Introduction to Java EE

Mobile Web Services

Open Source Frameworks (OSF)
Master of Science in Engineering (MSE)
Olivier Liechti
olivier.liechti@heig-vd.ch

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

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

The Notion of Systemic Quality

The notion of systemic quality

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

- How do we specify the **requirements** of a system?
  - We always build systems to deliver some sort of "**service**" (web, messaging, access control, customer management, etc.).
  - We first have to specify “**what**” the system should do. In other words, we have to specify **functional requirements**.
    - We also have to specify “**how**” the system should behave, i.e. what qualities it should exhibit. These are the **non-functional requirements**.
- Non-functional requirements characterize **systemic qualities, or “-ilities”**:
  - There are lots of different systemic qualities. Depending on the system, some are more important than others.
  - Choices made when defining the system architecture have a large impact on the systemic qualities.
  - Your life as an architect will be to deal with **trade-offs** in addressing conflicting systemic qualities.

Example

System
Vehicle

Functional requirements
Move people around

Non-functional requirements
Performance
Capacity
Reliability
Cost
Aestethics
Ease of use

Different systemic qualities often create **opposing forces**.

Defining the “right” architecture for a system means finding the right **balance** between these forces.

http://flickr.com/photos/jamesjordan/2652074635/sizes/o/#cc_license

Some systemic qualities...

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

- **Response time**
  - Measures the time required to present a result to the user
  - Important for the end-user
- **Throughput**
  - Measures the number of requests that can be processed in a given time frame
  - Important for the service provier
- **Scalability**
  - Measures how easy/costly it is to adapt the system in order to handle additional load
  - Ideally: linear scalability. "2 x more users => 2 x more servers”
- **Availability**
  - Measures the percentage of time during which the system can be used
  - 99% availability = average unavailability of 3.65 jours per year, i.e 1 hour and 41 minutes per week.

http://en.wikipedia.org/wiki/Ilities

Some systemic qualities...

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

- **Reliability**
  - Mesure la capacité du système à "remplir sa fonction" sur une période.
  - Un système peut avoir une fiabilité plus élevée que celle d'un de ces sous-système (notamment grâce à la redondance).
  - Mesure exprimée en "temps moyen entre 2 pannes" (Mean Time Between Failures).
- **Evolutivity**
  - Mesure la facilité avec laquelle le système pourra être adapté pour satisfaire à de nouvelles exigences (fonctionnelles et non-fonctionnelles!).
- **Security**
  - Mesure la capacité du système à empêcher une utilisation abusive.
- **Manageability**
  - Mesure la facilité avec laquelle l'état du système pourra être surveillé, les pannes détectées, des opérations de maintenance réalisées, etc.

http://en.wikipedia.org/wiki/Ilities

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

Java Enterprise Edition (Java EE)

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

J2EE or Java EE?

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

- Java Enterprise Edition is **not** a recent specification.
- The SDK 1.2 was published in 1999.
- The specification is managed through the JCP since version 1.3
- We used to talk about "Java 2 Enterprise Edition 1.3", or J2EE 1.3
- We then moved from J2EE 1.3 to J2EE 1.4 to... **Java EE 5**
- Today, we should speak of Java EE, but J2EE is still sometimes used...
- Today, most application servers implement Java EE 6
- The early draft 2 for Java EE 7 has been submitted in November 2012; final spec expected for Q2 2013

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

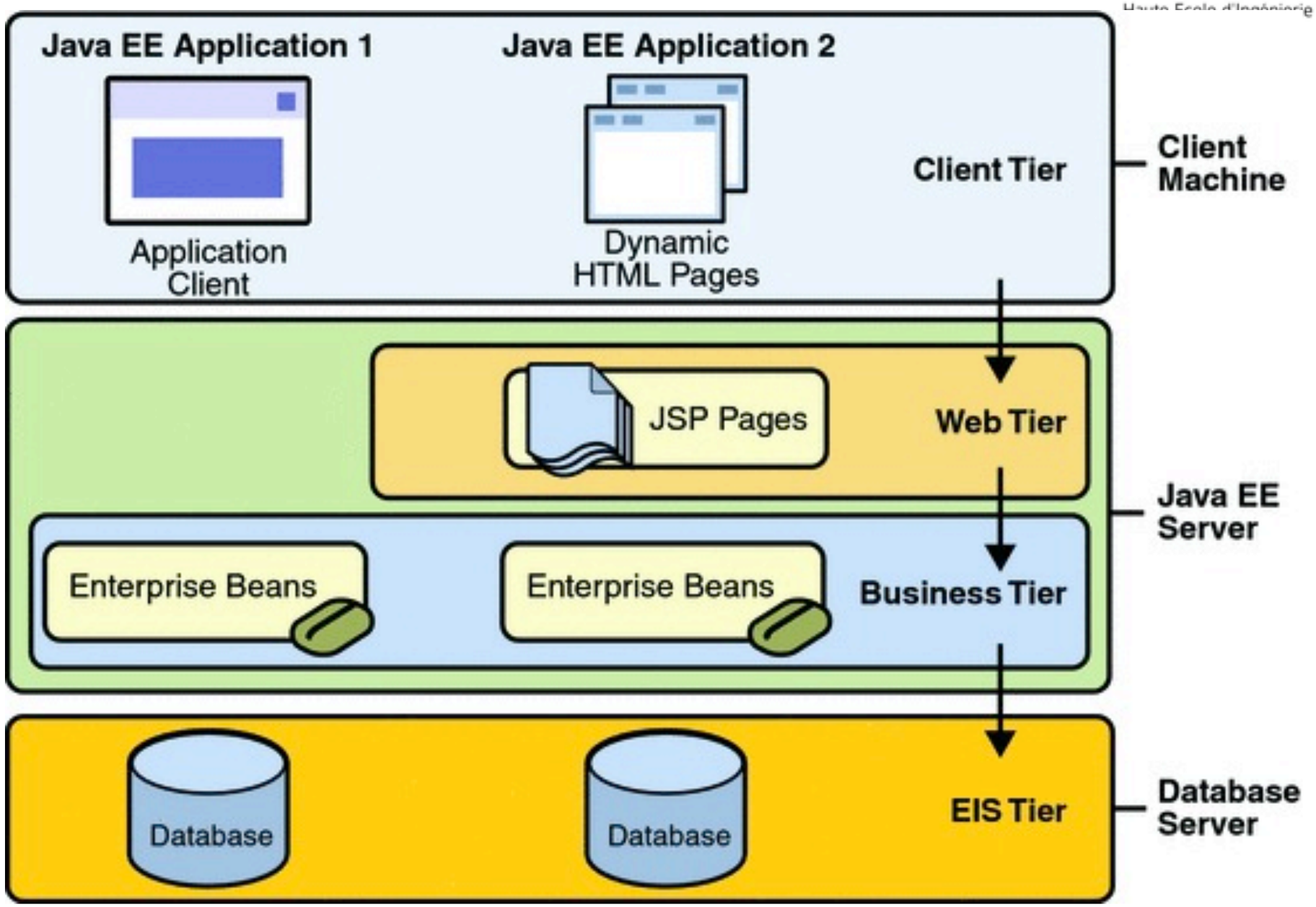
BEA Systems
Chandrasekaran, Muralidharan
E.piphany, Inc.
IBM
Leme, Felipe
OW2
Raible, Matt
SAP AG
Sun Microsystems, Inc.
Tmax Soft, Inc.

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

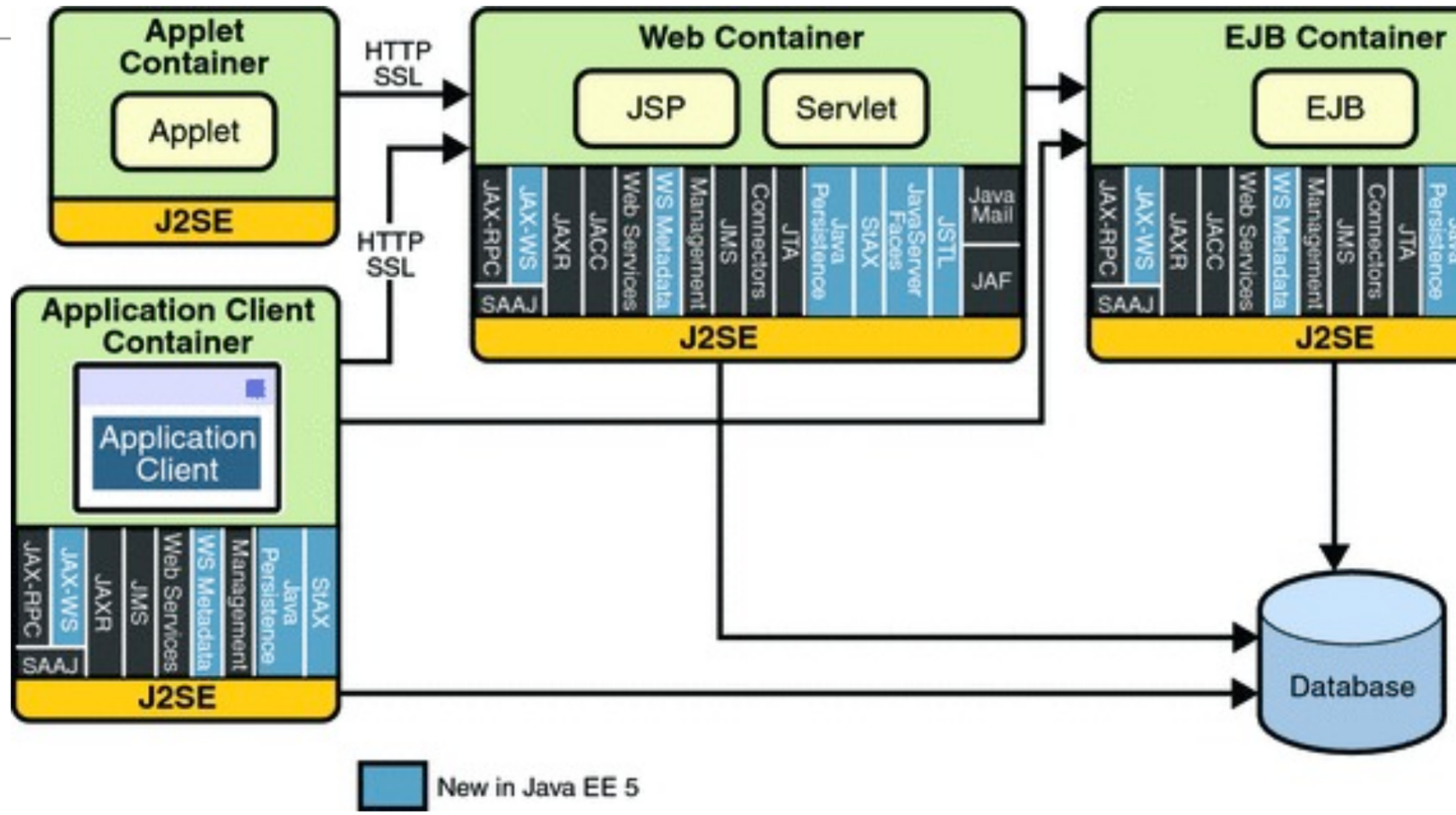
- The software that implements the Java EE specification is called an "**application server**"
  - There are **open source** and **proprietary** application servers.
  - Glassfish, JBoss, WebSphere, BEA WebLogic are examples of application servers.
    - Editors compete on aspects that are not defined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



http://java.sun.com/javaee/5/docs/tutorial/doc/bnaay.html

Java EE 5 APIs

- Enterprise JavaBeans Technology
- Java Servlet Technology
- JavaServer Pages Technology
- JavaServer Pages Standard Tag Library
- JavaServer Faces
- Java Message Service API
- Java Transaction API
- JavaMail API
- JavaBeans Activation Framework
- Java API for XML Processing
- Java API for XML Web Services (JAX-WS)
- Java Architecture for XML Binding (JAXB)
- SOAP with Attachments API for Java
- Java API for XML Registries
- J2EE Connector Architecture
- Java Database Connectivity API
- Java Persistence API
- Java Naming and Directory Interface
- Java Authentication and Authorization Service
- Simplified Systems Integration



http://java.sun.com/javaee/5/docs/tutorial/doc/bnacj.html

Components, packaging & deployment

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

- The first version of J2EE put a lot of emphasis on the separation of roles:
  - **Component developer** builds self-contained, reusable **components** and focuses on business logic (“Here is a customer management brick”, “Here is a pricing brick”).
  - **Application assemblers** create **applications** by gluing together different components (“I connect the customer management and the pricing bricks in order to have an insurance application”).
  - **Deployers** have to take applications and to put them into production. Every application needs resources (DBs, user repositories, etc.). The deployer has to connect the application to the resources (declaratively!!).
- We also want to be able to manage the life-cycle of applications and to have control on the release of new versions. It really helps to create **packages**!

Components, packaging & deployment

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

- Different types of components, different types of packages/archives:
  - Java libraries are useful in enterprise applications; we can bundle classes into **traditional .jar files (Java ARchives)** and integrate them in our applications.
  - Some applications do not require all the features provided by the Java EE platform; they are “lightweight” can live entirely in the web container. We package these applications in **.war files (Web ARchives)**.
  - Business services can be packaged and deployed independently from a presentation front-end. We can package **EJBs in .jar files**.
  - Full-blown Java EE applications bundle together web-tier components, service components, libraries. A Java EE application is packaged in a **.ear file (Enterprise ARchive)**. The .ear file contains a .war files and one or more .jar files.

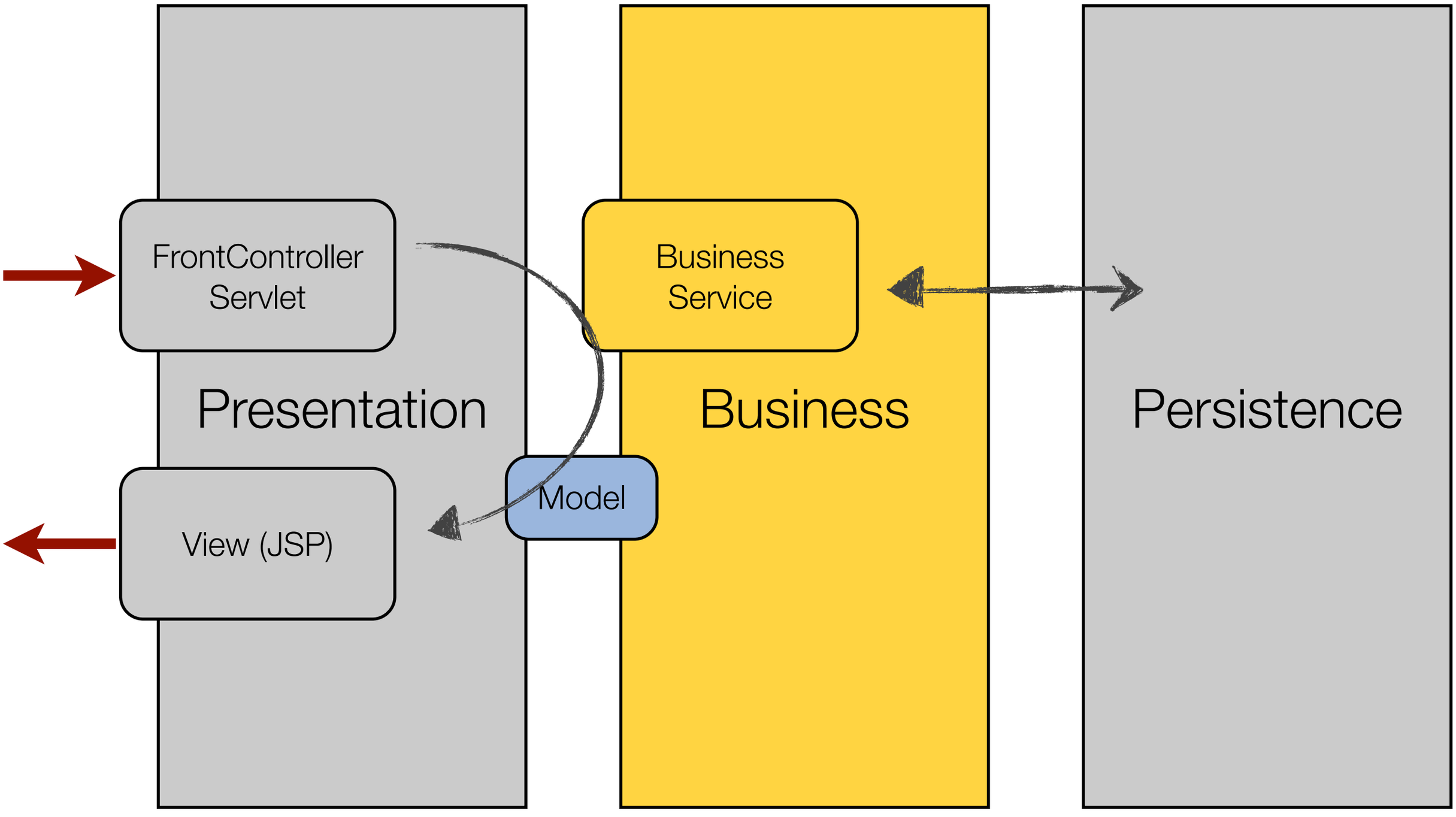
Content of archives

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

- In .war, .jar and .ear files, we find:
  - **Compiled classes and libraries**
  - **Resources** (images, property files, configuration files, etc.)
  - **Deployment descriptors**
- Deployment descriptors make it possible to alter the system-level behavior of the application **declaratively**, at deployment time:
  - Change security mappings, change transaction isolation level, change database connection parameters, change URL mappings, etc.
- In J2EE, deployment descriptors were mandatory and quite a pain to write (IDE added support over time, and XDoclet introduced a way to generate them automatically).
- In Java EE, deployment descriptors are still available but are largely replaced by @annotations. In Java EE 5, annotations are generally used for EJBs. The web.xml deployment descriptor is still used.

The Business Tier

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



Agenda

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

- Containers & components
- Enterprise Java Beans (EJB)
  - Stateless and stateful session beans (SLSB and SFSB)
  - Message-driven beans (MDB / JMS)
- Resource pooling
- Life cycle
- Getting a reference to the service: lookup vs. dependency injection

Containers and components

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

- There are **different ways** to design, build and deploy the “business service” components.
- In the **simplest scenario**, think of a plain vanilla web application, where you create a “controllers”, a “services” and a “model” packages. Here, you implement services as POJOs and **do everything yourself**:
  - Instantiation of services, concurrency control, transactions, security, etc.
- When the application/system has some complexity (because of the functionality, because of the volume of transactions, because of the integration, etc.), it becomes interesting to use **“managed” components**.
- The idea is to let “something” in the **infrastructure** manage the life cycle of your service components. That “something” can take different shapes.
- In Java EE, that “something” is the Enterprise Java Bean **container**.

Containers and components

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

- The **EJB container** is one sub-system of the Java EE container.
- It provides a **runtime environment** for the business services, implemented as Enterprise Java Beans.
- The **developer** knows that services will be “available” to its components, once they are deployed in the container. He can thus use these services through standard **APIs**.
- The **production manager** configures the EJB container and tunes it in different ways (this is implementation specific).
- With **some application servers**, the EJB container can be **distributed** over several physical nodes (for scalability and availability purposes). This distribution is **transparent to the clients** - they sill only see “one runtime environment”.

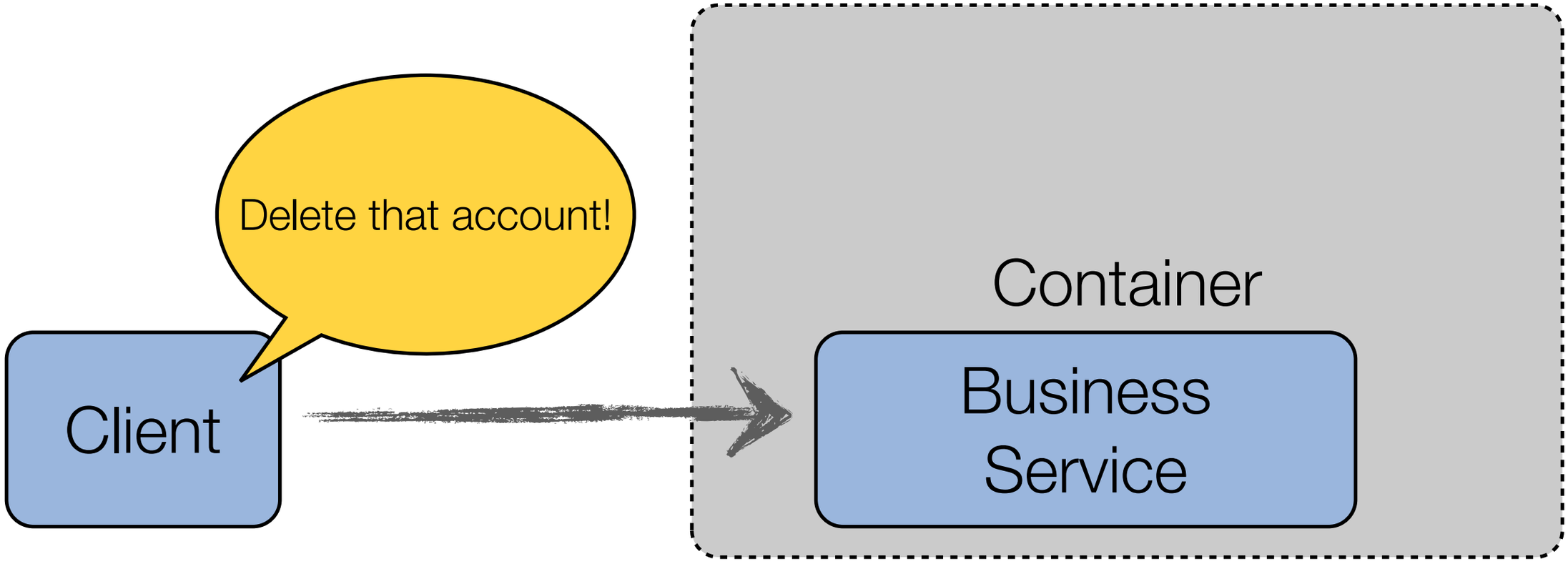
From the specs: what are EJBs (v3)

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

- "An enterprise bean typically contains **business logic** that operates on the **enterprise's data**.
- An enterprise bean's instances are **managed at runtime by a container**.
- An enterprise bean can be **customized at deployment time** by editing its environment entries.
- Various service information, such as **transaction** and **security** attributes, may be specified **together** with the business logic of the enterprise bean class in the form of **metadata annotations**, or **separately**, in an **XML deployment descriptor**. This service information may be extracted and managed by tools during application assembly and deployment.
- Client **access is mediated by the container** in which the enterprise bean is deployed."

Mediated access

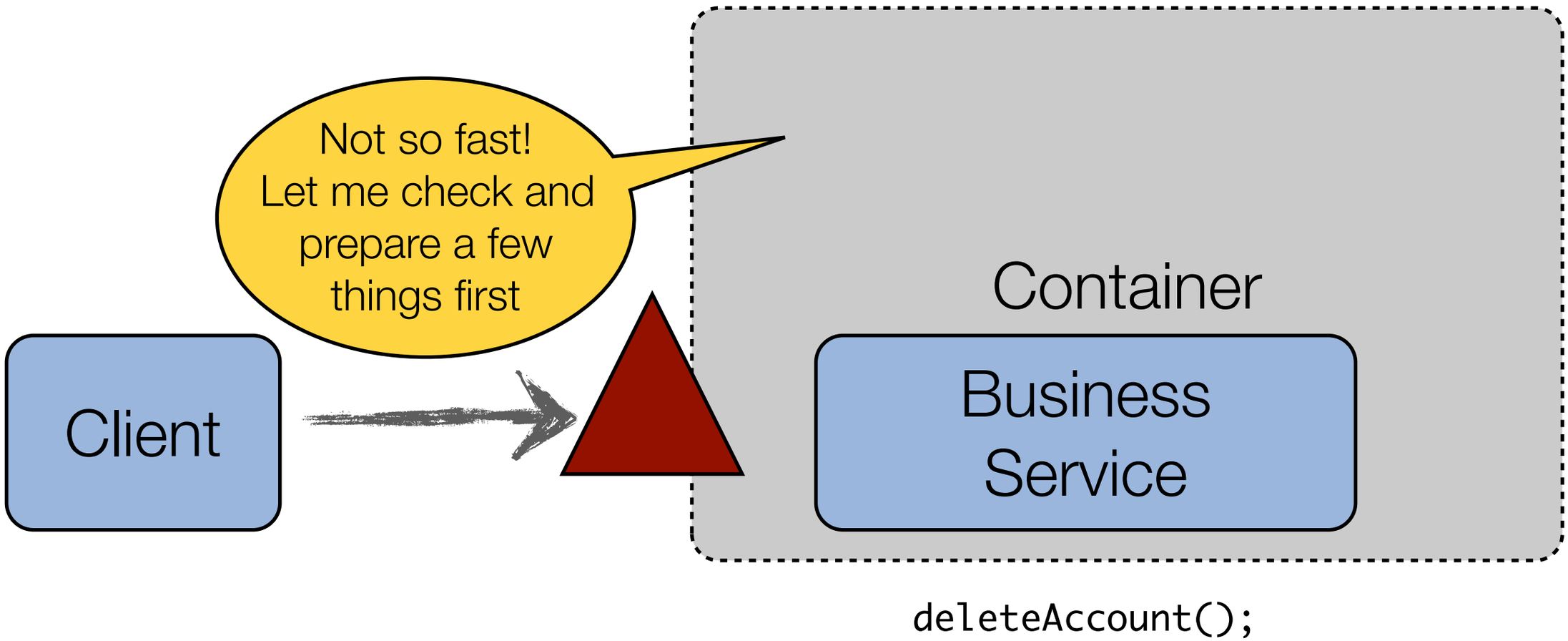
heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud




```
deleteAccount();
```

Mediated access

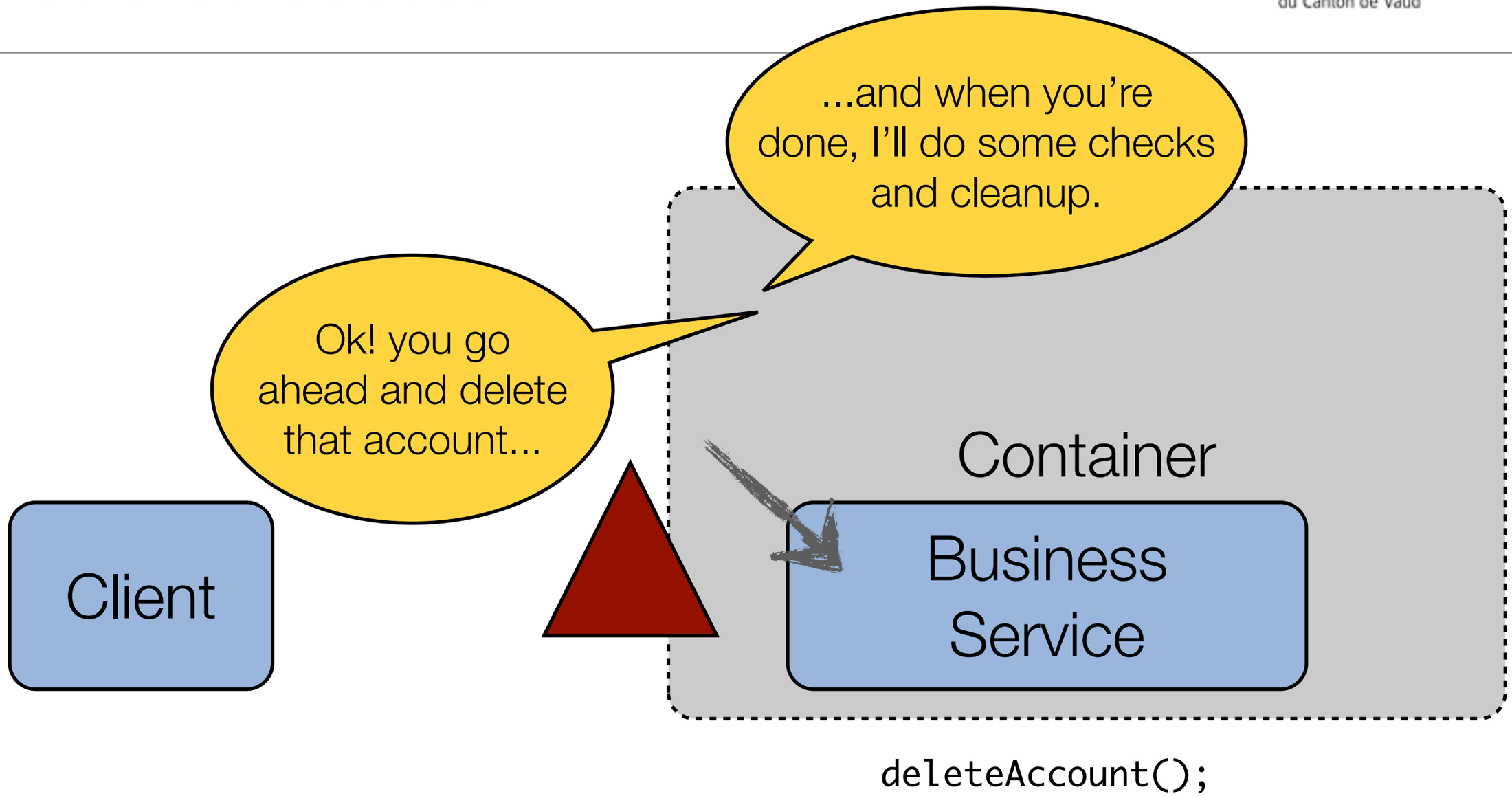
heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud




```
deleteAccount();
```

Mediated access

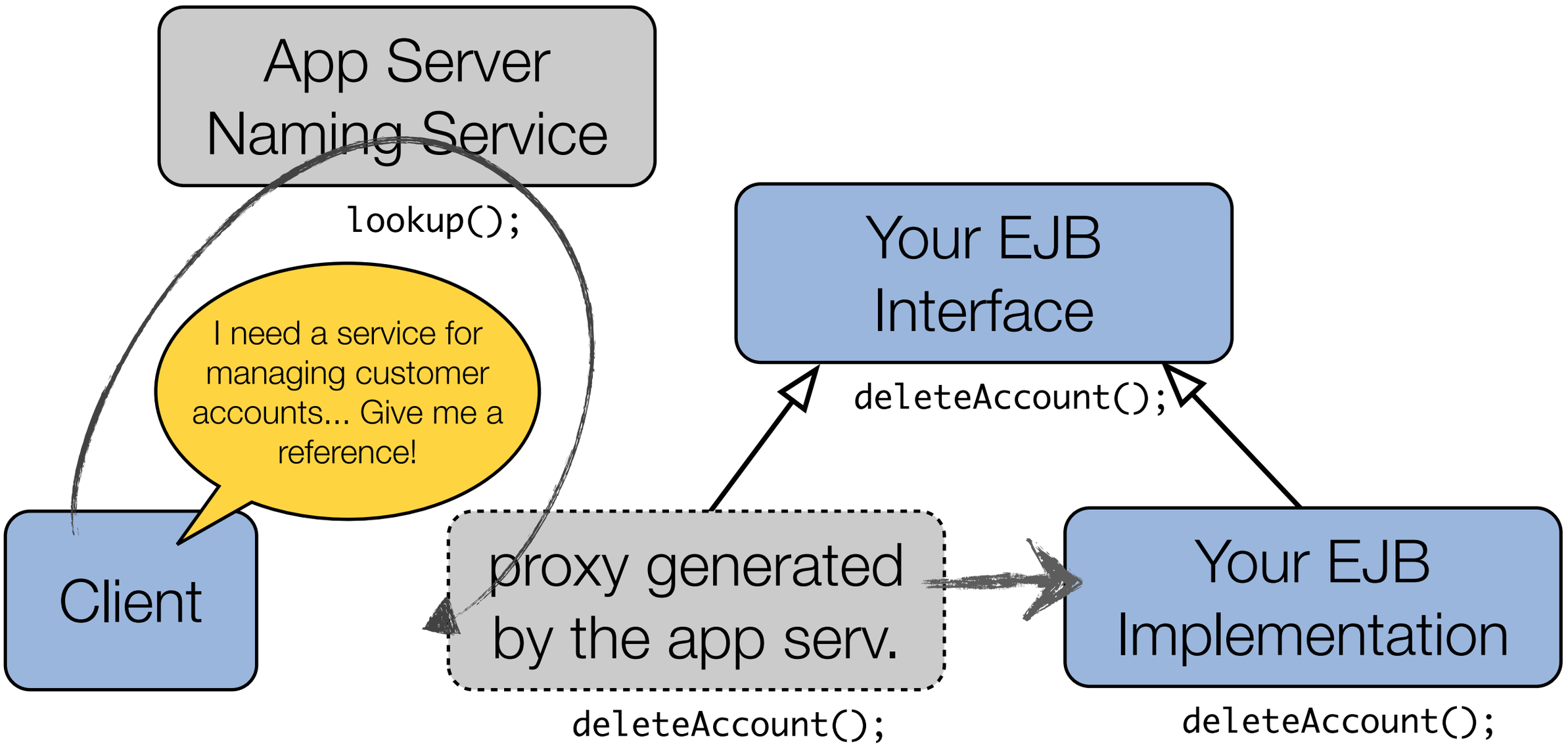
heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud




```
deleteAccount();
```

Mediated access

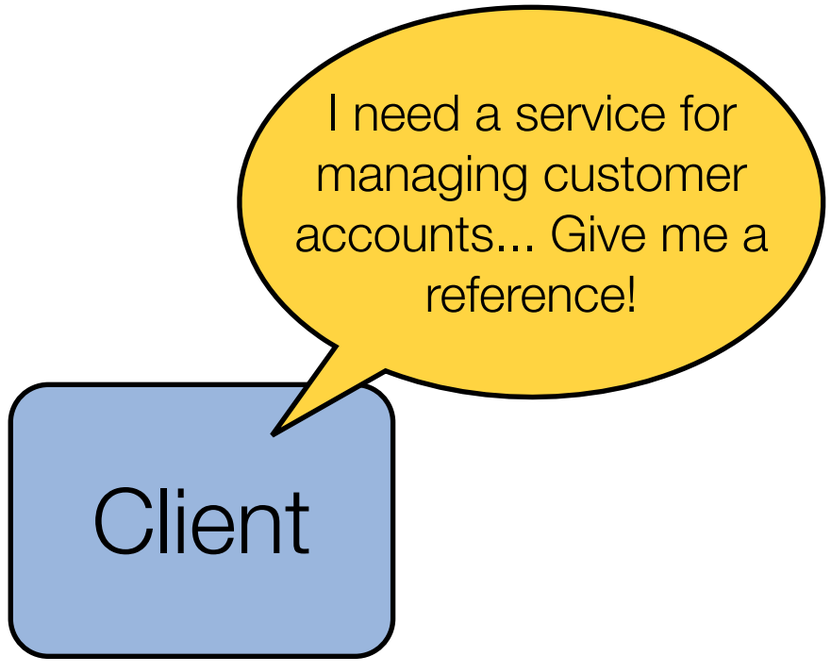
heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

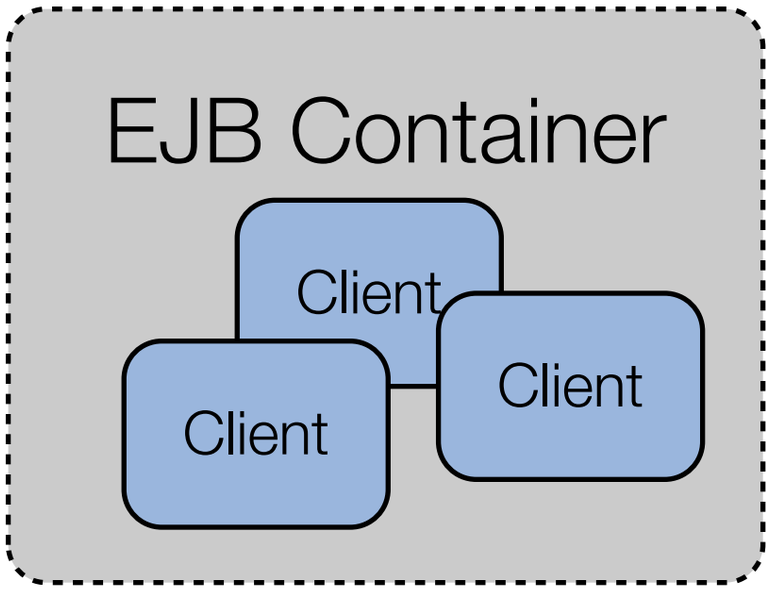


Resource pooling

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

- As a developer, you define a **service interface** and provide an **implementation**.
- **You do not instantiate the service** - the application server takes care of that.
- The container does not instantiate only one instance of a class implementing your interface; **it creates several instances that are placed into a pool**.
- When a client comes along, a **free instance** is picked from the pool and **dedicated** to the client (no concurrency issue).
- In production, you can tune the size of pools (tradeoff between processing **bandwidth** and resource **consumption**).





From the specs: component types

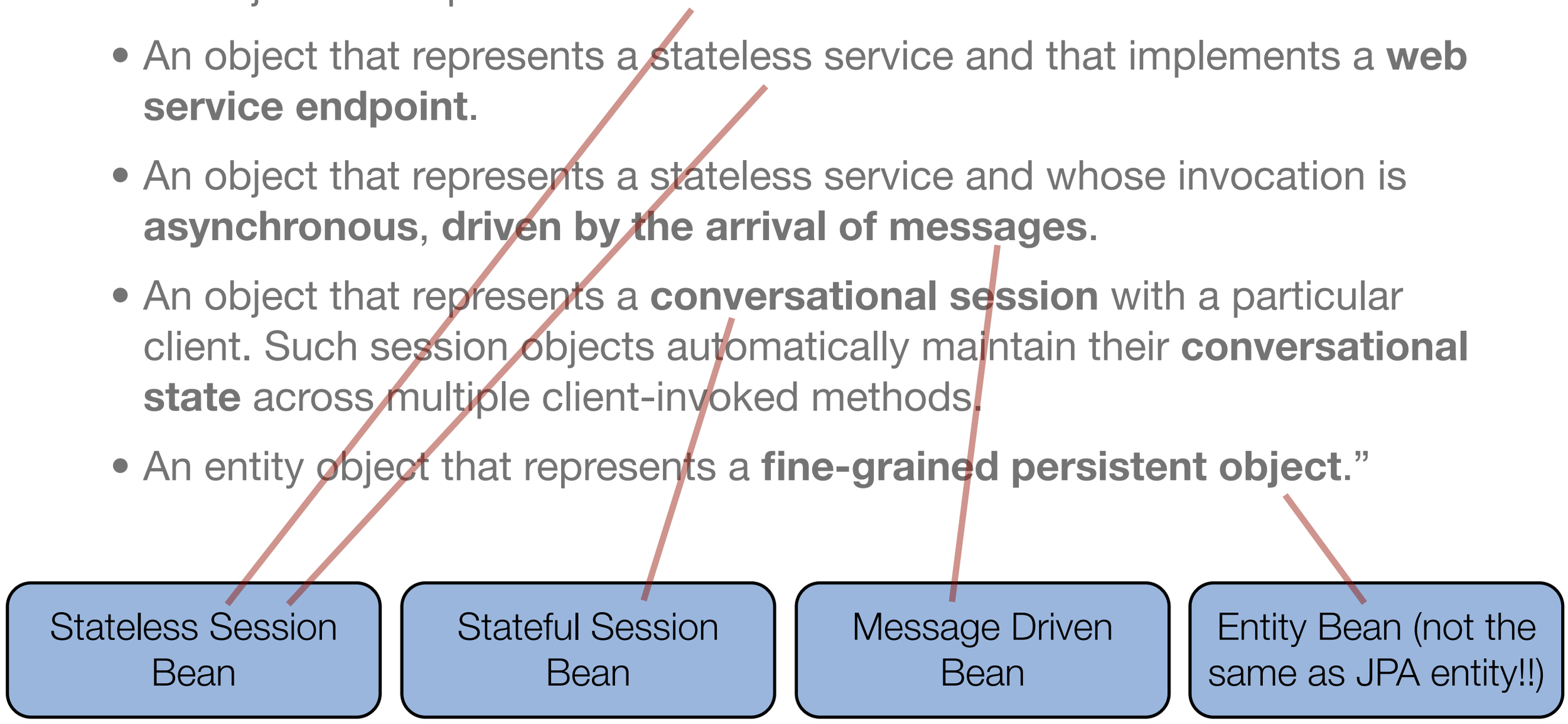
heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- “The enterprise bean architecture is flexible enough to implement the following:
  - An object that represents a **stateless service**.
  - An object that represents a stateless service and that implements a **web service endpoint**.
  - An object that represents a stateless service and whose invocation is **asynchronous**, **driven by the arrival of messages**.
  - An object that represents a **conversational session** with a particular client. Such session objects automatically maintain their **conversational state** across multiple client-invoked methods.
  - An entity object that represents a **fine-grained persistent object**.”

From the specs: component types

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

- "The enterprise bean architecture is flexible enough to implement the following:
  - An object that represents a **stateless service**.
  - An object that represents a stateless service and that implements a **web service endpoint**.
  - An object that represents a stateless service and whose invocation is **asynchronous**, **driven by the arrival of messages**.
  - An object that represents a **conversational session** with a particular client. Such session objects automatically maintain their **conversational state** across multiple client-invoked methods.
  - An entity object that represents a **fine-grained persistent object**."



Session Beans (EJB)

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

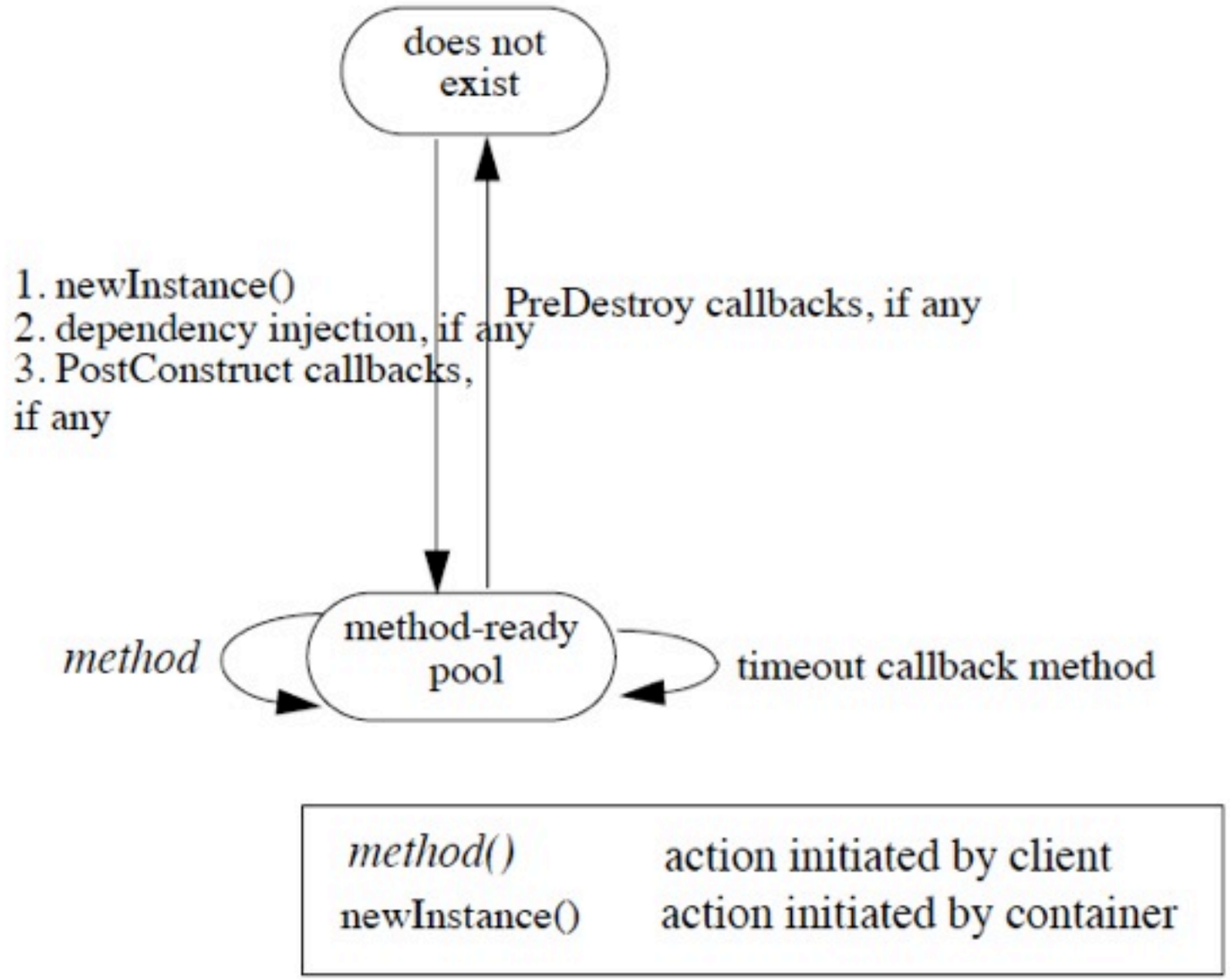
- A session bean may be either:
  - **stateless**—the session bean instances contain no conversational state between methods; **any instance** can be used for any client.
  - **stateful**—the session bean instances contain **conversational state** which must be **retained** across methods and transactions.

- Notes:
  - The term "**stateless**" signifies that an instance has no state **for a specific client**. However, the instance variables of the instance **can** contain the state across client-invoked method calls. Examples of such state include an open database connection and an object reference to an enterprise bean object.
  - The lifetime of stateful session bean is controlled by the client (the client call myStatefulService.remove())

Stateless Session Bean Life Cycle

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

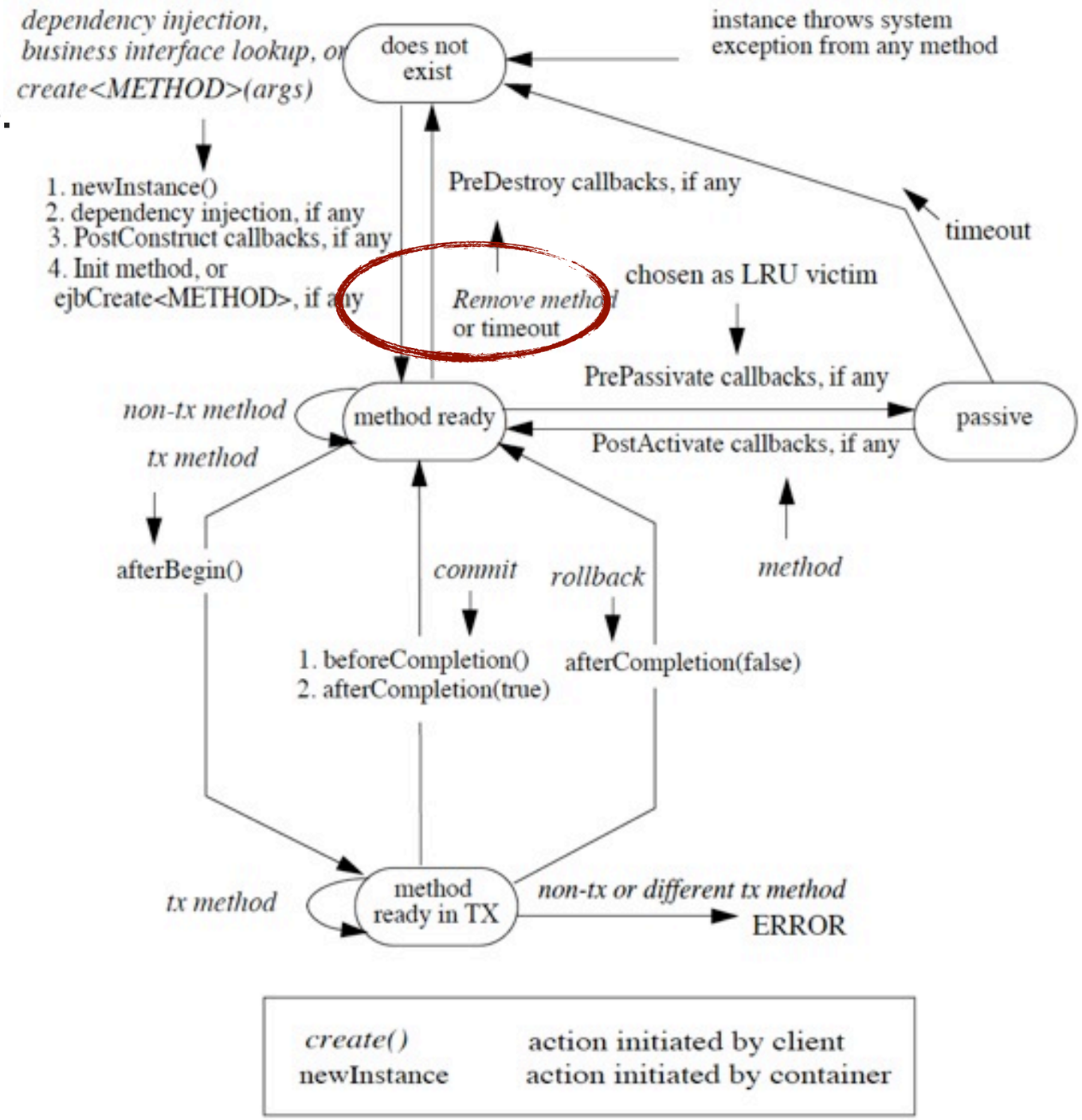
- The container manages the life cycle of the beans.
- The developer may implement callback methods, invoked when the state changes (e.g. for init/ clean-up tasks).
- With EJB3, you can do that with method annotations.
- Remember that you don't know how many instances there are, you don't know which one you're talking to.



Stateful Session Bean Lifecycle

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

- Things are more complicated here, because we manage client-specific state.
- What happens when we run out of memory? We need to move the state of "some beans" to secondary memory.
- This is called "passivation".
- When a client comes back and wants to invoke a method on a passivated bean, we need to bring it back to life.
- This is called "activation".
- You typically "remove" a bean at the end of the use case (conversation).
- Check out the JBoss Seam Framework for an architecture that makes use of SFSB.



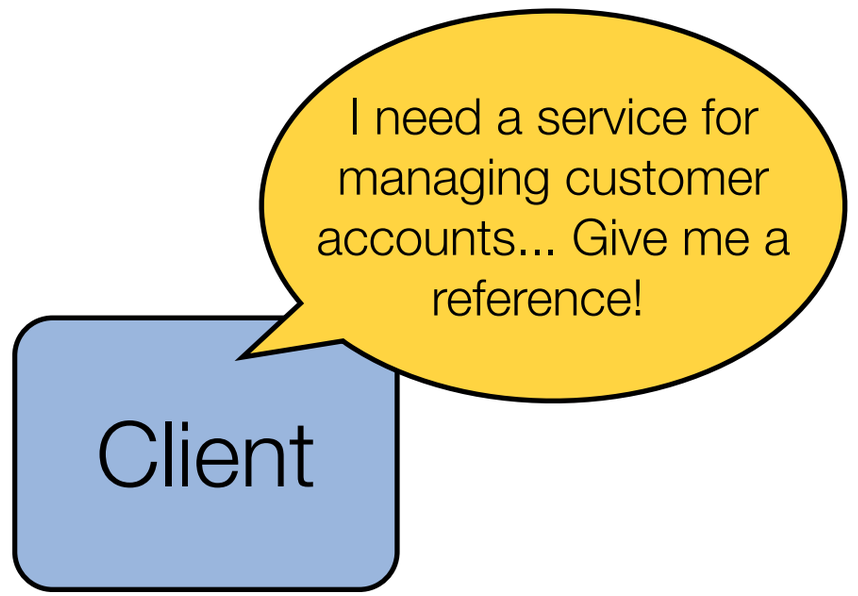
The client view: reaching the service

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

- Before using a service, the client needs to obtain a **reference** to a component that implements th service.
- Two approaches: **lookup** and **dependency injection**
- From the original J2EE model, we can do a lookup - the application server provides a **naming service** through JNDI.
- **Dependency injection** was introduced in the EJB3 / Java EE 5 specification. Here, the app server injects a reference into the client.

```
InitialContext ic = new InitialContext();
CustomerService service = (CustomerService)
ic.lookup("customerService")
```

```
@EJB
CustomerService service
```



Want to try it yourself?

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

- To experiment with Java EE, you will need:
  - A **Java EE application server** (e.g. glassfish, jboss, websphere, etc.). This is mandatory.
  - An **IDE** (e.g. Netbeans, eclipse, etc.). This is not strictly mandatory, but it will help you a lot.
    - A **build management system** (e.g. maven, gradle, etc.). This is not strictly mandatory either, but if you are working on a real project, it is not even a question.
    - *The Netbeans Java EE Download bundle gives you a complete and integrated environment.*
- There is a **ton of information** available in books and on-line. A good idea is to start with the official Java EE tutorial:
  - http://docs.oracle.com/javaee/6/tutorial/doc/