Review of the First Test

Olivier Liechti
AMT

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

Expliquez de manière très claire et précise quelle est la relation entre Glassfish et Java EE (en définissant les deux très précisément).

- The software that implements the Java EE specification is called an "**application server**"
  - There are **open source** and **proprietary** application servers.
  - Glassfish, JBoss, WebSphere, BEA WebLogic are examples of application servers.
  - Editors compete on aspects that are not defined the specification (clustering, administration, etc.).

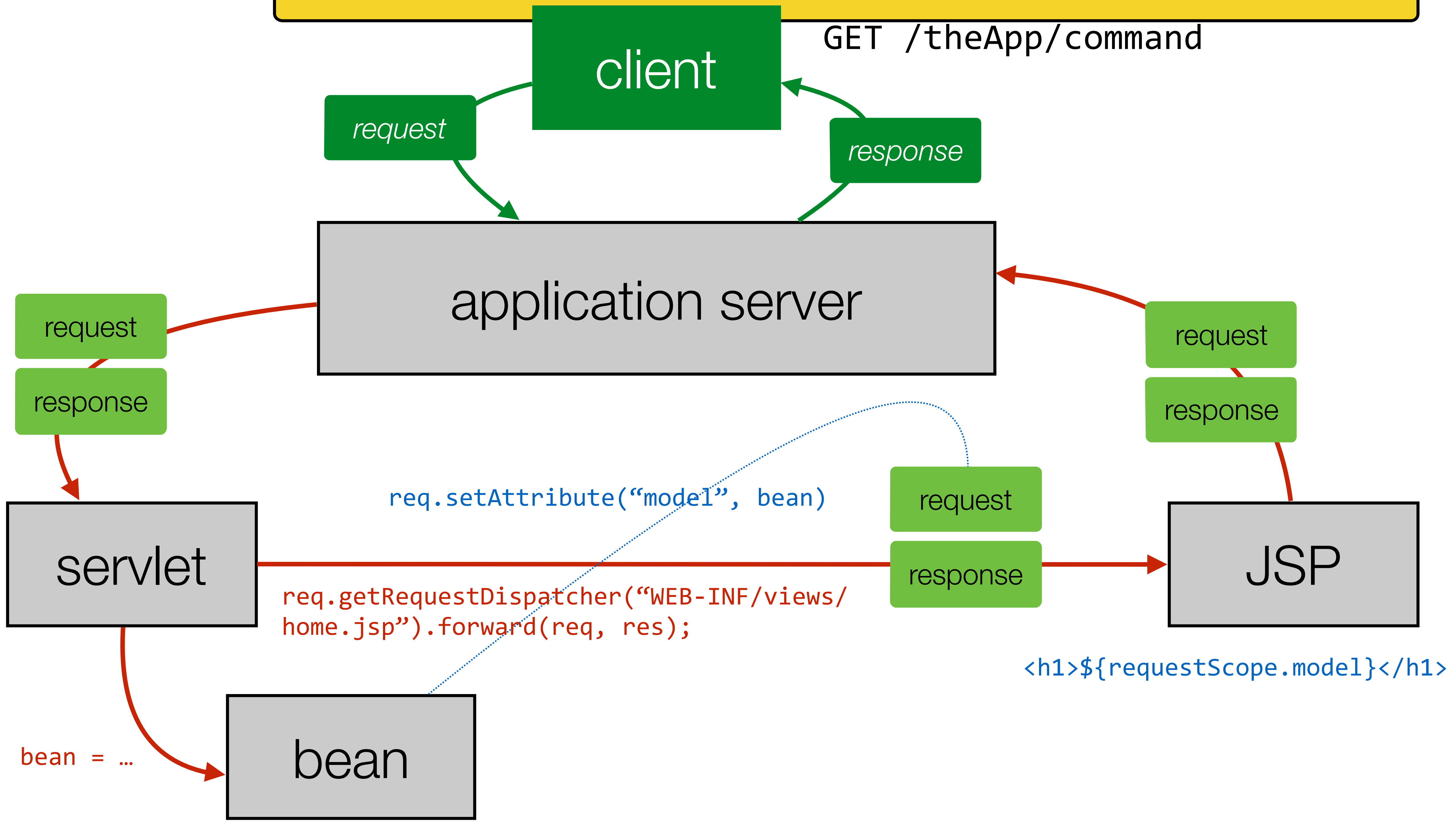
En utilisant des exemples basés sur 2 APIs différentes de la plate-forme Java EE, expliquez de manière claire et complète la notion d'inversion de contrôle (Inverstion of Control ou IoC).

- The HttpServletRequest interface defines a doGet method, which you have implemented in your class.
- In **your code**, you never wrote anything like this:

```
FrontController fc; HttpServletRequest request; HttpServletResponse response;
…
fc.doGet(request, response);
```

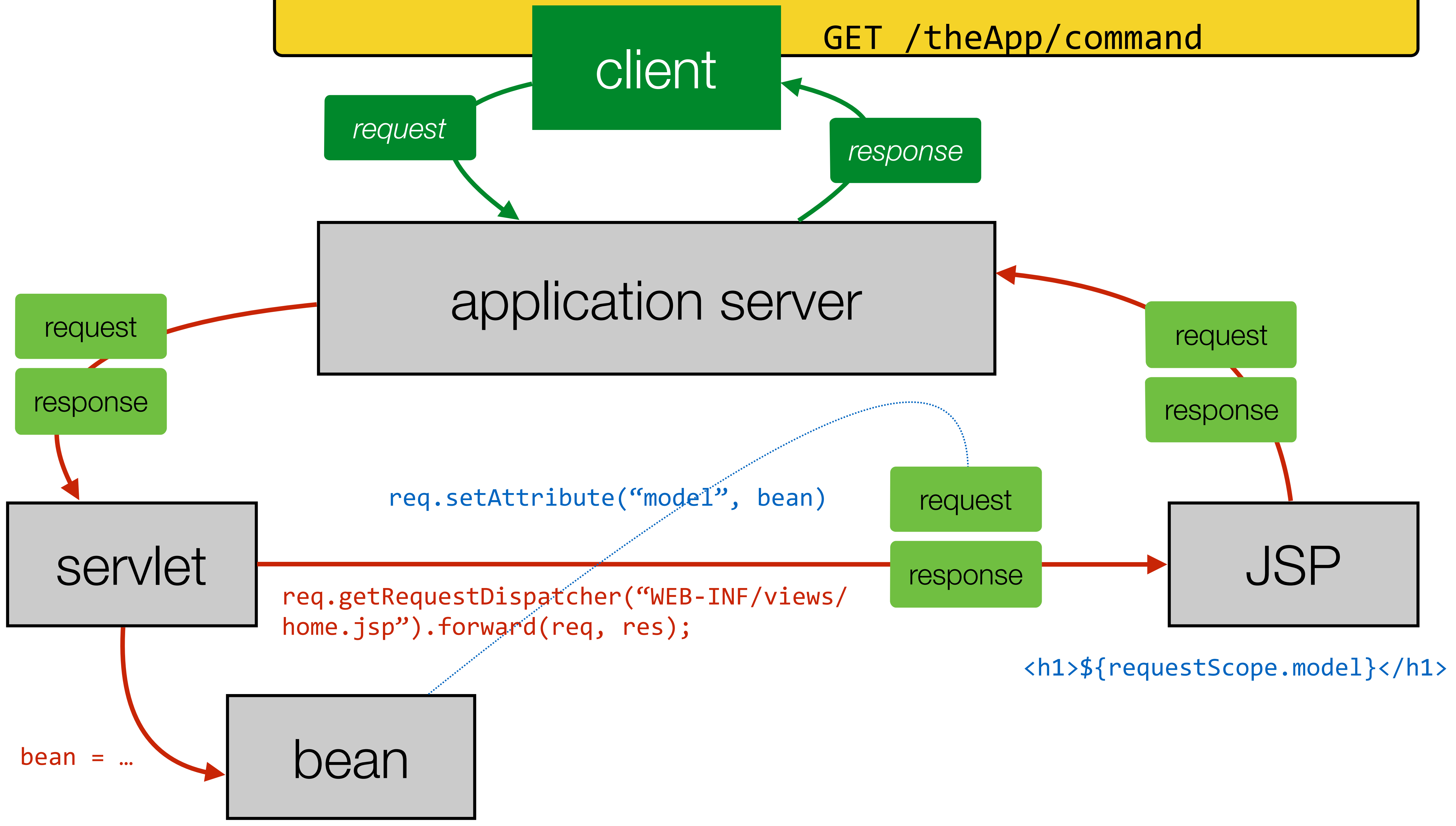
- Your code doesn't call the Java EE code. The Java EE code calls your code, at the right time.
- This is an example of **inversion of control** (IoC)

Quand vous développez un contrôleur en Java EE (MVC), comment faites-vous pour déléguer la suite du traitement à la vue? Expliquez le principe et écrivez à quoi ressemble le code.


GET /theApp/command
client
request
response
application server
request
response
request
response
req.setAttribute(“model”, bean)
request
response
servlet
JSP
req.getRequestDispatcher(“WEB-INF/views/home.jsp”).forward(req, res);
<h1>${requestScope.model}</h1>
bean = …
bean

Lecture 1, Slide 48

Quand on implémente le modèle MVC dans une application Java EE, on utilise différents types de composants (qui sont conformes à des APIs définies dans différents JSR).
Quels types de composants utilisez vous pour implémenter le M, le V et le C?


GET /theApp/command
client
request
response
application server
request
response
request
response
servlet
req.setAttribute(“model”, bean)
request
response
JSP
req.getRequestDispatcher(“WEB-INF/views/home.jsp”).forward(req, res);
<h1>${requestScope.model}</h1>
bean
bean = …

Lecture 1, Slide 48

Quand vous développez une vue en Java EE (MVC), comment faites-vous pour accéder au modèle qui vous a été transmis par le contrôleur? Expliquez le principe et écrivez à quoi ressemble le code.

Si on voulait supprimer la ligne 12 (@WebServlet(name = ...) de la classe MysteryController, que devrait-on faire pour que l'application continue à fonctionner de la même manière?

What is the purpose of the file named `web.xml`? How it called in the Java EE specifications? Why is it **not mandatory** anymore?

Les servlets sont utilisés uniquement pour générer des pages web complètes (ils ne peuvent pas répondre à des requêtes AJAX).

Un développeur a créé un fichier accountDetails.jsp dans une application Java EE. Comment peut-il, sans écrire de code, empêcher un client d'y accéder directement?

- Create a **views folder** in the WEB-INF directory (good for security: they will not be directly accessible, WEB-INF is protected!).

Avec un exemple de code, expliquez pourquoi on dit qu'un servlet n'est pas thread safe. Expliquez ce que le développeur doit faire pour se protéger d'éventuels bugs liés à cette propriété.

- In general, application servers create **one instance of every servlet** (see specifications of the Servlet API for exceptions to this rule).
- In general, incoming HTTP requests are processed in parallel, by **multiple threads**.
- **Servlets are NOT thread safe**. This means that if you use instance variables in your servlets, you must **synchronize** the access to these variables!
- This also highlights a very important point: if you do only **manual, ad-hoc testing**, you are probably not going to encounter concurrency bugs! This is why it is important to implement **tests where you simulate concurrent users**.

```
@WebServlet(name = "ConcurrencyServlet", urlPatterns = {"/pages/concurrency"})
public class ConcurrencyServlet extends HttpServlet {

  private long numberOfRequests = 0;
  private long numberOfRequests2 = 0;

  private void incrementNumberOfRequests() {
    long tempValue = numberOfRequests;
    tempValue = tempValue + 1;
    numberOfRequests = tempValue;
  }

  private synchronized void incrementNumberOfRequests2() {
    long tempValue = numberOfRequests2;
    tempValue = tempValue + 1;
    numberOfRequests2 = tempValue;
  }

  protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    incrementNumberOfRequests();
    incrementNumberOfRequests2();
    request.setAttribute("numberOfRequests", numberOfRequests);
    request.setAttribute("numberOfRequests2", numberOfRequests2);
    request.getRequestDispatcher("/WEB-INF/pages/concurrency.jsp").forward(request, response);
  }
```

Comment s’appelle le framework de test qui permet de valider votre interface utilisateur en pilotant un navigateur web?

Lecture 3, Lecture 4, DemoMVC

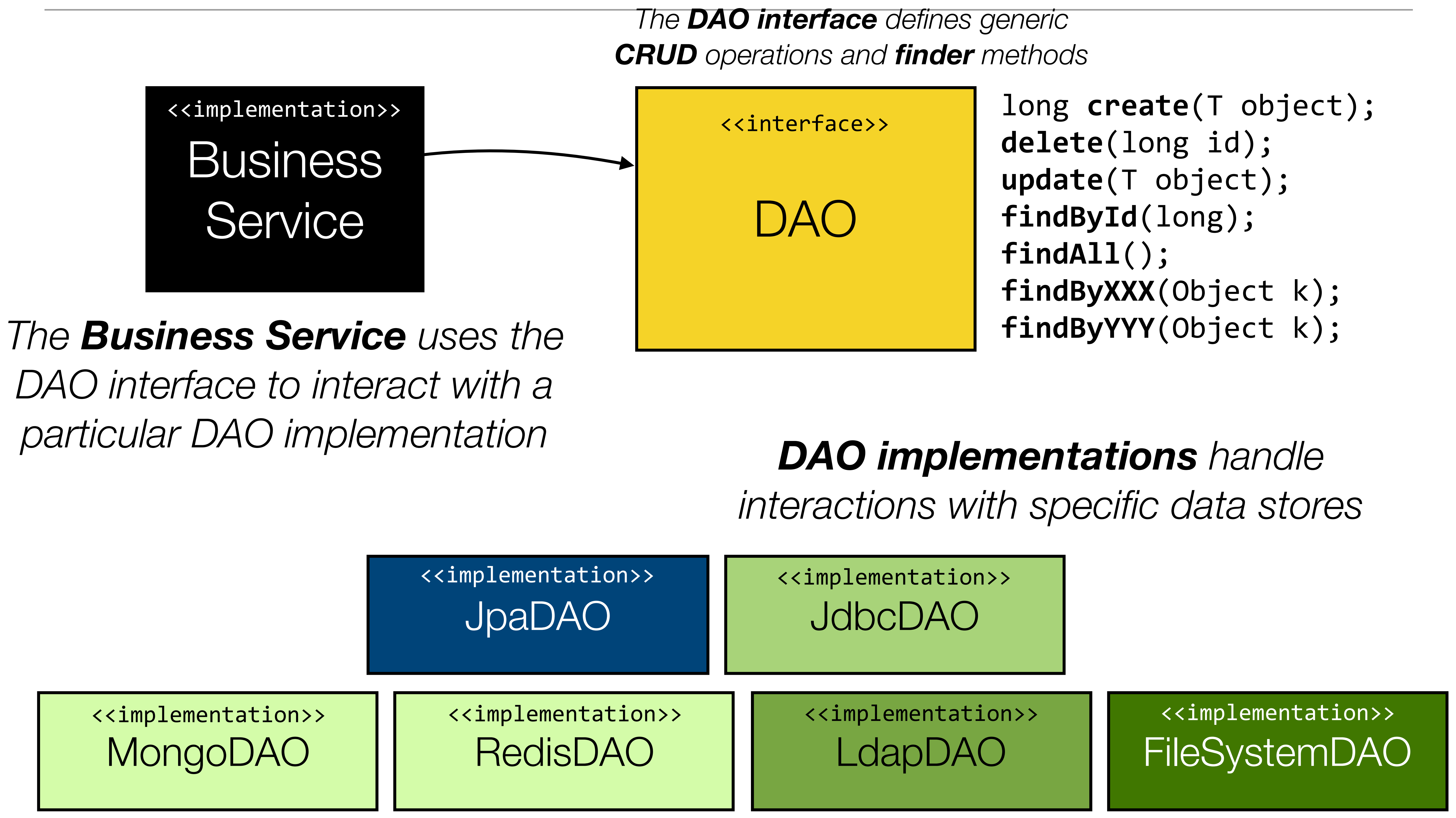
Citez deux types d’Enterprise Java Beans.

- **Stateless Session Beans** are used to implement business services, where every client request is independent.
- **Stateful Session Beans** are used for services which have a notion of conversation (e.g. shopping cart).
- **Singleton Session Beans** are used when there should be a single service instance in the app.
- **Message Driven Beans** are used together with the Java Message Service (JMS). Business logic is not invoked when a web client sends a request, but when a message arrives in a queue. We will see that later.

Que veut dire l’acronyme DAO? Expliquez l’objectif de ce design pattern.

- Most applications manipulate data that is stored in one or more **data stores**.
- There are **different ways** to implement a data store. Think about specific RDMS, NoSQL DBs, LDAP servers, file systems, etc.
- When you implement business logic, you would like to create code that is **independent** from a particular data store implementation (*).
- In other words, you want to **reduce coupling** between your business service and your data store implementation.
- When you apply the **Data Access Object** design pattern, you create an abstraction layer to achieve this goal.

Décrivez la structure du design pattern DAO, en fournissant un diagramme de classes (avec classes, interfaces, attributs et signature des méthodes).



The ***DAO interface*** *defines generic* ***CRUD*** *operations and* ***finder*** *methods*

```
long create(T object);
delete(long id);
update(T object);
findById(long);
findAll();
findByXXX(Object k);
findByYYY(Object k);
```

The ***Business Service*** *uses the DAO interface to interact with a particular DAO implementation*

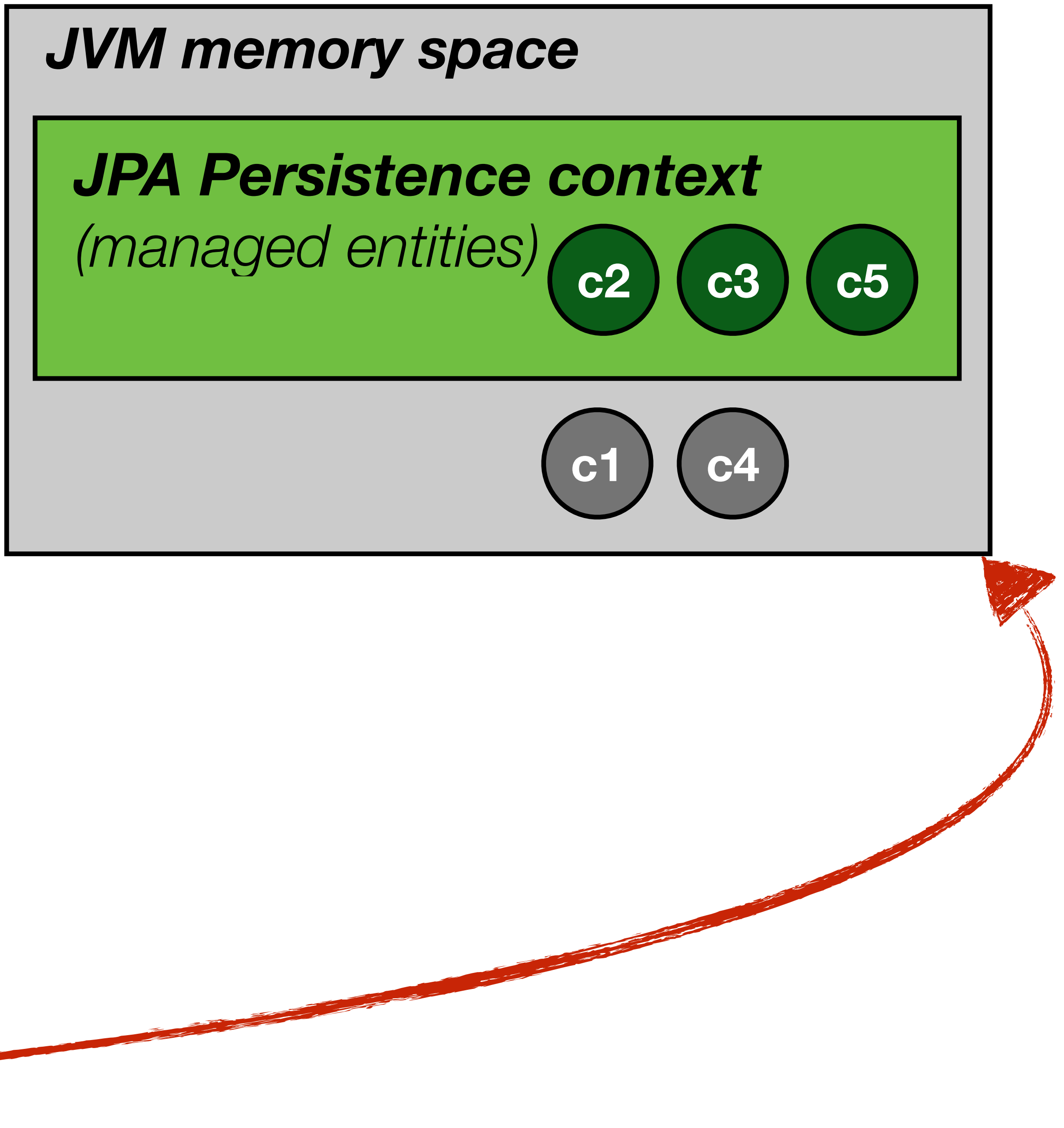
DAO implementations *handle interactions with specific data stores*

Lecture 3, slide 6

Listing Academic System: la colonne STUDENT_ID de la table BACHELORPROJECT contient la valeur NULL. Identifiez où se trouve le problème et expliquez de manière claire et complète la raison de ce comportement.

The persistence context is **created** when **transaction** begins and is **flushed** when transaction commits (or rollbacks).

A **transaction** is started by the **EJB container** whenever a business method is called. It is committed by the container when it returns (or rollbacked if there is an exception).



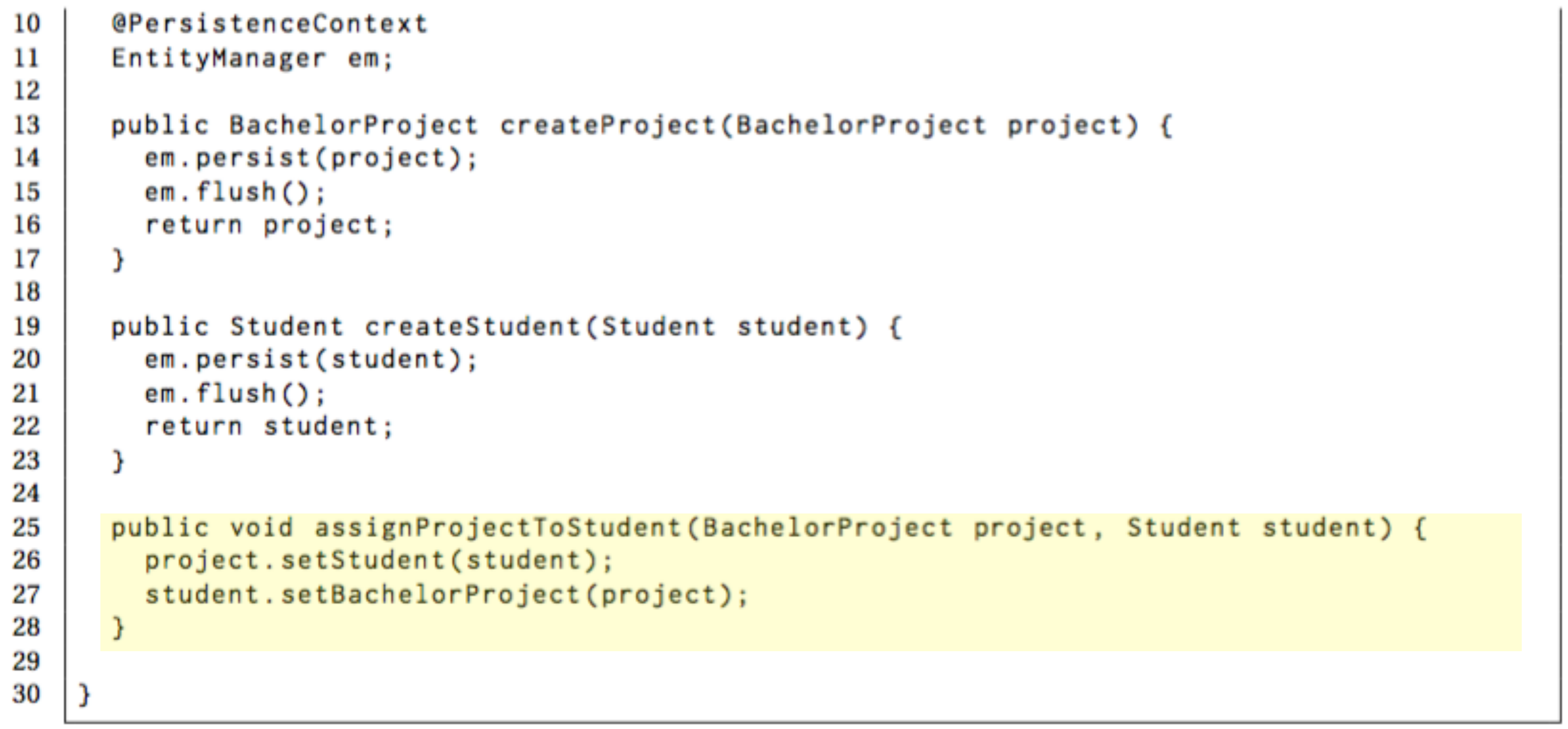
```
@Stateless
public class Manager {

  @PersistenceContext
  EntityManager em;

  public void businessMethod() {
    Customer c1 = new Customer();

    Customer c2 = new Customer();
    em.persist(c2);

    Customer c3 = em.find(123);

    Customer c4 = new Customer(246, "john", "doe");
    Customer c5 = em.merge(c4);
```

Lecture 3, slide 52

Proposez une modification de code pour corriger ce bug.

```java
package ch.heigvd.amt.school;

import java.io.IOException;
import java.io.PrintWriter;
import javax.ejb.EJB;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet(name = "AcademicController", urlPatterns = {"/academic"})
public class AcademicController extends HttpServlet {

  @EJB
  AcademicService academicService;

  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
      ServletException, IOException {
    response.setContentType("text/plain;charset=UTF-8");
    BachelorProject project = academicService.createProject(new BachelorProject("Smart
        Fridge"));
    Student student = academicService.createStudent(new Student("Olivier", "Liechti"));
    academicService.assignProjectToStudent(project, student);
    PrintWriter out = response.getWriter();
    out.println("done");
  }
}
```

Proposez une modification de code pour corriger ce bug.



```java
  @PersistenceContext
  EntityManager em;

  public BachelorProject createProject(BachelorProject project) {
    em.persist(project);
    em.flush();
    return project;
  }

  public Student createStudent(Student student) {
    em.persist(student);
    em.flush();
    return student;
  }

  public void assignProjectToStudent(BachelorProject project, Student student) {
    project.setStudent(student);
    student.setBachelorProject(project);
  }

}
```

L'interface javax.persistence.EntityManager définit la méthode <T> T merge (T entity). Expliquez en détails à quoi elle sert, sur la base d'un exem- ple. Expliquez ce qu'elle accepte en paramètre et qu'elle retourne comme résultat.

```
Customer c1 = n
```

```
Customer c2 = new Customer();
// c2 is not in persistence ctx

em.persist(c2);
// c2 is in the persistence ctx
```

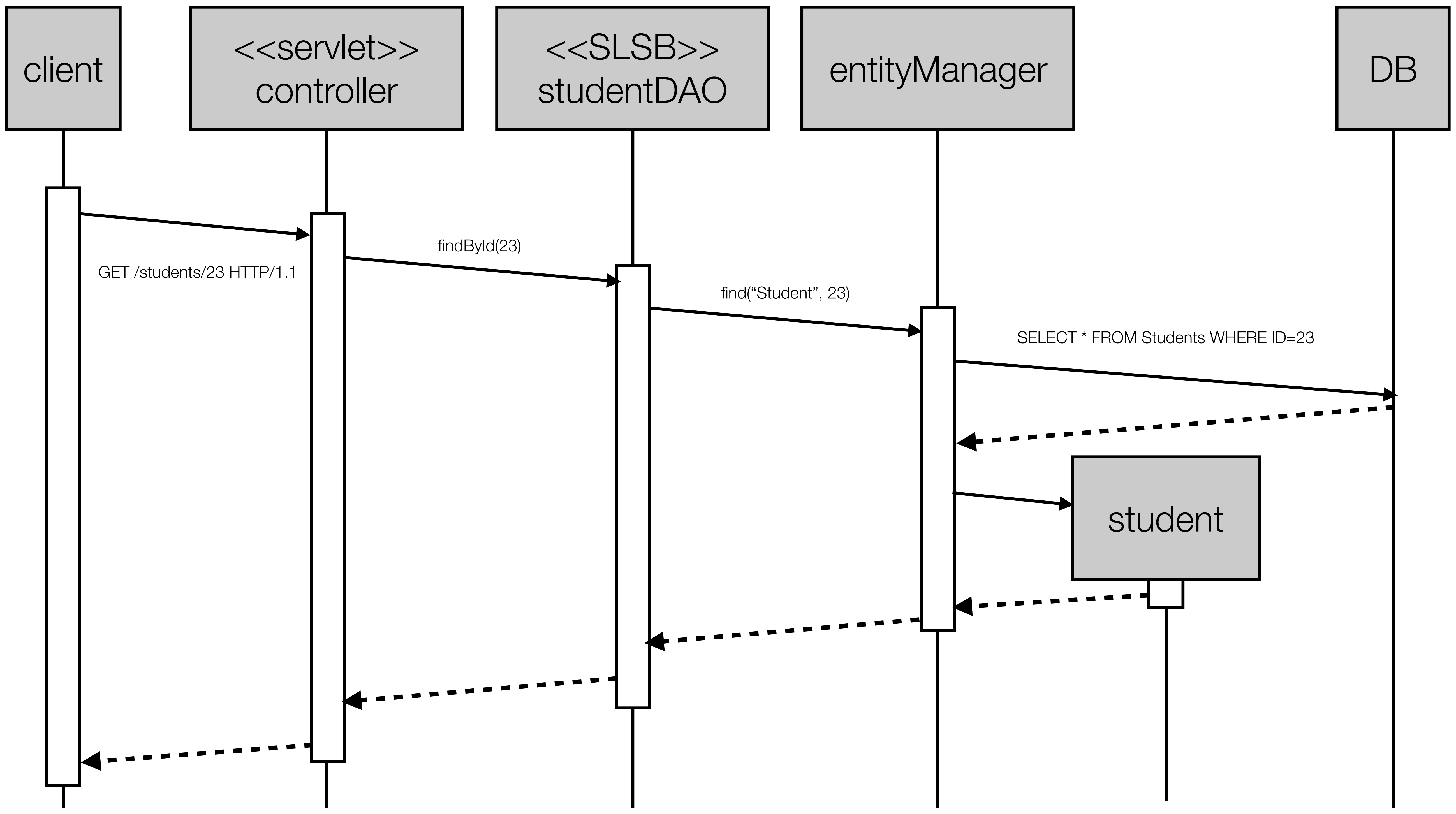
```
Customer c3 = em.find(123);
```

```
Customer c4 = new
  Customer(246, “john”, “doe”);
Customer c5 = em.merge(c4);
```

c4 is a simple POJO. When **em.merge(c4)** is invoked, a SELECT statement will be issued to retrieve a row where the primary key is equal to 246. A new entity is created and its properties are copied from c4 (to update the DB later on). **WARNING: c5 is in the persistence context, c4 is not!!**

Lecture 3, slide 53

Décrivez la structure du design pattern DAO, en fournissant un diagramme de séquence.



1. A quoi l’API JAX-RS sert-elle?

“à implémenter des API REST”

“Lorsqu’on implémente une API REST, on définit quelle logique doit être exécutée lorsque des requêtes HTTP sont envoyées par des clients (en fonction du verbe, de l’URL, du query string, des entêtes).

Il est tout à fait possible de faire cela avec des servlets.

JAX-RS (pour Restful Services) offre une alternative, en proposant une API de plus haut niveau, ce qui facilite la vie du développeur.

2. Donnez un exemple de code qui utilise JAX-RS. Commentez et expliquez ce code.

```
@Stateless
@Path("/beers")
public class BeersResource {

  @EJB
  BeersManagerLocal beersManager;

  @GET
  @Produces("application/json")
  public List<Beer> getBeers() {
    return beersManager.getAllBeers();
  }

  @POST
  @Consumes("application/json")
  public long addBeer(Beer beer) {
    return beersManager.add(beer);
  }
}
```

Even if we are conceptually in the presentation tier, we declare our JAX-RS resource class as a Stateless Session Bean.

This allows us to inject EJBs in the class. This also facilitates the use of the JPA persistence context (transaction starts when we enter the JAX-RS method)

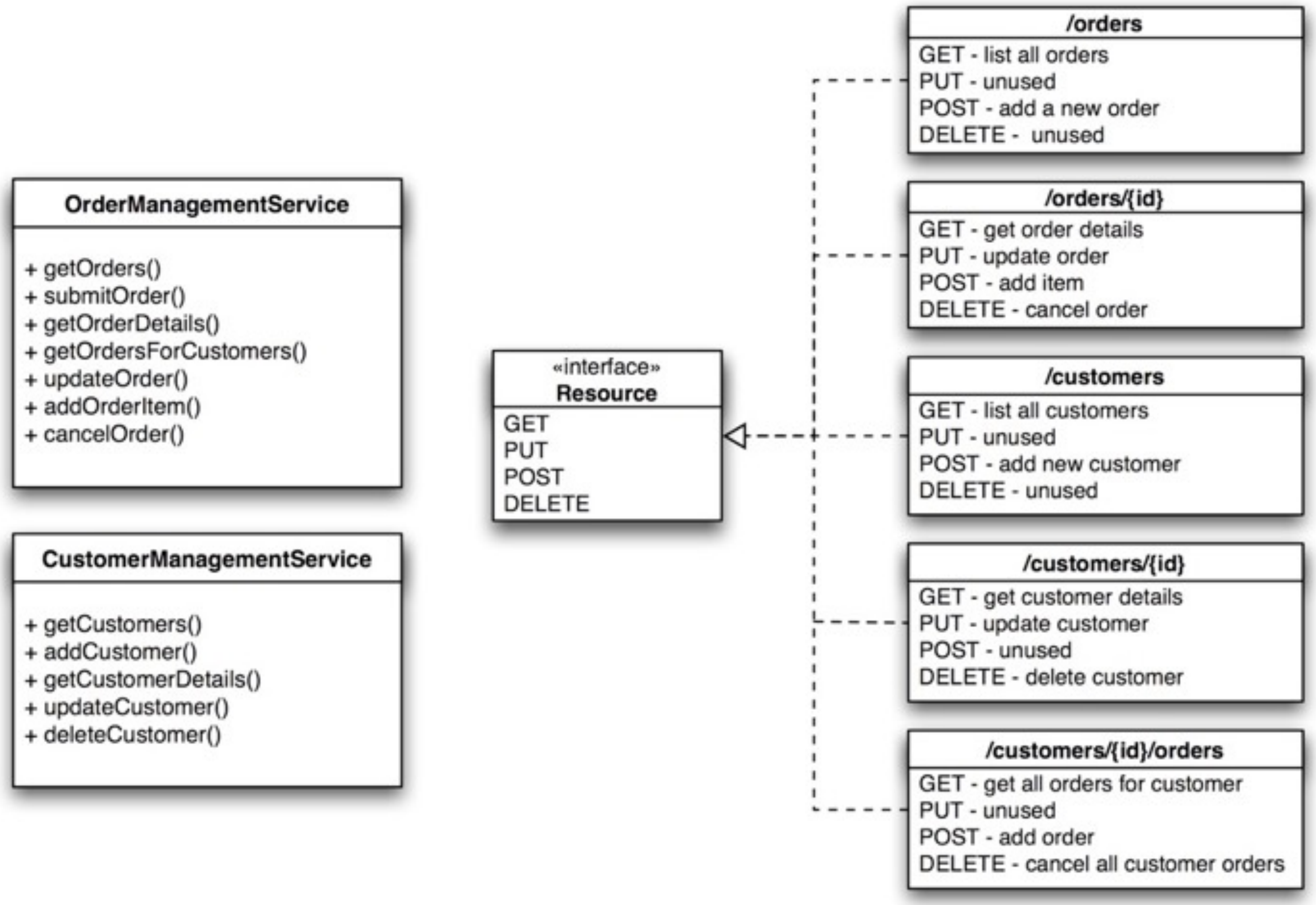
IN THIS CASE, **DO NOT DEFINE A LOCAL INTERFACE** (it makes things a bit more complicated...)

```
GET /beers/ HTTP/1.1
Host: localhost:8080
Accept: application/json
```

```
POST /beers/ HTTP/1.1
Host: localhost:8080
Content-type: application/json

{
  "name" : "Cardinal",
  "country" : "Switzerland"
}
```

Lecture 4, slide 49

Si vous implémentez une API REST en respectant les bonnes pratiques, pour quelles opérations décidez-vous d'utiliser les méthodes HTTP PUT, respectivement POST? Expliquez de manière précise et détaillée, avec des exemples.



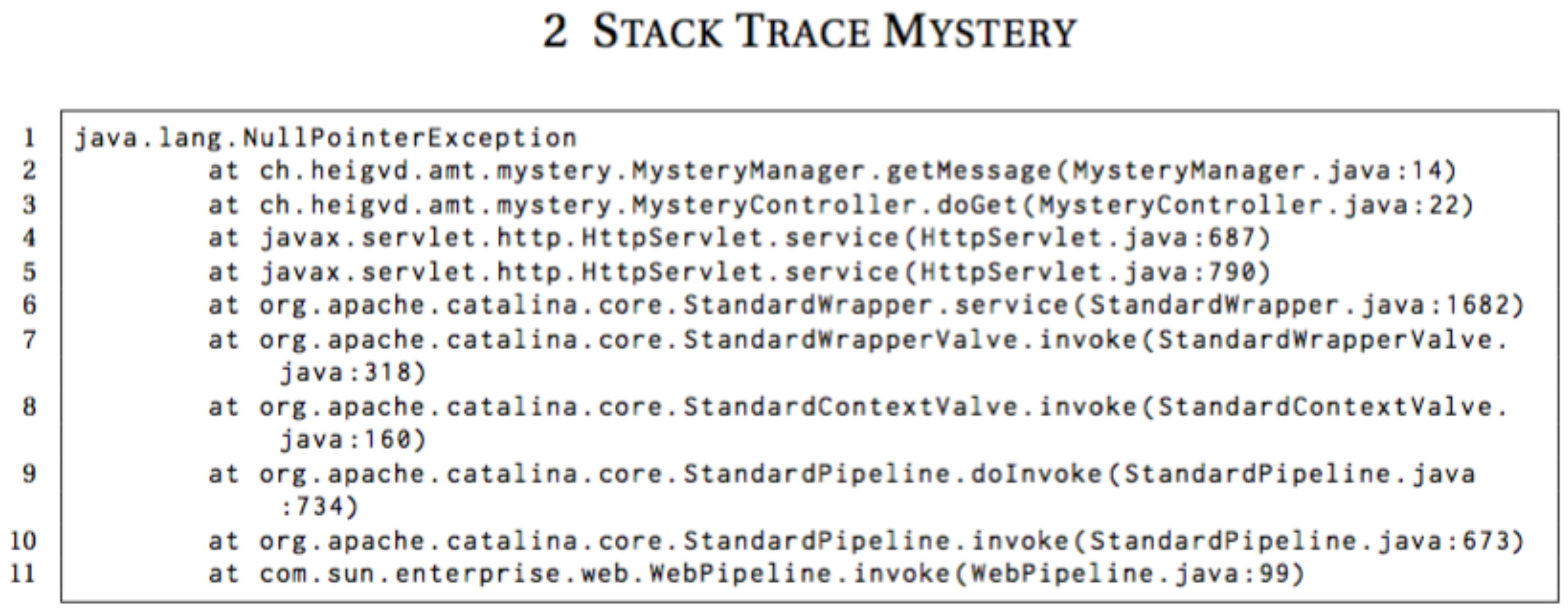
Donnez un argument pour appliquer le pattern DTO dans une application Java EE? Expliquez de manière claire et détaillée, avec des exemples.

- **Pattern benefits**: there are **at least 4 reasons** for applying the pattern:
  - Control on the **visibility** of your data (**security**, **confidentiality**). In the MVCDemo project, employees have a **salary**. While it is necessary to have this property in the JPA entity (because it has to be stored in the database), it is clearly not something that you want to leak out via your REST API.
  - Have **full control on the data structure** presented to your clients. **Your API will be cleaner and easier to use**.
  - **Reduce the chattiness** and **improve the performance** of your clients applications. If you use JPA entities, then it is likely that REST clients will need to send a lot of HTTP requests to get all components of a page (1 call for the company, n calls for the sectors, etc.). With a DTO layer, you can aggregate multiple small business entities into a coarse-grained object that you send over the network.
  - **Avoid tricky technical issues.** If you try to use JPA entities, you will have to deal with circular references (@XmlTransient) and other issues. Trust me, you will spend a lot of time fighting with the underlying frameworks.

Donnez un deuxième argument pour appliquer le pattern DTO dans une application Java EE? Expliquez de manière claire et détaillée, avec des exemples.

- **Pattern benefits**: there are **at least 4 reasons** for applying the pattern:
  - Control on the **visibility** of your data (**security**, **confidentiality**). In the MVCDemo project, employees have a **salary.** While it is necessary to have this property in the JPA entity (because it has to be stored in the database), it is clearly not something that you want to leak out via your REST API.
  - Have **full control on the data structure** presented to your clients. **Your API will be cleaner and easier to use**.
  - **Reduce the chattiness** and **improve the performance** of your clients applications. If you use JPA entities, then it is likely that REST clients will need to send a lot of HTTP requests to get all components of a page (1 call for the company, n calls for the sectors, etc.). With a DTO layer, you can aggregate multiple small business entities into a coarse-grained object that you send over the network.
  - **Avoid tricky technical issues.** If you try to use JPA entities, you will have to deal with circular references (@XmlTransient) and other issues. Trust me, you will spend a lot of time fighting with the underlying frameworks.

Lisez très attentivement les sources du listing Mystery et le stack trace Mystery qui est généré quand un client HTTP envoie une requête vers http://localhost:8080/MysteryProject/mystery. Quelle est la cause initiale (root cause) de cette exception? Que faut-il modifier dans le code pour éviter qu'elle se produise et que le texte abracadabra s'affiche correctement dans le navigateur?



2 Stack Trace Mystery

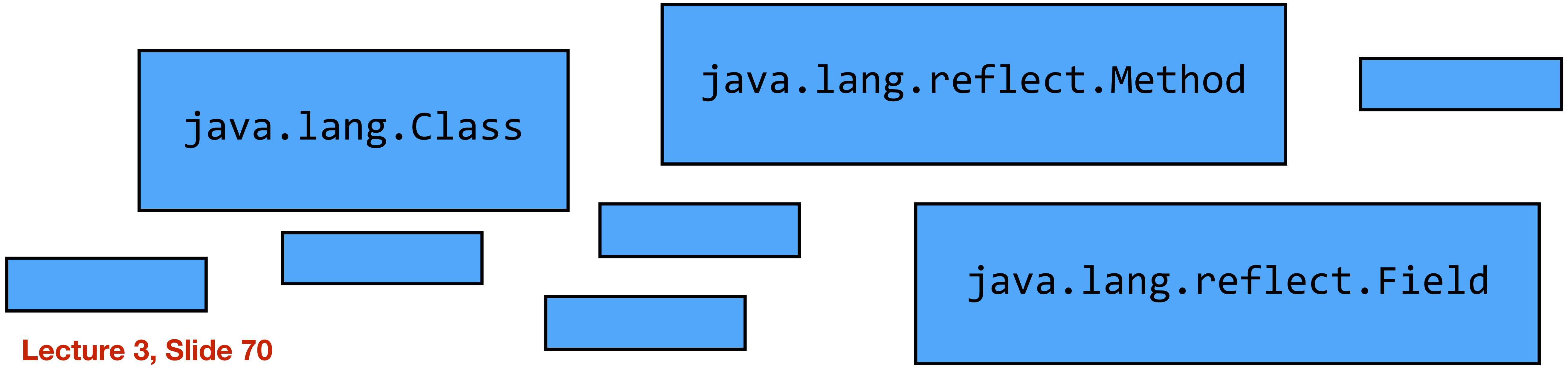
```
java.lang.NullPointerException
        at ch.heigvd.amt.mystery.MysteryManager.getMessage(MysteryManager.java:14)
        at ch.heigvd.amt.mystery.MysteryController.doGet(MysteryController.java:22)
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:687)
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:790)
        at org.apache.catalina.core.StandardWrapper.service(StandardWrapper.java:1682)
        at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.
            java:318)
        at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.
            java:160)
        at org.apache.catalina.core.StandardPipeline.doInvoke(StandardPipeline.java
            :734)
        at org.apache.catalina.core.StandardPipeline.invoke(StandardPipeline.java:673)
        at com.sun.enterprise.web.WebPipeline.invoke(WebPipeline.java:99)
```

```java
package ch.heigvd.amt.mystery;

import javax.ejb.EJB;
import javax.ejb.Stateless;

@Stateless
public class MysteryManager implements MysteryManagerLocal {

  @EJB
  MagicManagerLocal magicManager;

  @Override
  public String getMessage() {
    return magicManager.doMagic(null);
  }
}
```

```java
package ch.heigvd.amt.mystery;

import java.io.IOException;
import java.io.PrintWriter;
import javax.ejb.EJB;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet(name = "MysteryController", urlPatterns = {"/mystery"})
public class MysteryController extends HttpServlet {

  @EJB
  MysteryManagerLocal manager;

  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
       ServletException, IOException {
    manager = new MysteryManager();
    response.setContentType("text/plain;charset=UTF-8");
    PrintWriter out = response.getWriter();
    out.println("message: " + manager.getMessage());
  }
}
```

Expliquez à quoi sert l'API de reflection en Java. Expliquez clairement et donnez des exemples concrets (pas besoin d'écrire du code, mais dites très concrètement ce qu'on peut faire).

- Reflection is a mechanism, through which a program can **inspect and manipulate** its structure and behavior **at runtime.**
- In Java, this means that a program can **get information about classes, their fields, their methods, etc.**
- In Java, this also means that a program can **create instances of classes dynamically** (based on their names, as in the example of JDBC drivers), **invoke methods**, etc.



Donnez un exemple d'utilisation de la reflection au sein de Java EE. Expliquez de manière claire et complète.