessionScope**
 - A Map of the session attributes for this request, keyed by attribute name
- **view**
 - The root UIComponent in the current component tree for this request
- **cookie**
 - A Map of the cookie values for the current request, keyed by cookie name
- **facesContext**
 - The FacesContext instance for the current request

Unified Expression Language

- ▶ **Scopes in UEL (continue)**
 - **Header**
 - A Map of HTTP header values for the current request, keyed by header name
 - **headerValues**
 - A Map of String arrays containing all the header values for HTTP headers in the current request, keyed by header name
 - **initParam**
 - A Map of the context initialization parameters for this web application
 - **Param**
 - A Map of the request parameters for this request, keyed by parameter name
 - **paramValues**
 - A Map of String arrays containing all the parameter values for request parameters in the current request, keyed by parameter name

Unified Expression Language

- ▶ Use UEL in web pages to access data passed from controller
 - Data passed through the model dictionary
- ▶ Access to an object from the model dictionary thanks to its key

`${objectModelKey}`

- ▶ E.g,
 - Controller :
String message = "Hello";
model.addAttribute("messageID",message);
 - Jsp page :

`${messageID}`

Unified Expression Language

- To access a property of an object from the model dictionary

`${objectModelKey.property}`

- If property is **private** and **public getter** exists in the class of the object
- E.g,

Model class :

ModelClass
private String name public getName()

Controller: `model.addAttribute("modelObjectID", new ModelClass())`

Jsp page : **`${modelObjectID.name}`**

Unified Expression Language

- ▶ E.g,
 - If *ModelClass* has a private property called *user* (with public getter) which is a reference to an object of the *Person* class
 - and if *Person* class has a private property called *name* (with public getter)



⇒ In jsp : `${modelObjectID.user.name}`

Unified Expression Language

- ▶ To link Web page components to a collection (list, array,...)

`${ ...list[i] }`

- E.g,
 - If *ModelClass* has a private property called *persons* (with public getter) which is an ArrayList of objects from the *Person* class



⇒ In jsp : **`${modelObjectID.persons[2].name}`**

Unified Expression Language

- ▶ To link Web page components to a hash map

`${ ...hashMap['idValue'] }`

- E.g,

- If *ModelClass* has a private property called *personsMap* (with public getter) which is an *HashMap* of objects from the *Person* class

⇒ In jsp : `${modelObjectID. personsMap['id234'].name}`

Unified Expression Language

Other UEL operators

- ▶ Arithmetic
 - +, -, *, /, div, %, mod
- ▶ Logical
 - and, &&, or, ||, not, !
- ▶ Relational
 - ==, eq, !=, ne, <, lt, >, gt, <=, ge, >=, le
- ▶ Empty
 - To determine whether a value is null or empty
- ▶ Conditional
 - A ? B : C.
 - Evaluate B or C, depending on the result of the evaluation of A

Unified Expression Language

- ▶ The precedence of operators
 - []
 - () (used to change the precedence of operators)
 - - (unary) not ! Empty
 - * / div % mod
 - + - (binary)
 - < > <= >= lt gt le ge
 - == != eq ne
 - && and
 - || or
 - ? :

JSTL

- ▶ JavaServer Pages Standard Tag Library (JSTL)
 - Collection of tags to encapsulate core functionality
 - Support for common, structural tasks
 - such as iteration, conditionals, ...
- ▶ To include JSTL Core library
 - Usually prefix="c"
 - uri="http://java.sun.com/jsp/jstl/core"

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

JSTL

if

```
<c:if test = " ... " >
```

```
...
```

```
</c:if>
```

- ▶ *N.B. No else !*
- ▶ E.g,

```
<c:if test="{not empty message}"> ${message} </c:if>
```

```
<c:if test="{user.name=='guest'}"> Welcome, guest! </c:if>
```

JSTL

choose

```
<c:choose>
  <c:when test = " ... " >
    ...
  </c:when>
  <c:when test = " ... " >
    ...
  </c:when>
  ...
  <c:otherwise >
    ...
  </ c:otherwise >
</c:choose>
```


JSTL

forEach

Objects Collection
↑
`<c:forEach items = "Y" var = "X" >`
... `${X}` ...
`</c:forEach >`

E.g, on an **ArrayList**

ArrayList<Category>
↑
`<c:forEach items="${allCategories}" var="category">`
`${category.name}`
`</c:forEach>`

Category

```
private String name  
public getName()
```

JSTL

forEach

E.g, on a **hashmap**

```
<c:forEach items="${school.students}" var="item">  
    ${item.value.name}  
</c:forEach>
```

Key Value
↑ ↑
HashMap<Integer, Student>

Student

```
private String name  
public getName()
```

JSTL

set

```
<c:set
```

```
  var = "..."
```



Name of the variable

```
  scope= "..."
```



Scope of the variable

```
  value = "..."
```



Value of the variable

```
>
```

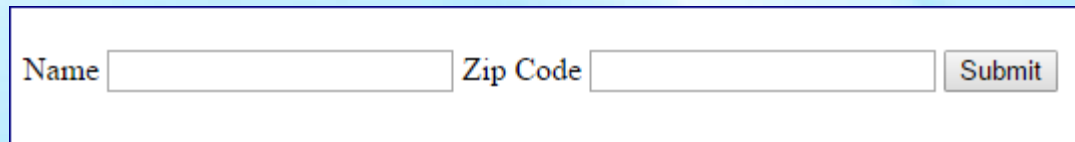
► Scope

- page
- request
- session
- application

Form Validation

- ▶ To validate data entered by the user in the form

- ▶ E.g,



The image shows a simple web form with a light yellow background. It contains two text input fields. The first field is preceded by the label 'Name' in a brown font. The second field is preceded by the label 'Zip Code' in a brown font. To the right of the second input field is a grey button with the word 'Submit' in black text.

- *Name*
 - *From 3 to 30 characters*
- *Zip code*
 - *Not empty number*
 - *Value between 1000 and 9999*

Form Validation – *pom.xml*

- ▶ *Add a dependency*

```
<dependency>  
    <groupId>org.hibernate</groupId>  
    <artifactId>hibernate-validator</artifactId>  
</dependency>
```

Form Validation – *Model Class*

- ▶ Through annotations in the Model class
- ▶ Import
 - `import javax.validation.constraints.*;`
- ▶ Annotations to properties (instance variables)

<http://docs.oracle.com/javaee/6/api/javax/validation/constraints/package-summary.html>

- **@NotNull** : mandatory property
- **@Size** : number of characters (for **strings**)
 - **min** attribute / **max** attribute
- **@Min** / **@Max** : minimum / maximum value allowed for **numbers**
 - **value** attribute

Form Validation – *Model Class*

► E.g,

```
public class UserForm {  
  
    @Size(min=3, max=30)  
    private String name;  
  
    @NotNull  
    @Min(value=1000)  
    @Max(value=9999)  
    private Integer zipCode;  
  
    //getters/settors  
}
```

Form Validation – Controller Class

► Import

- *import javax.validation.Valid;*
- *import org.springframework.validation.BindingResult;*

► Post method

- Inputs :
 - Use **@Valid** annotation on form Model object
 - Add **BindingResult** as input

Form Validation – Controller Class

► E.g,

```
@Controller
@RequestMapping(value="/form")
public class UserFormController {

    ... // get method

    @RequestMapping(value="/userInscription", method=RequestMethod.POST)
    public String getFormData(Model model,
                               @Valid @ModelAttribute(value="userForm") UserForm inscriptionForm,
                               final BindingResult errors) {
        String welcomeMessage;
        if (!errors.hasErrors())
            welcomeMessage = "Welcome, " + inscriptionForm.getName() + " !";
        else welcomeMessage = "Sorry, the form is not valid!";
        model.addAttribute("message", welcomeMessage);
        return "userFormPage";
    }
}
```

Form Validation – *Jsp Page*

- ▶ Specify where to display the error messages
 - Tag
 - **<Form:error >**
 - Attribute
 - **path** : id of the corresponding field

Form Validation – Jsp Page

► E.g,

```
<form:form id="inscription" method="POST"
  action="/first/form/userInscription"
  modelAttribute="userForm">
  <form:label path="name"> Name </form:label>
  <form:input path="name"/>
  <form:errors path="name"/>
  <br>
  <form:label path="zipCode"> Zip Code </form:label>
  <form:input path="zipCode"/>
  <form:errors path="zipCode"/>
  <br>
  <form:button>Submit</form:button>
  <br>
  <c:if test="${not empty message}"> ${message} </c:if>
</form:form>
```

Error messages to display