NOM Bäumer

Prénom Yannick

Promo M1

Date 09.01.14

16,25

BAEUMER Yannick
IECHANGE - 2014

MATIÈRE JEE

① Servlet:

Une Classe ~~de~~ Java, qui ~~regit~~ pour gérer des requests Web, ~~par exemple~~. Il existe des spécialisation par exemple pour le protocol HTTP avec les Methods GET et POST. Un Servlet vie dans le container Web et est géré par ce container (cycle de vie etc..)

JSP:

Une JSP (signification?) est une page dans quelle ~~on~~ on peut utiliser le HTML et du code java (encadré dans des balises ~~type~~ <% %>). Il y a aussi ~~des autre~~ la ~~langue python~~ Expression language et des different Librairies des balises (taglibs). Le Jsp est normalement utilisé pour le View de concept MVC et va être compilé dans un Servlet est géré dans le container Web.

EJB:

Les Enterprise Java Bean sont des Classes et objets qui répresent la Business Logic. Il y a des Beans Session, Message et Entity (pour les cas different). Les EJB ont un interface distinct.

JPA:

Le JPA (signification?) est une technique (API) pour généraliser l'interface entre application et ~~data~~ base des données. Il gère aussi le ORM (Object-Relation Mapping)

② JSTL : Java Standard Tag Library
est utilisée pour ajouter le contrôle de fait dans une JSP.
EL : Expression Language
est une langage pour simplifier l'interaction avec des Bean enregistré avec le context. Grace a l'EL c'est moins necessaire d'utiliser du code Java native (<% %>) dans les JSPs. (2)

③ Context, Session, Page, ~~Request~~ : (1)

④ Un Java Bean est un ~~contact~~ Class container pour echanger des données. Chaque Java Bean a un constructeur de fault, des propriétés privés et de getter et setter public:

```
class car {
    public car()
    {}
    private int regNumber;
    public void setRegNumber(int regNumber)
    { this.regNumber = regNumber; }
    public int getRegNumber()
    { return this.regNumber; }
}
```

(2)

⑤
- Expression Language: un objet du bean Membre est registré comme attribut "memOb" dans la page. Dans le JSP on peut faire:
  ${memOb.nom}

(1)

- <%- Jsp:useBean class="Membre" id="memOb" %>
  <%- Jsp:getValue id="memOb" property="nom" %>
- <%- page.getAttribute("memOb").getNom() %>

⑥ request.getAttribute("xXx");
et
request.SetAttribute("xXx");
Request est un objet implicit, disponible dans les methode do... et handle... et xXx est le nom (name="xXx") du champ.

⑦ On fait une heritage du class HTTPServlet, parce qu'on utilise normalement le protocol HTTP.

(1) On implement les methods doGet et doPost, parce qu'on peut gérer le request mieux qu'on utilise la methode doRequest.

⑧ (1) HTTPServletRequest request et HTTPServletResponse response.

⑨ Le Session Bean, Stateless ou Statefull:
pour implementer la Businesslogic qui a besoin d'un session, ça veut dir plusieur requests successifves qui sont invoquée par un usager. Par exemple des bien qui peuvent être ajoutée à un panier.

(2)

Le Message bean: Stateless
pour implementer une Message driven Businesslogic. Ils cachent les exeptions contre le l'usager.?

Entity Bean: (deprecated)

Représente des entités dans la Business Logic des EJB sont ~~automatiquement~~ persistée dans une base de données.

(10) Je pense que une 'Classe Entity' est créé comme une Java Bean (constructeur, propriétées privé, getter/Setter) avec des annotations pour le model de donnée. des Entity peuvent être persisté directement. Si on ~~est~~ a besoin de persistence, c'est une bonne idée, si pas, c'est peut-être trop pour un Bean très simple. Une entity, par exemple ~~est~~ a obligatoirement besoin d'un @Id.

(11) Il manque l'annotation @Id

(2) Dans une base de donnée chaque entité ~~est~~ a besoin d'un clé primaire, et le ORM du JPA ne sais pas quel propriété peut-être utilisé comme clé.

(12) Il manque une isbn, parce que dans la definition du Entity il y a l'annotation @Column(nullable = false). ~~Ce que ça~~ ~~extra~~

(2) Il faut qu'on ajoute : monLivre.setIsbn("666-888-333-21"); avant de appeller em.persist(monLivre).