JEE : JavaBean (bean)

Mohammed OUANAN

m.ouanan@umi.ac.ma

Plan

Introduction

Définition

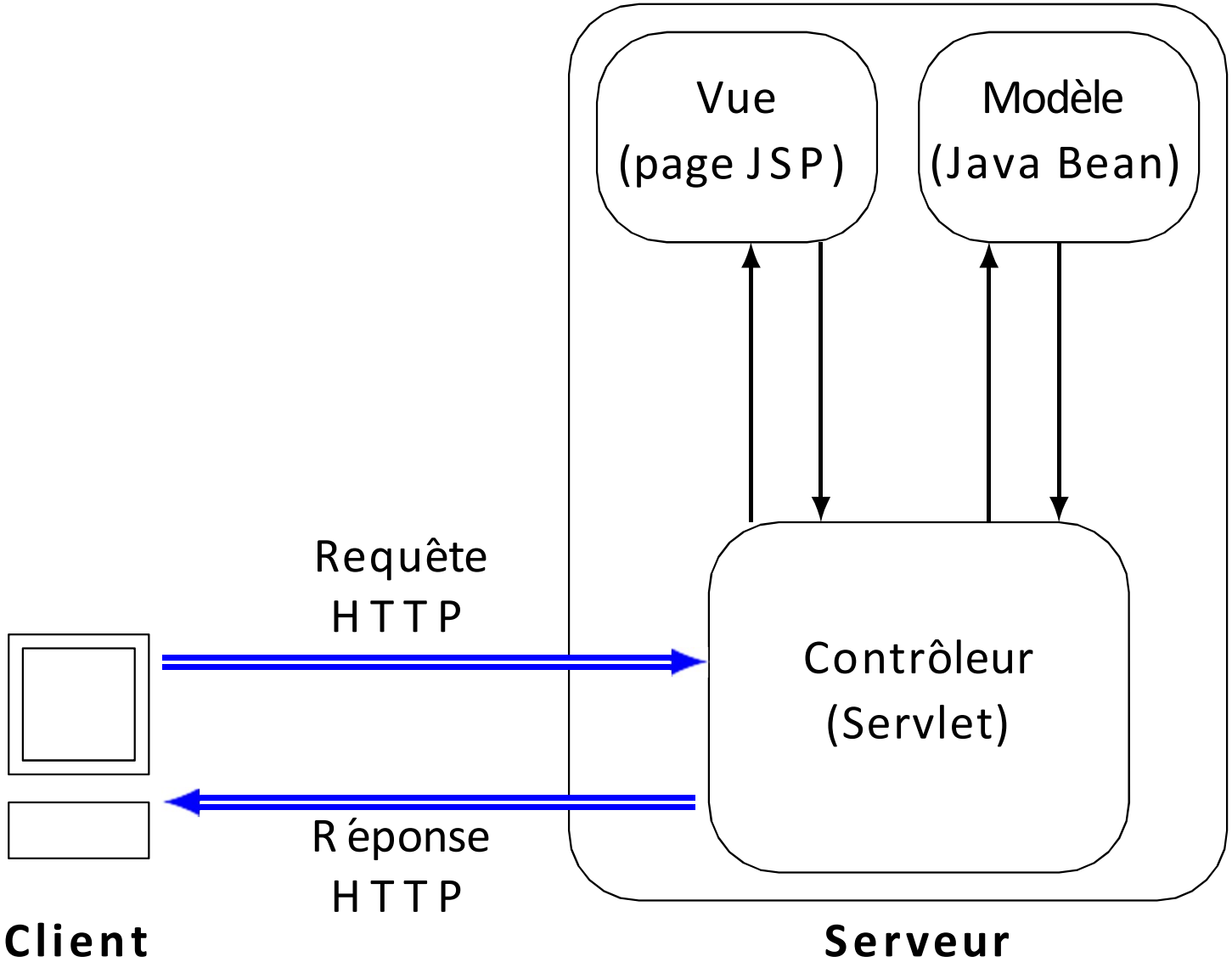
- `JavaBean` (on dit aussi `bean`)
- Une simple classe Java qui respecte les règles/conventions sur
  - la construction
  - le nommage
  - les attributs
  - les méthodes
  - ...

Introduction

Les conventions

- Le `bean` doit être public et non final
- Il doit avoir un constructeur public et sans paramètre
- Il ne doit pas avoir d'attributs publics (appelés ici champ)
- Il doit avoir un getter/setter public pour chaque attributs non public (appelés ici propriétés)
- La première lettre du nom de l'attribut dans les getters/setters doit être capitalisée
- Il peut implémenter l'interface `Serializable`
- ...

Un bean : le modèle d'une application JEE



Création d'un bean

Déroulement

- Faire un clic droit sur `src` de notre projet
- Aller dans `New` et choisir `Class`
- Remplir le champ `File name:` par Personne (par example)
- Modifier le champ `Package:` et écrire `org.eclipse.model` (pour séparer les deux types de classes Java : le contrôleur (Servlet) et Modèle (bean)
- Valider

Création d'un bean

```
package org.eclipse.model;
public class Personne { //classe public et non final
    private int num;
    private String nom;
    private String prenom; // tous les champs sont private
    public int getNum() {
        return num;
    }
    public void setNum(int num) {
        this.num = num;
    }
    public String getNom() {
        return nom;
    }// des getters/setters public pour tous les champs
    public void setNom(String nom) {
        this.nom = nom;
    }
    public String getPrenom() {
        return prenom;
    }
    public void setPrenom(String prenom) {
        this.prenom = prenom;
    }
}
```

Création d'un bean

```
// on peut aussi redefinir la methode toString

@Override
public String toString() {
    return "Personne [num= " + num + ", nom= " + nom
        + ", prenom= " + prenom +  "]";
}

// Et evidemment d'autres methodes
```

Utilisation d'un bean dans le contrôleur

Modifions la méthode `doGet()` de notre Servlet

```
protected void doGet(HttpServletRequest request,
   HttpServletResponse response) throws
   ServletException, IOException {
     Personne perso = new Personne();
     perso.setNom("Wick");
     perso.setPrenom("John");
     perso.setNum(100);
     PrintWriter out = response.getWriter();
     out.print(perso);
}
```

Utilisation d'un bean dans le contrôleur

Modifions la méthode `doGet()` de notre Servlet

```
protected void doGet(HttpServletRequest request,
   HttpServletResponse response) throws
   ServletException, IOException {
     Personne perso = new Personne();
     perso.setNom("Wick");
     perso.setPrenom("John");
     perso.setNum(100);
     PrintWriter out = response.getWriter();
     out.print(perso);
}
```

Pour tester

Aller à l'adresse `http://localhost:8080/nomProjet/mapage`

Comment et pourquoi sérialiser ? (créer un `Java Project` et déplacer le package `org.eclipse.model`)

```
public class Personne implements Serializable{
    private int num;
    private String nom;
    private String prenom;
    public int getNum() {
        return num;
    }
    public void setNum(int num) {
        this.num = num;
    }
    public String getNom() {
        return nom;
    }
    public void setNom(String nom) {
        this.nom = nom;
    }
    public String getPrenom() {
        return prenom;
    }
    public void setPrenom(String prenom) {
        this.prenom = prenom;
    }
}
```

Considérons la méthode `main` suivante :

```
private static final String FILE_NAME = "personne.txt";
public static void main(String[] args) {
  Personne p = new Personne();
  p.setNom("Wick");
  p.setPrenom("John");
  p.setNum(100);
  try {
    FileOutputStream fs = new FileOutputStream(FILE_NAME);
    ObjectOutputStream os = new ObjectOutputStream(fs);
    os.writeObject(p);
    os.close();
  } catch (Exception e) {
    e.printStackTrace();
  }
  try {
    FileInputStream fis = new FileInputStream(FILE_NAME);
    ObjectInputStream ois = new ObjectInputStream(fis);
    Personne p2 = (Personne) ois.readObject();
    System.out.println(p2.getNom() + " " + p2.getPrenom() + " " + p2.
       getNum());
    ois.close();
  } catch (Exception e) {
    e.printStackTrace();
  }
```

Implémenter l'interface

Remarques

- La méthode `writeObject()` sérialise l'objet ensuite l'écrit dans le fichier
- La méthode `readObject()` lit l'objet ensuite le désérialise
- Si la classe `Personne` n'implémente pas l'interface `Serializable`, il est impossible d'utiliser les deux méthodes précédentes

Imple´menter l'interface

Remaques

- La méthode `writeObject()` sérialise l'objet ensuite l'écrit dans le fichier
- La méthode `readObject()` lit l'objet ensuite le desérialise
- Si la classe `Personne` n'implémente pas l'interface `Serializable`, il est impossible d'utiliser les deux me´thodes précédentes

Tester le `main` et aller vérifier le contenu du fichier `personne.txt` (Rafraichir le projet pour le trouver)