

1 - reprise des cours du jours precedent

2 - correction du tp XHTML

3 - cours XHTML

-> Balises input

```
<input type="text" value="#{variable}" />
<input type="password" value="#{variable}" />
<input type="email" value="#{variable}" />
<input type="number" value="#{variable}" />
<input type="date" value="#{variable}" />
d<input type="time" value="#{variable}" />
<input type="color" value="#{variable}" />
<input type="checkbox" value="#{variable}" />
```

-> Les beans: variables multiples bordéliques remplacées par les beans. POJO plain old java object; classe qui correspond a des standards, doit avoir des getters et seters, un constructeur

LES BEANS

Les beans sont :

☕ des pojos

Ils doivent avoir :

- ☕ de getters et de setters qui respectent une convention de nommage particulière pour les attributs
- ☕ un constructeur par défaut sans arguments
- ☕ etre serializable
- ☕ etre annoté avec @ManagedBean



```
1 package entity;
2
3 import javax.faces.bean.ManagedBean;
4 import javax.faces.bean.SessionScoped;
5
6 @SessionScoped
7 @ManagedBean(name="bean")
8 public class Bean {
9     private String nom;
10    private String prenom;
11    private String sexe;
12
13    public String getNom() {
14        return nom;
15    }
16
17    public void setNom(String nom) {
18        this.nom = nom;
19    }
20
21    public String getPrenom() {
22        return prenom;
23    }
24
25    public void setPrenom(String prenom) {
26        this.prenom = prenom;
27    }
28
29    public String getSexe() {
30        return sexe;
31    }
32
33    public void setSexe(String sexe) {
34        this.sexu = sexe;
35    }
36
37    public String valider() {
38        return "page2";
39    }
40 }
41
```

LES MANAGED BEANS

Les managed beans sont des beans gérés par le conteneur d'application(jsf).

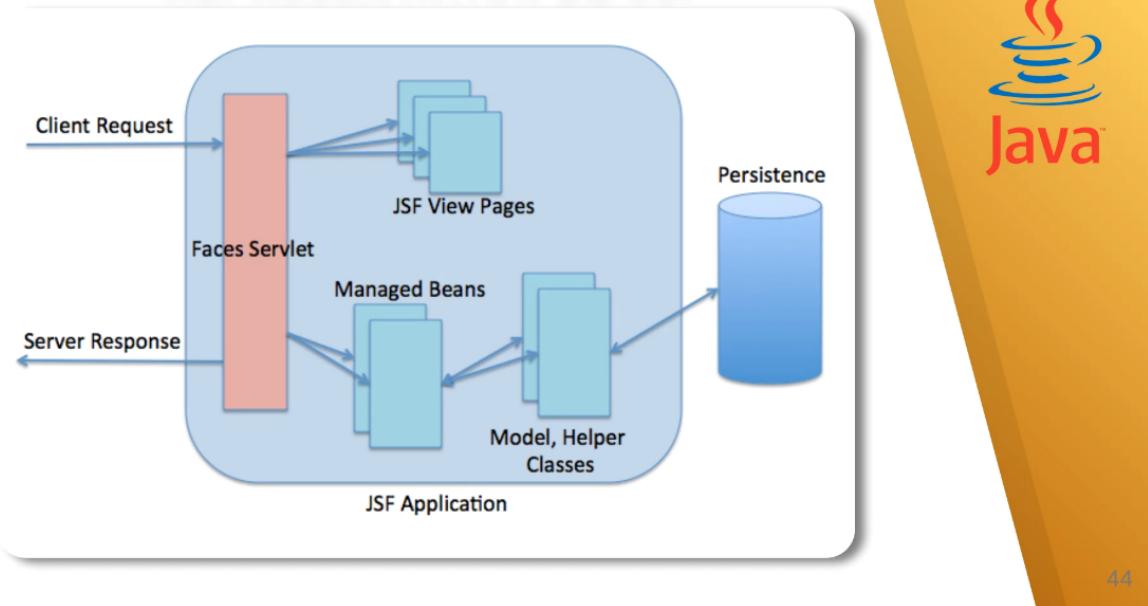
Ils sont créés et détruits par le conteneur d'application.

Dans le fichier web.xml, nous devons déclarer le bean.

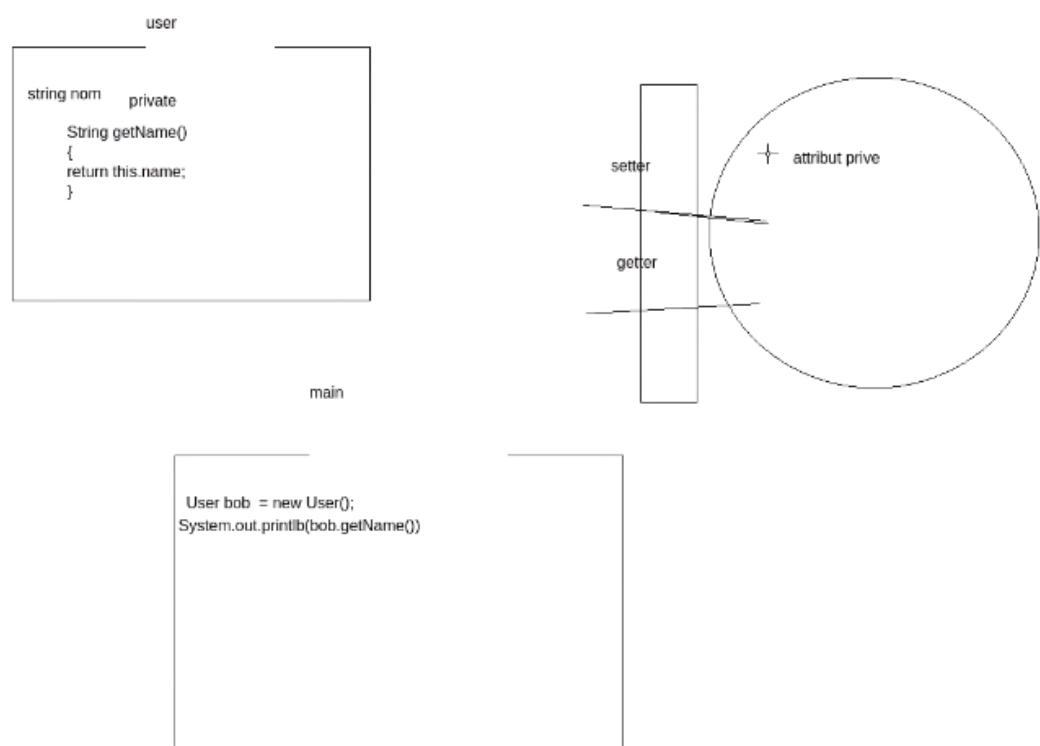
```
<managed-bean>
<managed-bean-name>bean</managed-bean-name> // nom du bean
<managed-bean-class>com.bean.Bean</managed-bean-class> // le chemin de la classe
<managed-bean-scope>session</managed-bean-scope> // session, request, application
</managed-bean>
```



REPRESENTATION DE JSF



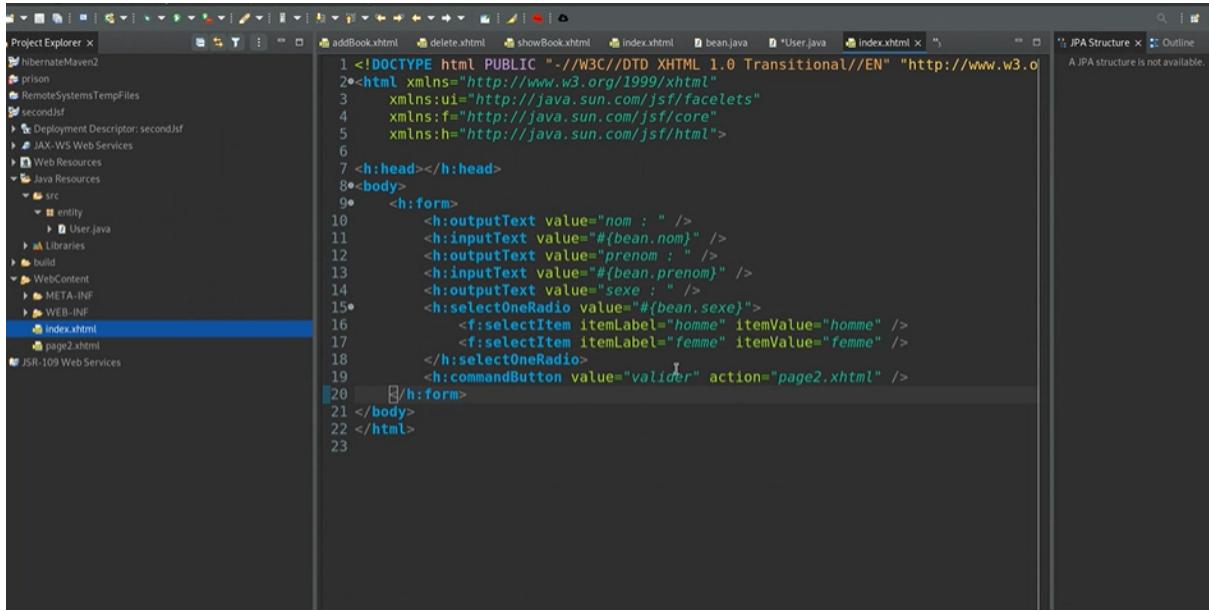
44



LES CYCLES DE VIE DES BEANS

Le cycle de vie d'un bean est composé de 6 étapes :

- ⌚ restore view / reconstruct component tree : cette étape permet de restaurer la vue et de reconstruire l'arbre des composants.
- ⌚ apply request values : cette étape permet de récupérer les valeurs des composants.
- ⌚ process validations : cette étape permet de valider les composants.
- ⌚ update model values : cette étape permet de mettre à jour les valeurs des composants.
- ⌚ invoke application : cette étape permet d'appeler l'application.
- ⌚ render response : cette étape permet de rendre la réponse.



The screenshot shows an IDE interface with several windows:

- Project Explorer:** Shows the project structure with packages like hibernateMaven2, RemoteSystemsTempFiles, secondjsf, Deployment Descriptor: secondjsf, JAX-WS Web Services, Web Resources, Java Resources, and src containing entity and User.java.
- Code Editor:** Displays an XHTML file named index.xhtml with the following code:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/1999/xhtml">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">
<head></head>
<body>
    <h:form>
        <h:outputText value="nom : " />
        <h:inputText value="#{bean.nom}" />
        <h:outputText value="prenom : " />
        <h:inputText value="#{bean.prenom}" />
        <h:outputText value="sexe : " />
        <h:selectOneRadio value="#{bean.sexe}" />
            <f:selectItem itemLabel="homme" itemValue="homme" />
            <f:selectItem itemLabel="femme" itemValue="femme" />
        </h:selectOneRadio>
        <h:commandButton value="valider" action="page2.xhtml" />
    </h:form>
</body>
</html>
```
- JPA Structure:** A panel indicating "A JPA structure is not available."

LES EXPRESSIONS DE LIAISON DE BEANS



Les expressions de liaison peuvent être utilisées dans les balises jsf pour :

- ─ afficher des valeurs
- ─ récupérer des valeurs
- ─ appeler des méthodes

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/1999/xhtml">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">

    <h:head></h:head>
    <body>
        <h:form>
            <h:outputText value="nom : " />#{[]}
            <h:inputText value="#{bean.nom}" />
            <h:outputText value="prenom : " />
            <h:inputText value="#{bean.prenom}" />
            <h:outputText value="sexe : " />
            <h:selectOneRadio value="#{bean.sexe}">
                <f:selectItem itemLabel="homme" itemValue="homme" />
                <f:selectItem itemLabel="femme" itemValue="femme" />
            </h:selectOneRadio>
            <h:commandButton value="valider" action="page2.xhtml" />
        </h:form>
    </body>
</html>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<faces-config xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/
version="2.2">

    <managed-bean>
        <managed-bean-name>bean</managed-bean-name>
        <managed-bean-class>entity.Bean</managed-bean-class>
        <managed-bean-scope>session</managed-bean-scope>
    </managed-bean>

</faces-config>
```

4 - Elements de bases de la page XHTML

LES COMPOSANTS DE BASE

Les composants de base sont des composants qui permettent de créer des formulaires.

- ⌚ **inputText** : permet de créer un champ de texte.
- ⌚ **inputSecret** : permet de créer un champ de mot de passe.
- ⌚ **inputHidden** : permet de créer un champ caché.
- ⌚ **inputTextarea** : permet de créer un champ de texte multiligne.
- ⌚ **inputFile** : permet de créer un champ de fichier.
- ⌚ **inputCheckbox** : permet de créer une case à cocher.
- ⌚ **inputRadio** : permet de créer un bouton radio.



LES CHECKBOX



Les checkbox permettent de créer des cases à cocher.

```
<h:selectBooleanCheckbox value="#{bean.bac}" />
```

```
private boolean bac;
```

LES BOUTONS RADIO



Les boutons radio permettent de créer des boutons radio.

```
<h:selectOneRadio value="#{bean.sex}">
    <f:selectItem itemLabel="homme" itemValue="homme" />
    <f:selectItem itemLabel="femme" itemValue="femme" />
</h:selectOneRadio>
```

```
private String sexe;
```

LES CONDITIONS EN JSF

Les conditions permettent d'afficher ou non des composants.

- ⌚ **h:outputText** : permet d'afficher du texte.
 - ⌚ value : permet de définir le texte à afficher.
 - ⌚ rendered : permet de définir si le composant doit être affiché ou non.

exemple :

```
<h:outputText value="Bonjour #{bean.nom}" rendered="#{bean.bac}" />
```



59

6 - Les formulaires

LES FORMULAIRES

Jsf permet la création de formulaires, nous pouvons valider les données saisies par l'utilisateur.

Le bouton submit permet de valider les données saisies par l'utilisateur.

Le bouton reset permet de réinitialiser les données saisies par l'utilisateur.

```
<h:form>
    <h:inputText value="#{bean.attribut}" />
    <h:commandButton value="valider" action="#{bean.methode()}" />
</h:form>
```



60

TP : FORMULAIRE

Vous devez créer un formulaire qui permet de saisir les informations suivantes :

- nom
- prénom
- sexe

Dans la deuxième page vous afficherez : "Bonjour ", "monsieur" ou "madame", le nom et le prénom de l'utilisateur.

