Java EE

Olivier Liechti & Laurent Prévost
COMEM Web Services

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

What is Java Enterprise Edition?

- It is a development platform: it provides high-level APIs to develop software components.
- It is an execution platform: it provides an environment to deploy and bring these components “to live”.
- It is an entreprise platform: it provides support for distributed transactions, security, integration, etc.
- Separation of concerns: "The developer takes care of the business logic. Java EE takes care of the systemic qualities".

http://flickr.com/photos/decade_null/427124229/sizes/m/#cc_license

Java EE and standards

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

- Java EE is a specification
  - Defined through the JCP, it is a specification that software editors can decide to implement. Java EE 5 is defined in JSR 244.
- Java EE is an “umbrella” specification
  - Java EE builds upon other specifications (servlets, EJBs, JDBC, etc.) and specifies which specifications (and which versions) need to be implemented by a Java EE certified application server.
  - Java EE also defines a programming model and defines several roles (developer, assembler, deployer, etc.).

Java Community Process

Specification Lead
Bill Shannon Sun Microsystems, Inc.

Expert Group
Barreto, Charlton
Capgemini
Dudney, Bill
Hewlett-Packard
Kohen, Elika S.
Oracle
Pratap, Rama Murthy Amar
Reinshagen, Dirk
Shah, Suneet
Tiwari, Ashish
Umapathy, Sivasundaram

BEA Systems
Chandrasekaran, Muralidharan
E.piphany, Inc.
IBM
Leme, Felipe
OW2
Raible, Matt
SAP AG
Sun Microsystems, Inc.
Tmax Soft, Inc.

Borland Software Corporation
Crawford, Scott
Genender, Jeff
Ironflare AB
Novell, Inc.
Pramati Technologies
Red Hat Middleware LLC
SeeBeyond Technology Corp.
Sybase
Trifork

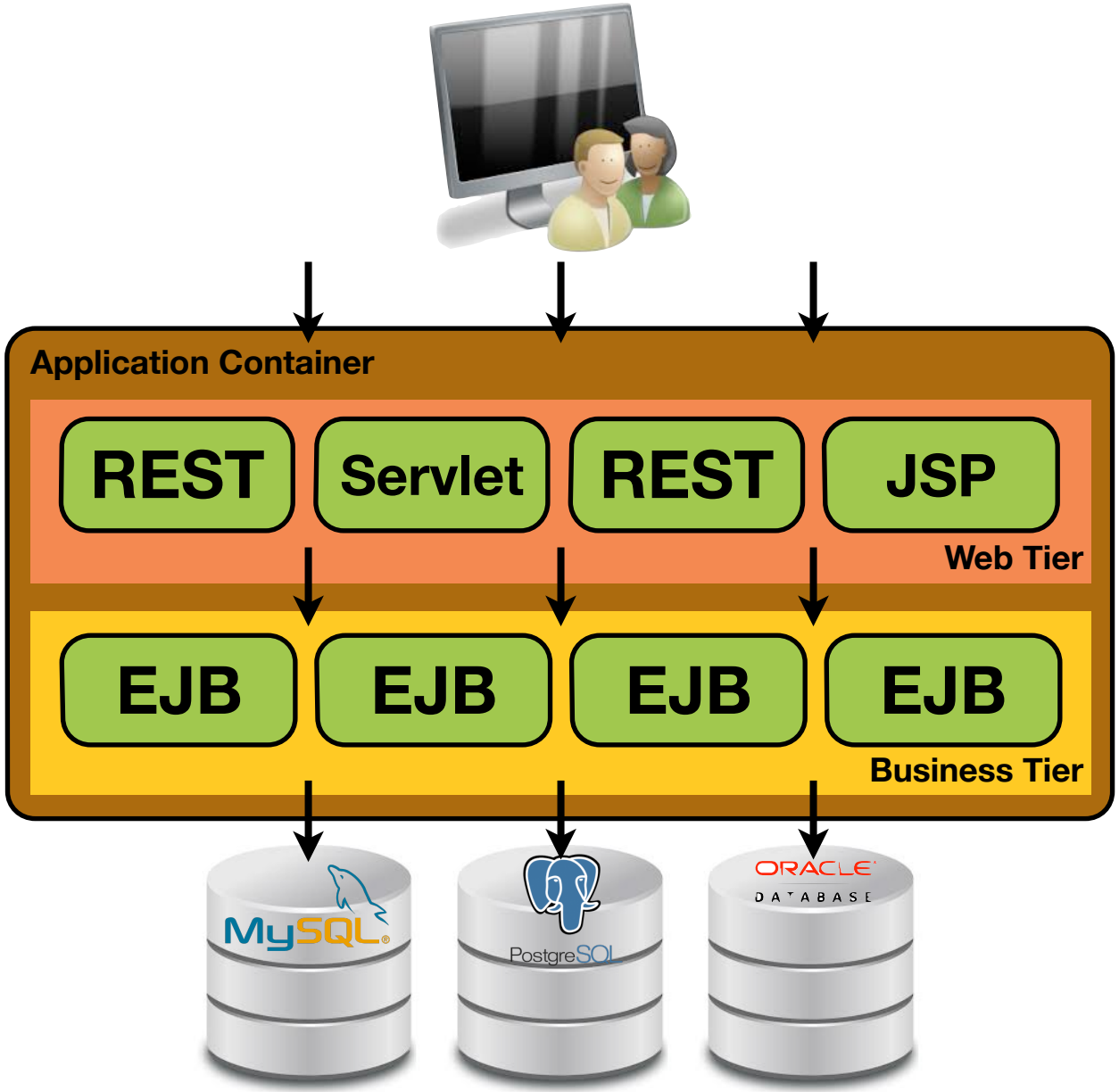
Architecture

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

- 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.).
- Key notion in the Java EE architecture: the **containers**
  - a container is an **environment** in which we deploy components;
  - a container **provides services** (transactions, security, etc.) through APIs;
  - there are different containers in Java EE: the "**web**" container, the "**ejb**" container and even a "**client**" container that can be used for rich clients.

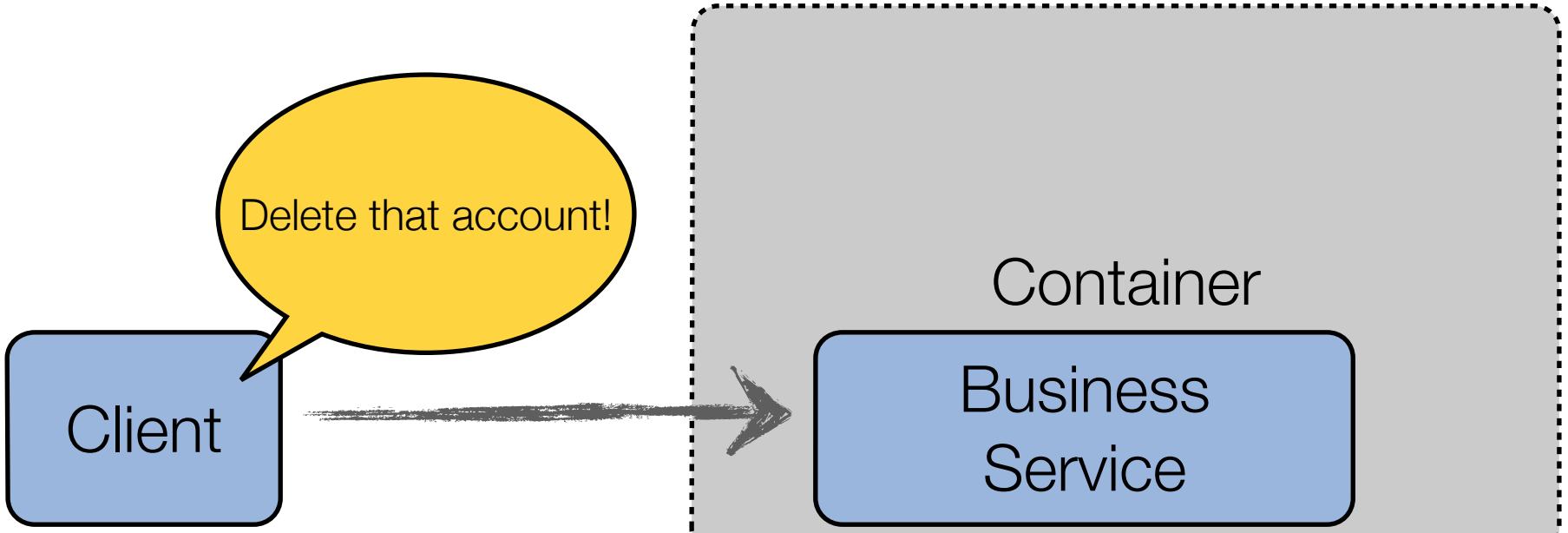
Java EE - Tiers

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



Mediated access

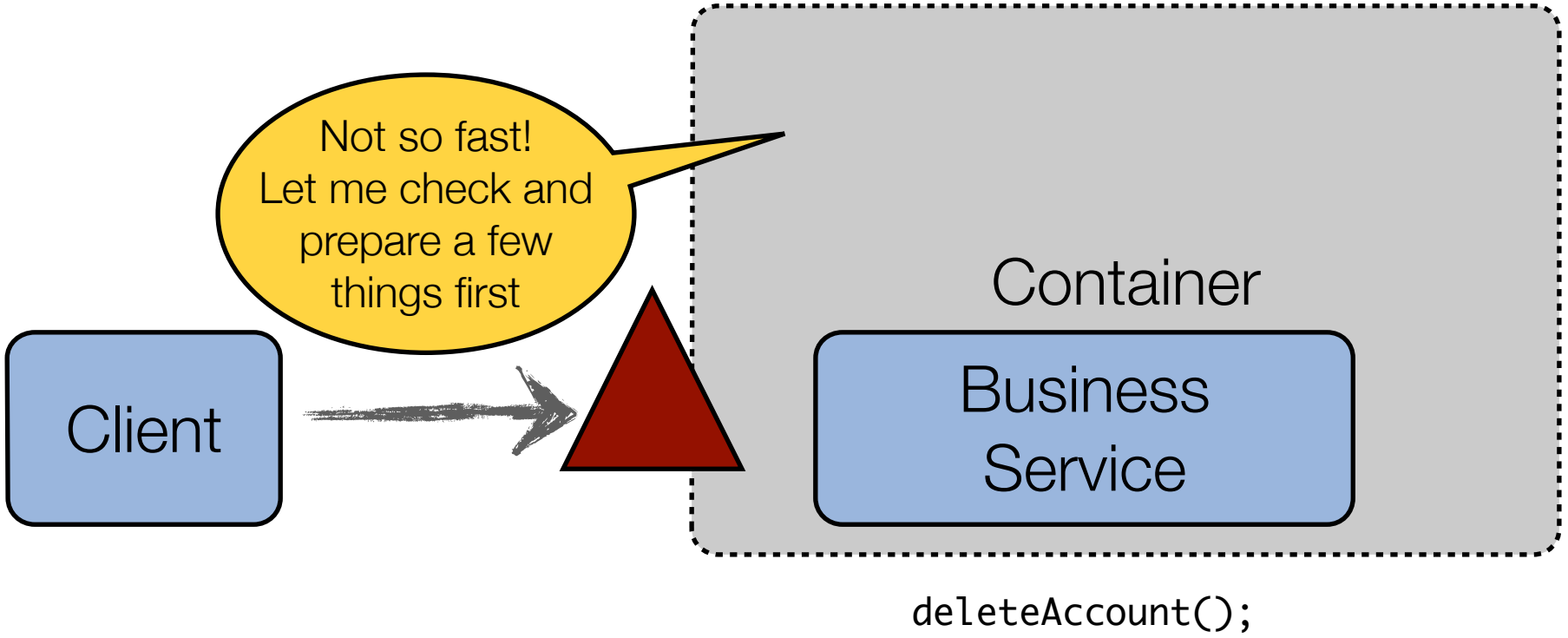
heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud




```
deleteAccount();
```

Mediated access

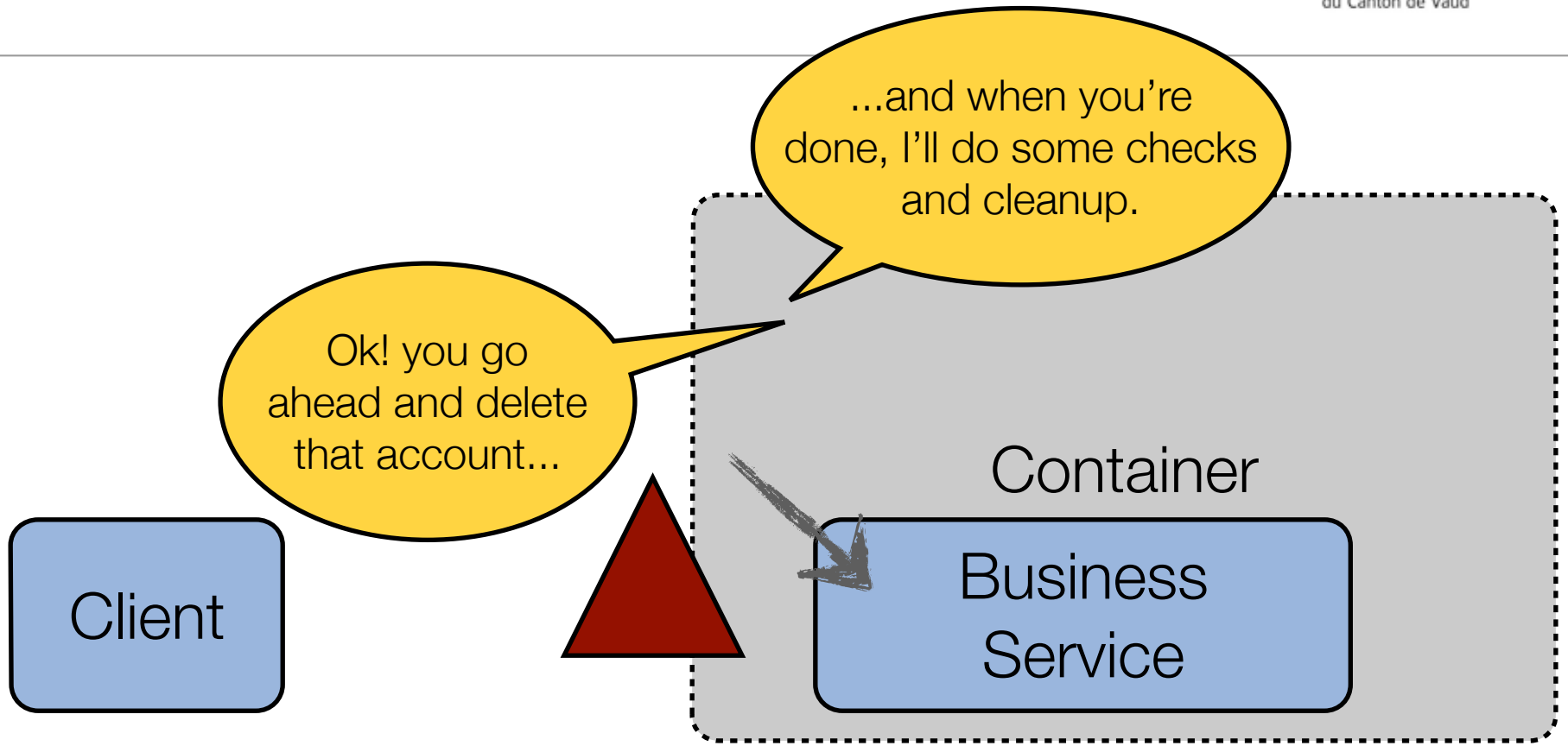
heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud




```
deleteAccount();
```

Mediated access

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




```
deleteAccount();
```

Entity in code

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

```java
package ch.heigvd.ptl.jee.sample.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.Id;

@Entity
public class User {
   public static enum Role {
    ADMIN,
    MEMBER;
   }

   @Id
   private Long id;

   @Column(length = 50)
   private String username;

   @Enumerated(EnumType.STRING)
   private Role role;
}
```

Entity in code

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

Allow the container to recognize models.

```
package ch.heigvd.ptl.jee.sample.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.Id;

@Entity
public class User {
   public static enum Role {
    ADMIN,
    MEMBER;
   }

   @Id
   private Long id;

   @Column(length = 50)
   private String username;

   @Enumerated(EnumType.STRING)
   private Role role;
}
```

Entity in code

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

Allow the container to recognize models.

Tell JPA this is the primary key.

```
package ch.heigvd.ptl.jee.sample.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.Id;

@Entity
public class User {
   public static enum Role {
    ADMIN,
    MEMBER;
   }

   @Id
   private Long id;

   @Column(length = 50)
   private String username;

   @Enumerated(EnumType.STRING)
   private Role role;
}
```

Entity in code

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

```java
package ch.heigvd.ptl.jee.sample.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.Id;

@Entity
public class User {
   public static enum Role {
    ADMIN,
    MEMBER;
   }

   @Id
   private Long id;

   @Column(length = 50)
   private String username;

   @Enumerated(EnumType.STRING)
   private Role role;
}
```

Allow the container to recognize models.

Tell JPA this is the primary key.

Tell JPA to constraint the length of the data to 50 chars.

Entity in code

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

```java
package ch.heigvd.ptl.jee.sample.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.Id;

@Entity
public class User {
   public static enum Role {
    ADMIN,
    MEMBER;
   }

   @Id
   private Long id;

   @Column(length = 50)
   private String username;

   @Enumerated(EnumType.STRING)
   private Role role;
}
```

Allow the container to recognize models.

Tell JPA this is the primary key.

Tell JPA to constraint the length of the data to 50 chars.

Tell JPA how are stored the enumeration values in the database.

Entity in code

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

```
package ch.heigvd.ptl.jee.sample.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.Id;

@Entity
public class User {
   public static enum Role {
    ADMIN,
    MEMBER;
   }

   @Id
   private Long id;

   @Column(length = 50)
   private String username;

   @Enumerated(EnumType.STRING)
   private Role role;
}
```

These annotations are used to generate / bind the underlying database.

EJB in code

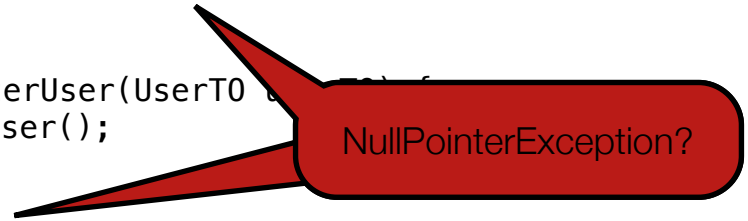
heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

```
package ch.heigvd.ptl.jee.sample.service;

import ch.heigvd.ptl.jee.sample.model.User;
import ch.heigvd.ptl.jee.sample.to.UserTO;
import javax.ejb.EJB;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

@Stateless
public class UserServiceImpl implements UserService {
   @PersistenceContext(name = "PU")
   private EntityManager em;

   @EJB
   private UtilityService utilityService;

   @Override
   public User registerUser(UserTO userTO) {
    User user = new User();

    user.setUsername(
      utilityService.trimString(userTO.getUsername())
    );
    user.setRole(User.Role.MEMBER);

    em.persist(user);

    return user;
   }
}
```

EJB in code

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

```
package ch.heigvd.ptl.jee.sample.service;

import ch.heigvd.ptl.jee.sample.model.User;
import ch.heigvd.ptl.jee.sample.to.UserTO;
import javax.ejb.EJB;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

@Stateless
public class UserServiceImpl implements UserService {
   @PersistenceContext(name = "PU")
   private EntityManager em;

   @EJB
   private UtilityService utilityService;

   @Override
   public User registerUser(UserTO
    User user = new User();

    user.setUsername(
      utilityService.trimString(userTO.getUsername())
    );
    user.setRole(User.Role.MEMBER);

    em.persist(user);

    return user;
   }
}
```



EJB in code

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

```
package ch.heigvd.ptl.jee.sample.service;

import ch.heigvd.ptl.jee.sample.model.User;
import ch.heigvd.ptl.jee.sample.to.UserTO;
import javax.ejb.EJB;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

@Stateless
public class UserServiceImpl implements UserService {
   @PersistenceContext(name = "PU")
   private EntityManager em;

   @EJB
   private UtilityService utilityService;

   @Override
   public User registerUser(UserTO
    User user = new User();

    user.setUsername(
      utilityService.trimString(userTO.getUsername())
    );
    user.setRole(User.Role.MEMBER);

    em.persist(user);

    return user;
   }
}
```

NullPointerException?

Same here ?

EJB in code

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

```
package ch.heigvd.ptl.jee.sample.service;

import ch.heigvd.ptl.jee.sample.model.User;
import ch.heigvd.ptl.jee.sample.to.UserTO;
import javax.ejb.EJB;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

@Stateless
public class UserServiceImpl implements UserService {
    @PersistenceContext(name = "PU")
    private EntityManager em;

    @EJB
    private UtilityService utilityService;

    @Override
    public User registerUser(UserTO
     User user = new User();

     user.setUsername(
       utilityService.trimString(userTO.getUsername())
     );
     user.setRole(User.Role.MEMBER);

     em.persist(user);

     return user;
    }
}
```

Nope. Both injected at runtime.

NullPointerException?

Same here ?

EJB in code

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

```java
package ch.heigvd.ptl.jee.sample.service;

import ch.heigvd.ptl.jee.sample.model.User;
import ch.heigvd.ptl.jee.sample.to.UserTO;
import javax.ejb.EJB;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

@Stateless
public class UserServiceImpl implements UserService {
   @PersistenceContext(name = "PU")
   private EntityManager em;

   @EJB
   private UtilityService utilityService;

   @Override
   public User registerUser(UserTO userTO) {
    User user = new User();

    user.setUsername(
      utilityService.trimString(userTO.getUsername())
    );
    user.setRole(User.Role.MEMBER);

    em.persist(user);

    return user;
   }
}
```

```java
package ch.heigvd.ptl.jee.sample.service;

import ch.heigvd.ptl.jee.sample.model.User;
import ch.heigvd.ptl.jee.sample.to.UserTO;
import javax.ejb.Local;

@Local
public interface UserService {
   User registerUser(UserTO userTO);
}
```

EJB in code

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

```
package ch.heigvd.ptl.jee.sample.service;

import ch.heigvd.ptl.jee.sample.model.User;
import ch.heigvd.ptl.jee.sample.to.UserTO;
import javax.ejb.EJB;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

@Stateless
public class UserServiceImpl implements UserService {
   @PersistenceContext(name = "PU")
   private EntityManager em;

   @EJB
   private UtilityService utilityService;

   @Override
   public User registerUser(UserTO userTO) {
    User user = new User();

    user.setUsername(
      utilityService.trimString(userTO.getUsername())
    );
    user.setRole(User.Role.MEMBER);

    em.persist(user);

    return user;
   }
}
```

```
package ch.

import ch.h
import ch.h
import java

@Local
public inte
   User reg
}
```

Tell the container that the service is available only in the context of the application deployed.

Another application deployed in the same container cannot inject this service.

EJB in code

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

```
package ch.hei

import ch.hei
import ch.hei
import javax.
import javax.
import javax.
import javax.

@Stateless
public class UserServiceImpl implements UserService {
    @PersistenceContext(name = "PU")
    private EntityManager em;

    @EJB
    private UtilityService utilityService;

    @Override
    public User registerUser(UserTO userTO) {
     User user = new User();

     user.setUsername(
       utilityService.trimString(userTO.getUsername())
     );
     user.setRole(User.Role.MEMBER);

     em.persist(user);

     return user;
    }
}
```

Tell to the container which kind of life cycle should be applied to this session bean.

In this case, there is not state preserved between two calls. This means that after the first call to this service, the service is put back in the EJB pool.

```
package ch.

import ch.H
import ch.H
import java

@Local
public inte
    User reg
}
```

Tell the container that the service is available only in the context of the application deployed.

Another application deployed in the same container cannot inject this service.

EJB in code

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

```
package ch.hei

import ch.hei
import ch.hei
import javax.
import javax.
import javax.
import javax.

@Stateless
public class UserServiceImpl implements UserService {
    @PersistenceContext(name = "PU")
    private EntityManager em;

    @EJB
    private U

    @Override
    public Use
     User user = new User();

     user.setUsername(
       utilityService.trimString(userTO.getUsername())
     );
     user.setRole(User.Role.MEMBER);

     em.persist(user);

     return user;
    }
}
```

Tell to the container which kind of life cycle should be applied to this session bean.

In this case, there is not state preserved between two calls. This means that after the first call to this service, the service is put back in the EJB pool.

We ask the container to manage and inject the link between our service and the data base layer. The "**PU**" name refers to the configuration file where the persistence-unit is configured.

```
package ch.

rt ch.h
rt ch.h
rt java

al
ic inte
 User reg
}
```

Tell the container that the service is available only in the context of the application deployed.

Another application deployed in the same container cannot inject this service.

EJB in code

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

```
package ch.hei

import ch.hei
import ch.hei
import javax.
import javax.
import javax.
import javax.

@Stateless
public class UserServiceImpl implements UserService {
    @PersistenceContext(name = "PU")
    private EntityManager em;

    @EJB
    private U

    @Override
    publi Use
     User  er = new User();

     user.s
       util
     );
     user.se

     em.persist(user);

     return user;
    }
}
```

Tell to the container which kind of life cycle should be applied to this session bean.

In this case, there is not state preserved between two calls. This means that after the first call to this service, the service is put back in the EJB pool.

We ask the container to manage and inject the link between our service and the data base layer. The "**PU**" name refers to the configuration file where the persistence-unit is configured.

Finally, we ask the container to inject the **UtilityService** that we use later in the code.

```
package ch.

rt ch.h
rt ch.h
rt java

al
ic inte
 User reg
}
```

Tell the container that the service is available only in the context of the application deployed.

Another application deployed in the same container cannot inject this service.

JAX-RS in code

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

```
package ch.heigvd.ptl.jee.sample.rest;

import ch.heigvd.ptl.jee.sample.model.User;
import ch.heigvd.ptl.jee.sample.service.UserService;
import ch.heigvd.ptl.jee.sample.to.UserTO;
import javax.ejb.EJB;
import javax.ws.rs.Consumes;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

@Path("/users")
public class UserResource {
    @EJB
    private UserService userService;

    @POST
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
    public Response register(UserTO userTO) {
     User userRegistered = userService.registerUser(userTO);

     UserTO userRegisteredTO = new UserTO();

     userRegisteredTO.setUsername(userRegistered.getUsername());
     userRegisteredTO.setRole(userRegistered.getRole().name());

     return Response.ok(userRegisteredTO).build();
    }
}
```

JAX-RS in code

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

```
package ch.heigvd.ptl.jee.sample.rest;

import ch.heigvd.ptl.jee.sample.model.User;
import ch.heigvd.ptl.jee.sample.service.UserService;
import ch.heigvd.ptl.jee.sample.to.UserTO;
import javax.ejb.EJB;
import
import
import
import
import
import

@Path("/users")
public class UserResource {
    @EJB
    private UserService userService;

    @POST
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
    public Response register(UserTO userTO) {
     User userRegistered = userService.registerUser(userTO);

     UserTO userRegisteredTO = new UserTO();

     userRegisteredTO.setUsername(userRegistered.getUsername());
     userRegisteredTO.setRole(userRegistered.getRole().name());

     return Response.ok(userRegisteredTO).build();
    }
}
```

Define a base path for the whole class. This means that all exposed verbs inside this class will have, at least, /users in their path.

JAX-RS in code

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

```
package ch.heigvd.ptl.jee.sample.rest;

import ch.heigvd.ptl.jee.sample.model.User;
import ch.heigvd.ptl.jee.sample.service.UserService;
import ch.heigvd.ptl.jee.sample.to.UserTO;
import javax.ejb.EJB;
import
import
import
import
import
import

@Path("/users")
public class U
    @EJB
    private UserService userService;

    @POST
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
    public Response register(UserTO userTO) {
     User userRegistered = userService.registerUser(userTO);

     UserTO userRegisteredTO = new UserTO();

     userRegisteredTO.setUsername(userRegistered.getUsername());
     userRegisteredTO.setRole(userRegistered.getRole().name());

     return Response.ok(userRegisteredTO).build();
    }
}
```

Define a base path for the whole class. This means that all exposed verbs inside this class will have, at least, /users in their path.

Again, container magic happens for injection.

JAX-RS in code

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

```
package ch.heigvd.ptl.jee.sample.rest;

import ch.heigvd.ptl.jee.sample.model.User;
import ch.heigvd.ptl.jee.sample.service.UserService;
import ch.heigvd.ptl.jee.sample.to.UserTO;
import javax.ejb.EJB;
import
import
import
import
import
import

@Path("/users")
public class U
    @EJB
    private Us

    @POST
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
    public Response register(UserTO userTO) {
     User userRegistered = userService.registerUser(userTO);

     UserTO userRegisteredTO = new UserTO();

     userRegisteredTO.setUsername(userRegistered.getUsername());
     userRegisteredTO.setRole(userRegistered.getRole().name());

     return Response.ok(userRegisteredTO).build();
    }
}
```

Define a base path for the whole class. This means that all exposed verbs inside this class will have, at least, /users in their path.

Again, container magic happens for injection.

Method register will be called when we do a post on /users

JAX-RS in code

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

```
package ch.heigvd.ptl.jee.sample.rest;

import ch.heigvd.ptl.jee.sample.model.User;
import ch.heigvd.ptl.jee.sample.service.UserService;
import ch.heigvd.ptl.jee.sample.to.UserTO;
import javax.ejb.EJB;
import
import
import
import
import
import

@Path("/users")
public class U
    @EJB
    private Us

    @POST
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
    public Respon e register(UserTO userTO) {
     User userRe

     UserTO use

     userRegiste
     userRegisteredTO.setRole(userRegistered.getRole().name());

     return Response.ok(userRegisteredTO).build();
    }
}
```

Define a base path for the whole class. This means that all exposed verbs inside this class will have, at least, /users in their path.

Again, container magic happens for injection.

Method register will be called when we do a post on /users

Consumes and **Produces** will define what is accepted as a representation format and what will be rendered as a response.

JAX-RS in code

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

```java
package ch.heigvd.ptl.jee.sample.rest;

import ch.heigvd.ptl.jee.sample.model.User;
import ch.heigvd.ptl.jee.sample.service.UserService;
import ch.heigvd.ptl.jee.sample.to.UserTO;
import javax.ejb.EJB;
import javax.ws.rs.Consumes;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

@Path("/users")
public class UserResource {
    @EJB
    private UserService userService;

    @POST
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
    public Response register(UserTO userTO) {
     User userRegistered = userService.registerUser(userTO);

     UserTO userRegisteredTO = new UserTO();

     userRegisteredTO.setUsername(userRegistered.getUsername());
     userRegisteredTO.setRole(userRegistered.getRole().name());

     return Response.ok(userRegisteredTO).build();
    }
}
```

Request
POST /users HTTP/1.1
Content-Type: application/json

```
{
  "username": "fuubar"
}
```

Response
Content-Type: application/json

```
{
  "username": "fuubar",
  "role": "MEMBER"
}
```

JAX-RS in code

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

```java
package ch.heigvd.ptl.jee.sample.rest;

import ch.heigvd.ptl.jee.sample.model.User;
import ch.heigvd.ptl.jee.sample.service.UserService;
import ch.heigvd.ptl.jee.sample.to.UserTO;
import javax.ejb.EJB;
import javax.ws.rs.Consumes;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

@Path("/users")
public class UserResource {
    @EJB
    private UserService userService;

    @POST
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
    public Response register(UserTO userTO) {
     User userRegistered = userService.registerUser(userTO);

     UserTO userRegisteredTO = new UserTO();

     userRegisteredTO.setUsername(userRegistered.getUsername());
     userRegisteredTO.setRole(userRegistered.getRole().name());

     return Response.ok(userRegisteredTO).build();
    }
}
```

Magic happens there between **JAX-RS** (and the **Jersey** implementation with **Jackson**) and the container.

Once everything is correctly configured, serialization and deserialization is mainly done automatically.

JAX-RS in code

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

```java
package ch.heigvd.ptl.jee.sample.rest;

import ch.heigvd.ptl.jee.sample.model.User;
import ch.heigvd.ptl.jee.sample.service.UserService;
import ch.heigvd.ptl.jee.sample.to.UserTO;
import javax.ejb.EJB;
import javax.ws.rs.Consumes;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

@Path("/users")
public class UserResource {
    @EJB
    private UserService userService;

    @POST
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
    public Response register(UserTO userTO) {
     User userRegistered = userService.registerUser(userTO);

     UserTO userRegisteredTO = new UserTO();

     userRegisteredTO.setUsername(userRegistered.getUsername());
     userRegisteredTO.setRole(userRegistered.getRole().name());

     return Response.ok(userRegisteredTO).build();
    }
}
```

Magic happens there between **JAX-RS** (and the **Jersey** implementation with **Jackson**) and the container.

Once everything is correctly configured, serialization and deserialization is mainly done automatically.

Same for the response.

JAX-RS Plumbing

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

```
package ch.heigvd.ptl.jee.sample.rest;

import java.util.HashSet;
import java.util.Set;
import javax.ws.rs.ApplicationPath;
import javax.ws.rs.core.Application;

@ApplicationPath("/api")
public class ApiApplication extends Application {
   @Override
   public Set<Class<?>> getClasses() {
    Set<Class<?>> classes = new HashSet<>();

    classes.add(UserResource.class);

    return classes;
   }
}
```

JAX-RS Plumbing

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

```
package ch.heigvd.ptl.jee.sample.rest;

import java.util.HashSet;
import java.util.Set;
import javax.ws.rs.ApplicationPath;
import javax.ws.rs.core.Application;

@ApplicationPath("/api")
public class ApiApplication extends Application {
   @Override
   public Set<Class<?>> getClasses() {
    Set<Class<?>> classes = new HashSet<>();

    classes.add(UserResource.class);

    return classes;
   }
}
```

JAX-RS force us to inherits from an Application class provided.

JAX-RS Plumbing

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

Well, we define the base path for the all the resource that will be managed by this application class.

In this our example, the user resource will finally be available under /api/users.

JAX-RS force us to inherits from an Application class provided.

```
heigvd.ptl.jee.sample.rest;

util.HashSet;
util.Set;
x.ws.rs.ApplicationPath;
x.ws.rs.core.Application;

@ApplicationPath("/api")
public class ApiApplication extends Application {
   @Override
   public Set<Class<?>> getClasses() {
    Set<Class<?>> classes = new HashSet<>();

    classes.add(UserResource.class);

    return classes;
   }
}
```

JAX-RS Plumbing

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

Well, we define the base path for the all the resource that will be managed by this application class.

In this our example, the user resource will finally be available under /api/users.

JAX-RS force us to inherits from an Application class provided.

```
heigvd.ptl.jee.sample.rest;

util.HashSet;
util.Set;
x.ws.rs.ApplicationPath;
x.ws.rs.core.Application;

@ApplicationPath("/api")
public class ApiApplication extends Application {
   @Override
   public Set<Class<?>> getClasses() {
    Set<Class<?>> classes = new HashSet<>();

    classes.add(UserResource.class);
```

We simply add the user resource class to the set of classes managed by this application.

Therefore, you can imagine having two different REST applications using the same resources.

REMARK: HTTP 404 -> Resource class probably missing there!