

JEE

Hibernate:

65%

```

1) <class name = "Team" table = "team">
    <id name = "idTeam" column = "IDTEAM">
        <generator class = "increment" />
    </id>
    <set name = "players" table = "player" inverse = "true"
        cascade = "all" lazy = "false">
        <key column = "IDTEAM" not-null = "true" />
        <one-to-many class = "Player" />
    </set>
    <one-to-one name = "Game" constrained = "true" />
</class>

```

manage coach!! is it

```

<class name="Player" table="player">
  <id name="idPlayer" column="IDPLAYER">
    <generator class="increment"/>
  </id>
  <many-to-one name="team" column="IDTEAM"
    class="Team"/>
</class>

```

```

<class name="Coach"
  <id name="idTeam" column="IDTEAM">
    <generator class="foreign">
      <param name="property">team</param>
    </generator>
  </id>
  <one-to-one name="team" constrained="true"/>
</class>

```

(- 21) *pro*

19/11 2) Si on ne définit pas des setter/getter et des constructeurs, l'hibernate ne peut pas accéder aux classes. On doit créer un constructeur sans paramètre pour les objets, car quand on utilise «get» dans la BDD, pour retourner, elle crée une nouvelle classe. Donc on est obligé d'avoir des constructeurs.

06/11 3)

Spring

82%

- 1) Avec Spring, nous avons 2 manières d'injecter la dépendance
- injection en utilisant setters
 - injection en utilisant le constructeur.

Nous pouvons instancier un objet en appelant une fonction extérieure
et ??

100%

- 2) Spring est un conteneur léger, car il n'a pas besoin de serveur, il est exécutable avec une simple machine virtuelle Java. La configuration est faite via un fichier XML, il gère les cycles de vie des composants, et il gère la dépendance entre les composants.

Si on utilise le EJB, on est obligé de les exécuter sur un serveur, on ne peut pas faire de test s'il n'y a pas de serveur. Même pour tester une simple classe, on doit lancer tout le serveur, donc ceci est un conteneur lourd.

100%

- 3) Il est possible de créer ^{dynamiquement} le fichier XML ou de le modifier durant l'exécution.

En utilisant Post processing, Bean Factory Post Processor permet de modifier la Bean Factory ^{configuration de} après sa création.

100%

- 4) Tout d'abord, Spring est un conteneur léger, et EJB un conteneur lourd. EJB est très coûteux en période d'essai (de test).

Ensuite, avec Spring, il est difficile d'atteindre les "Scalability" (load balancing, replication, clustering).

Soit on utilise des frameworks (hibernate et Spring), soit on utilise JSP/servlet et EJB.

S'il y a beaucoup de connexions simultanées et des transactions, alors il faut utiliser EJB, et dans le cas contraire, il faut utiliser des framework.

EJB

(10/1)

A	B	sens	multiplicité
etudiant	~~etudiant~~ stage	unidirectionnelle	many-to-many

2) @Entity

```
public class Etudiant {
```

```
    private Collection<stage> stages;
```

~~@Many-to-Many~~

```
    public Collection<stage> getStages()
```

```
    { return stages; }
```

```
    public void setStage(Collection<stage> stages) {
```

```
        this.stages = stages;
```

```
    }
```

mappe (et)
mappe l'assé

3) voir la page (copie) suivante.