Lecture 2: The Business Tier

Olivier Liechti
AMT

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

Today's agenda

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

14h00 - 15h00	60'	**Lecture** Business tier: EJBs, SLSBs, SFSBs, MDBs Inversion of Control, dependency injection JMeter
15h00 - 15h10	*10'*	*Break*
15h10 - 16h25	75'	**Demo / experiment** Servlets are not thread-safe **Lecture** EJBs and resource pooling

heig-vd

Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

Back to the MVC lab…

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

3 interesting questions

- How was the servlet **instantiated** and **by who**?
- Who **invoked** the doGet method on the servlet?
- How did the servlet get a **reference** to the JSP page?

These 3 questions are related to **3 key concepts**:
managed components, *inversion of control* and *dependency injection*

How was the servlet **instantiated**?

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

- A servlet is a **Java object**.
- In **your code**, you never wrote anything like this:

```
HttpServlet fc = new FrontControllerServlet();
```

- The servlet instance(s) is (are) created **by the application server**.
- In other words, we say that the application server is **managing** the servlet. Or that the servlet is a **managed component**.

Who **invoked** the doGet method?

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

- 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).

How did the servlet **get a ref**. to the JSP?

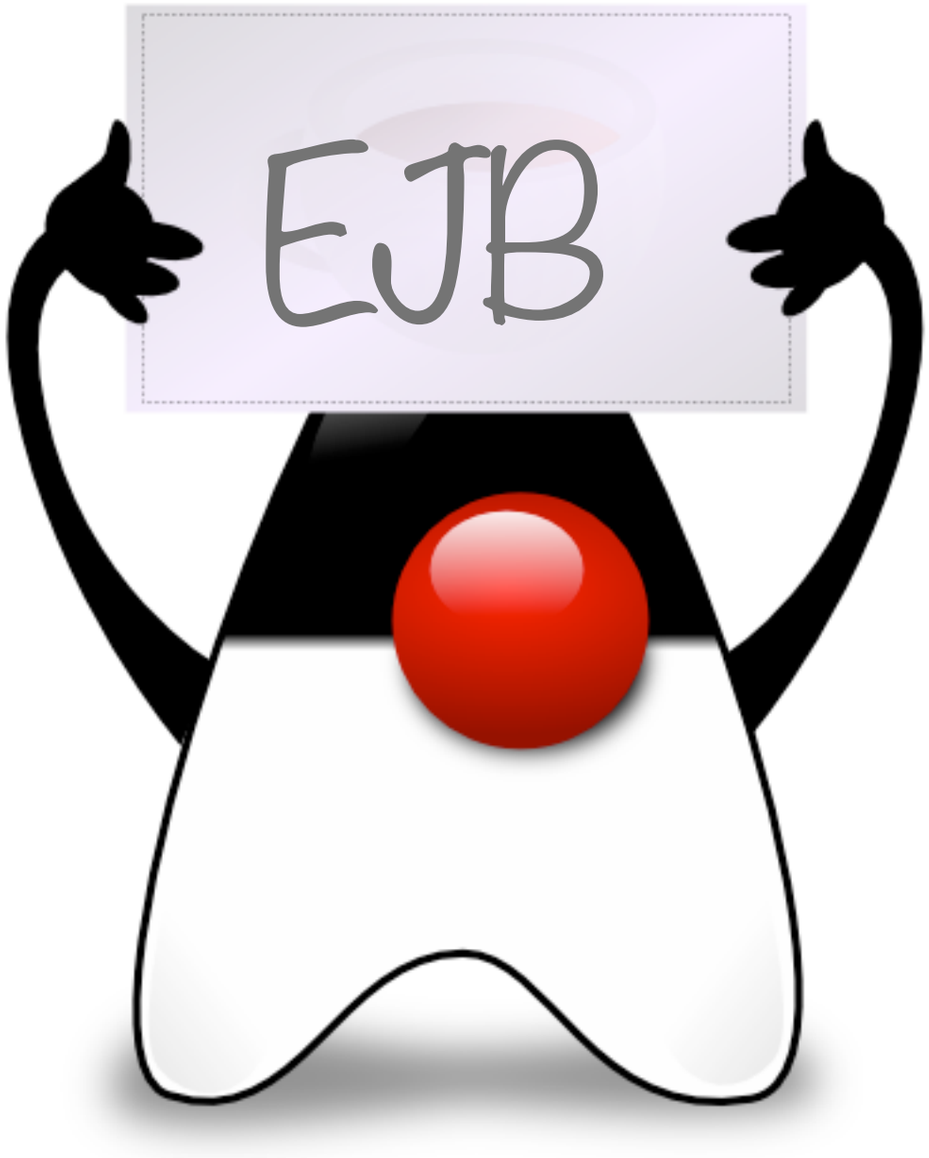
heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- In **your servlet code**, you wrote something like this:

```
request.getRequestDispatcher("/WEB-INF/views/
        measures.jsp").forward(request, response);
```

- You did a **lookup operation** to get hold of the JSP.
- In other words, **you asked the application server** to give you the reference, based on a name.
- We will see that **dependency injection** is a valuable alternative to that process.

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

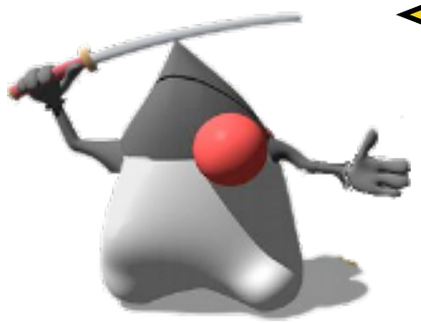


Business Services & EJB

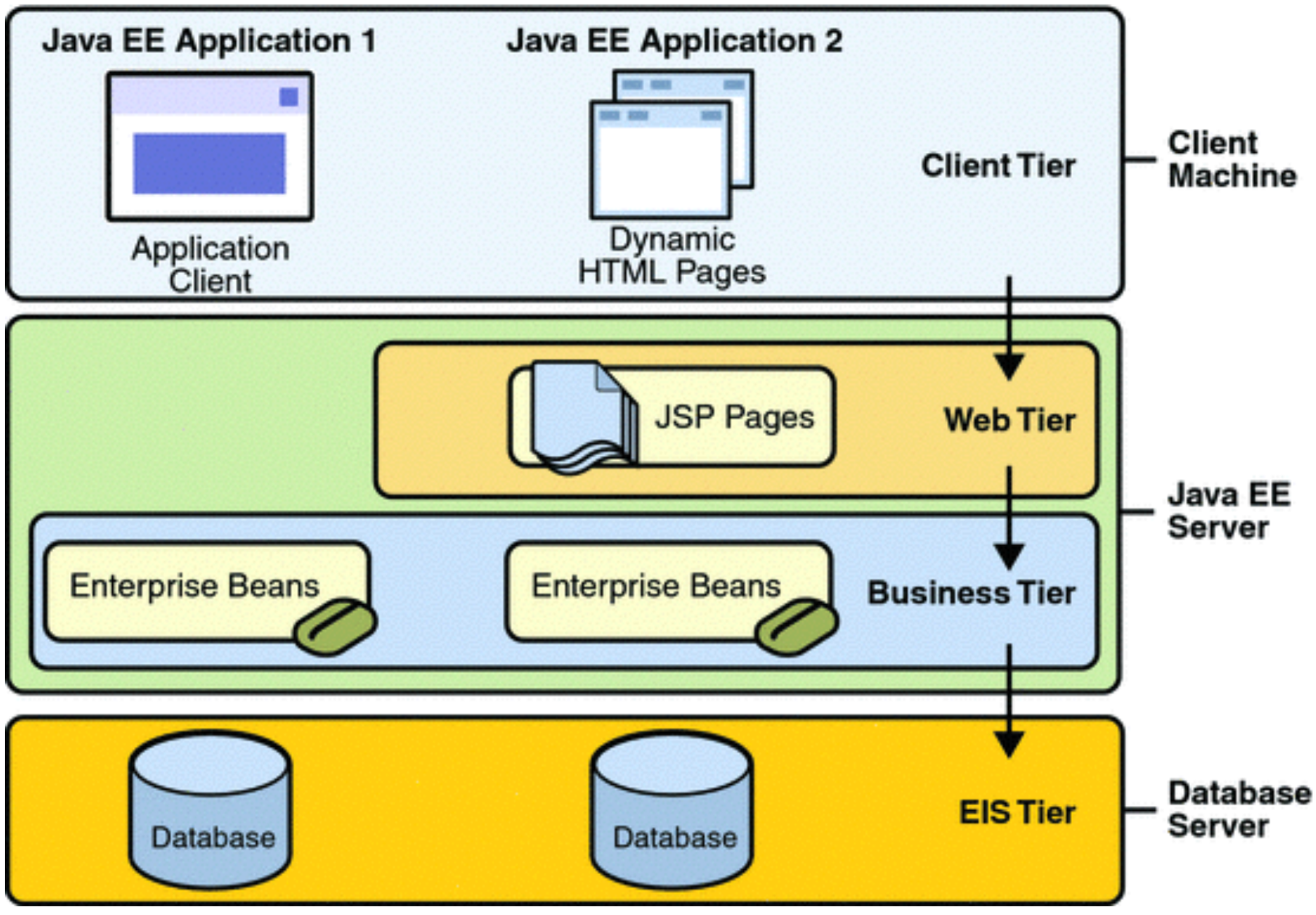
Services in a Java EE application

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

- Last week, we implemented a very simple Java EE application.
- When we implemented the MVC pattern, we implemented a service as a **Plain Old Java Object** (POJO).
- **The POJO was not a managed component**. We created the instance(s) of the service (*in the web container*).
- This week, we will see an **alternative solution** for implementing Java EE services: Enterprise Java Beans (EJBs).



What is the best way to implement services, POJOs or EJBs?
There is not a single right answer to this question! There are pros and cons in both approach.



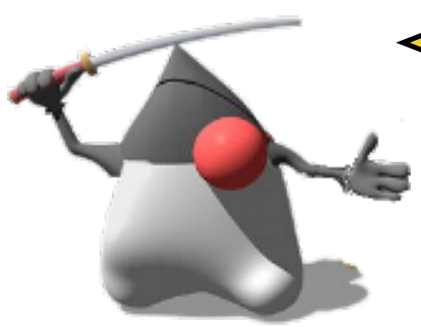
http://java.sun.com/javaee/5/docs/tutorial/doc/bnaay.html

What is an **Enterprise Java Bean** (EJB)?

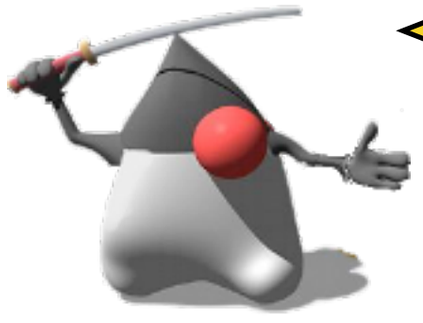
- An EJB is a **managed component**, which implements **business logic** *in a UI agnostic way*.
- The EJB container manages the **lifecycle** of the EJB instances.
- The EJB container also **mediates the access** from clients (i.e. it is an “invisible” intermediary) to EJBs.
- This allows the EJB container to perform technical operations (especially related to **transactions** and **security**) when EJBs are invoked by clients.
- The EJB container manages a **pool** of EJB instances.
- Note: the EJB 3.1 API is **specified is JSR 318**.

What are the **4 types** of EJBs used today?

- **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.

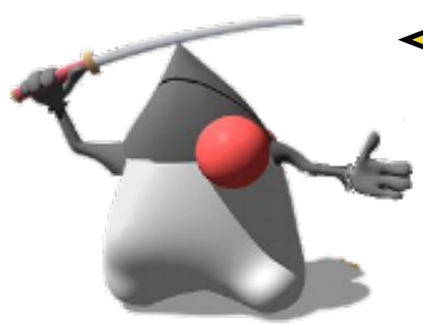


When you implement a stateful application in Java EE, **you have the choice to store the state in different places**. One option is to do it in the web tier (in the HTTP session). Another option is to use **Stateful Session Beans**. Many (most) developers use HTTP sessions.



In older versions of Java EE (before Java EE 5), there was another type of EJBs: **Entity Beans**.

Entity Beans were used for **accessing the database**. They were a nightmare to use and raised a number of issues. You might find them in legacy applications.



Entity Beans (as a legacy type of EJB) are **not the same thing** as **JPA Entities**, which are now widely used!

A first example

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

```
package ch.heigvd.amt.lab1.services;
import javax.ejb.Local;

@Local
public interface CollectorServiceLocal {

  void submitMeasure(Measure measure);

}
```

```
package ch.heigvd.amt.lab1.services;
import javax.ejb.Stateless;

@Stateless
public class CollectorService implements CollectorServiceLocal {

  @Override
  public void submitMeasure(Measure measure) {
    // do something with the measure (process, archive, etc.)
  }

}
```

These **annotations** are processed by the application server at **deployment time**.

They are an **declaration** that the service must be handled as a **managed component**!

How does a “client” find and use an EJB?

- By “**client**”, we refer to a **Java component** that wants to get a reference to the EJB and invoke its methods.
- In many cases, the client is a **servlet** or **another EJB** (i.e. a service that delegates part of the work to another service).
- The application server is providing a **naming and directory service** for managed components. Think of it as a “white pages” service that keeps track of component names and references.
- Remember that we mentioned **Dependency Injection** earlier today?

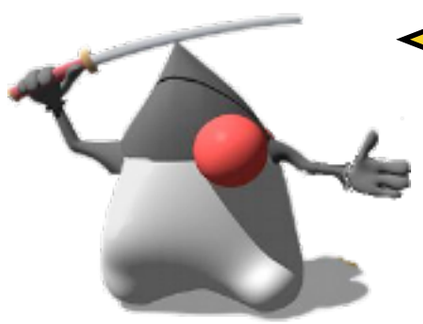
The Java Naming and Directory Interface (JNDI) provides an API to access directory services. It can be used to access an LDAP server. It can also be used to lookup components in a Java EE server.

The **first method** to find an EJB is to do an **explicit lookup**, with JNDI.

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

  private CollectorServiceLocal collectorService;

  @Override
  public void init() throws ServletException {
    super.init();
    try {
     Context ctx = new InitialContext();
      collectorService = (CollectorServiceLocal) ctx.lookup("java:module/CollectorService");
    } catch (NamingException ex) {
      Logger.getLogger(FrontController.class.getName()).log(Level.SEVERE, null, ex);
    }
  }

}
```

This gives me access to the app server's naming service

I am using the app server's naming service



Warning! These 2 JNDI operations are **costly** (performance-wise). You don't want to re-execute them for every single HTTP request!!!!
It is much better to do it once and to **cache the references** to the services.

The **second method** is to ask the app server to **inject a dependency** to the service.

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

  @EJB
  private CollectorServiceLocal collectorService;

}
```

With the @EJB annotation, **I am declaring a dependency** from between my servlet and my service. The servlet *uses* the service.

With the @EJB annotation, I am also giving instructions to the app server. The servlet and the service are **managed components**. When the app server instantiates the servlet, it **injects a value** into the **collectorService** variable.

EJBs in the MVCDemo project

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

```
@Singleton
public class BeersDataStore implements BeersDataStoreLocal {

 private final List<Beer> catalog = new LinkedList<>();

 public BeersDataStore() {
    catalog.add(new Beer("Cardinal", " Feldschlösschen", "Switzlerland", "Lager"));
    catalog.add(new Beer("Punk IPA", " BrewDog", "Scotland", "India Pale Ale"));
 }
...
}
```

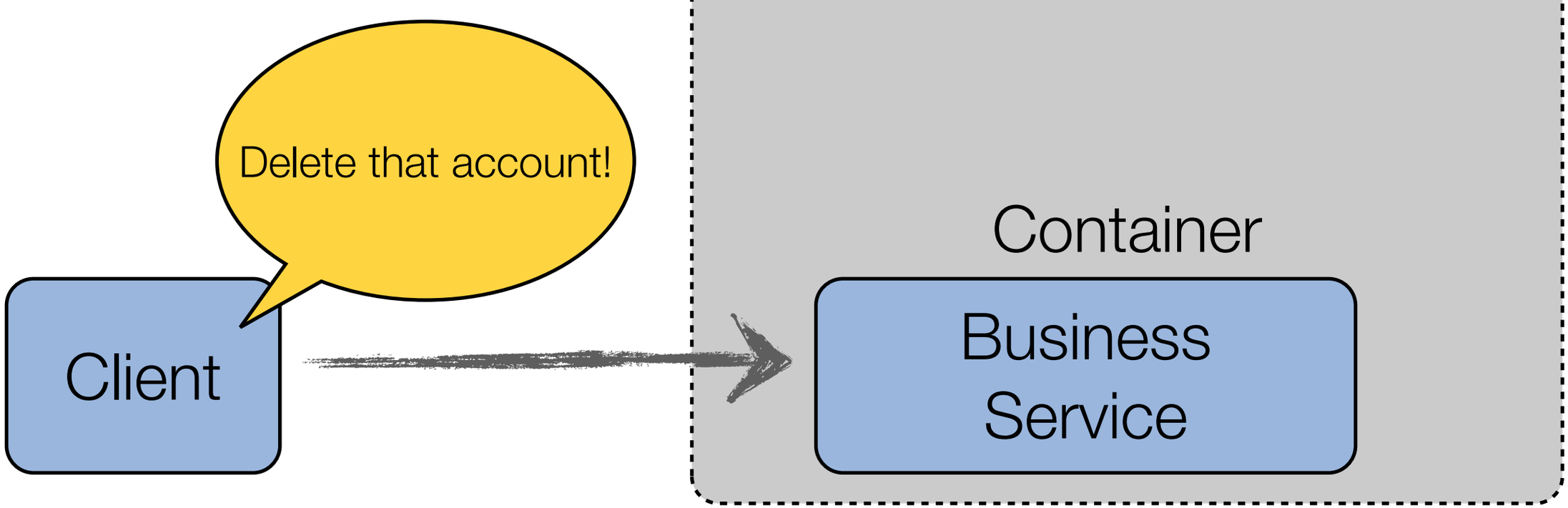
```
@Stateless
public class BeersManager implements BeersManagerLocal {

  @EJB
  BeersDataStoreLocal beersDataStore;

  @Override
  public List<Beer> getAllBeers() {
    simulateDatabaseDelay();
    return beersDataStore.getAllBeers();
  }
...
}
```

EJBs in the MVCDemo project

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

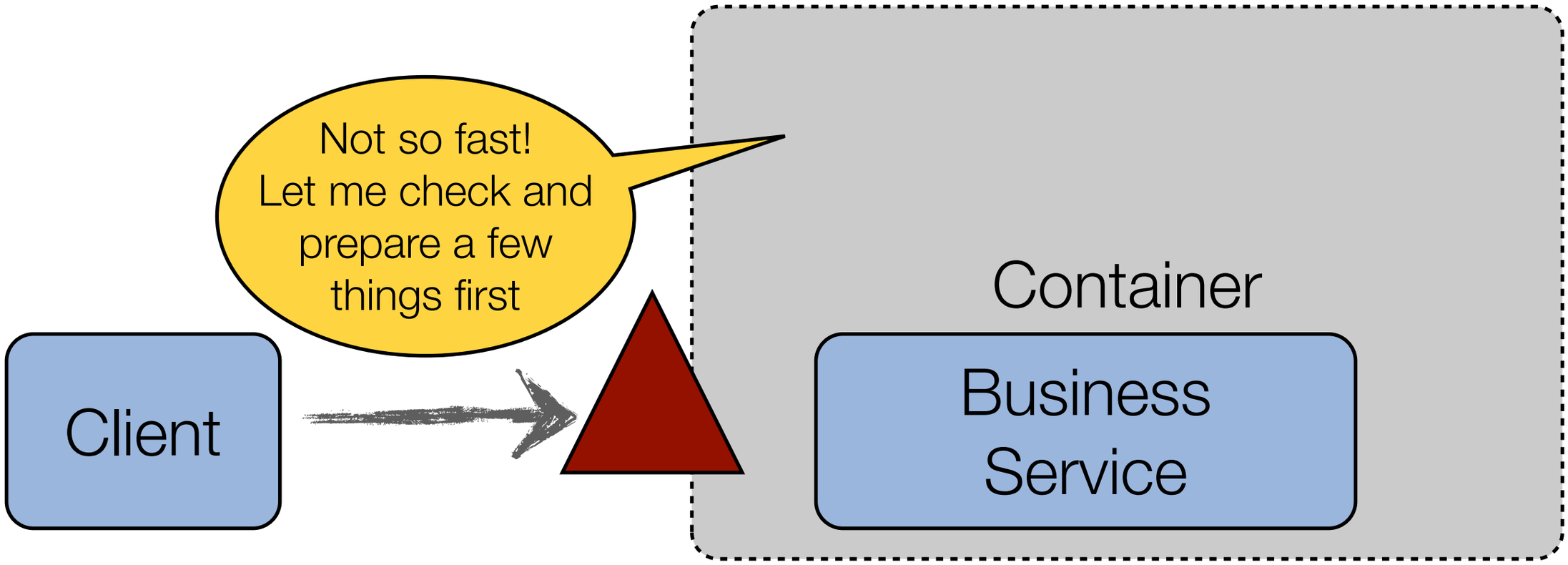
```
public class BeersServlet extends HttpServlet {

  @EJB
  BeersManagerLocal beersManager;

  @Override
  protected void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
     /*
      Firstly, we need to get a model. It is not the responsibility of the servlet
      to build the model. In other words, you should avoid to put business logic
      and database access code directly in the controller. In this example, the
      beersManager takes care of the model construction.
      */
     Object model = beersManager.getAllBeers();
     ...
   }
   ...
}
```

The app server **mediates** the access between clients and EJBs. What does it mean?



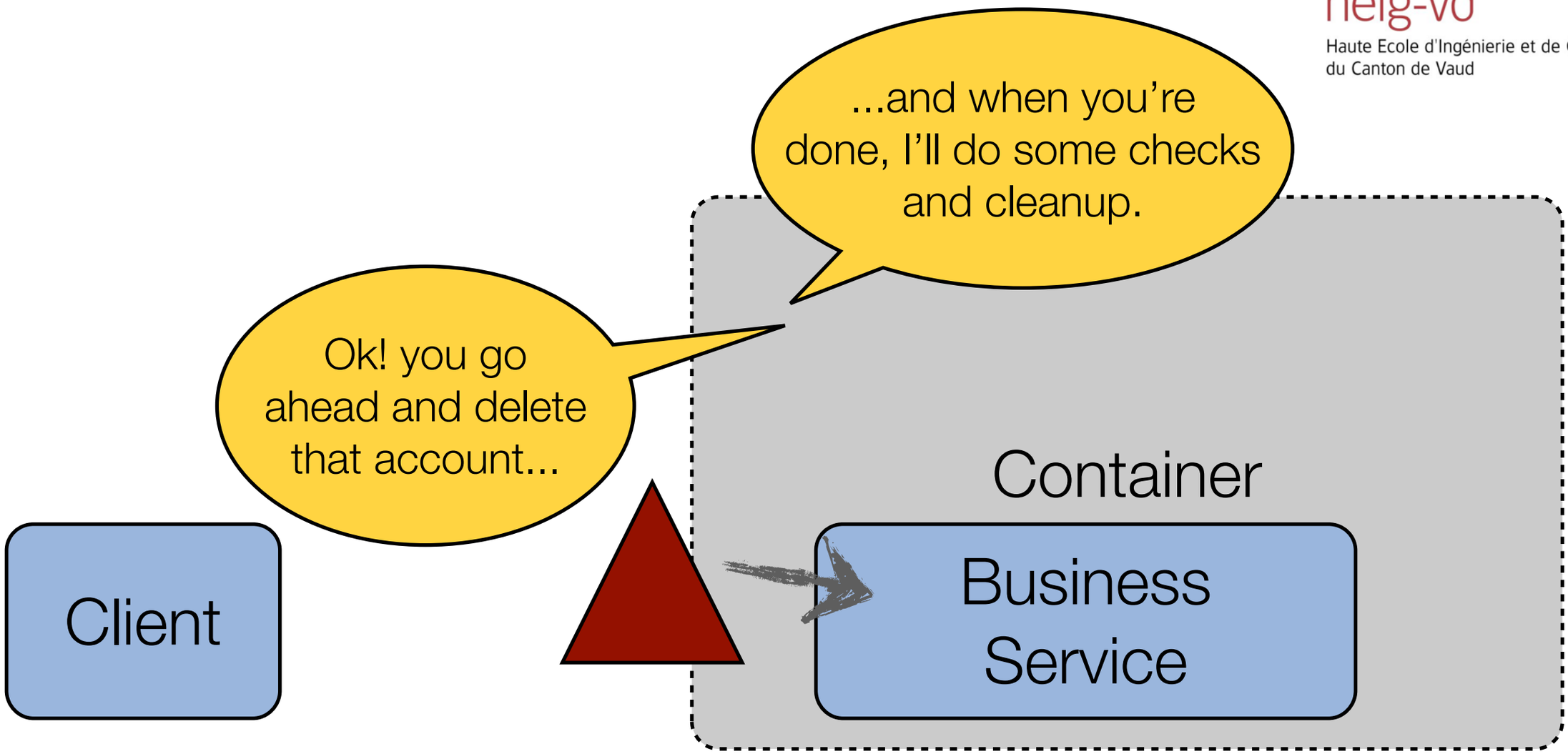

```
deleteAccount();
```

The business service, implemented as a Stateless Session Bean, is a **managed component**.
The client ***thinks*** that he has a direct reference to a Java object.
He is ***wrong***.




```
deleteAccount();
```

In reality, when the client invokes the deleteAccount() methods, the call is going **through the container**.
The container is in a position to **perform various tasks** (security checks, transaction demarcation, etc.)




```
deleteAccount();
```

When done, the container can forward the method call to the business service (your implementation).
On the way back, the response also goes back **via the container**.

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

How is that possible?
How does it work?

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

App Server
Naming Service

lookup();

I need a service for managing customer accounts... Give me a reference!

Client

Your EJB
Interface

deleteAccount();

proxy generated by the app serv.

deleteAccount();

Your EJB
Implementation

deleteAccount();

Your service implementation implements your interface.
The container dynamically generates a class, which implements the same interface. This class performs the technical tasks and invokes your class (proxy).

Scientific Method

Formulate a **question**

"How is the container able to **intercept** method calls on EJBs?"

Propose an **hypothesis**

"I believe that it generates **dynamic proxies**"

Predict the **consequences**

"**If** I am right, then I should see these proxies in the call stack."

Experiment and **measure**

"Let's **throw an exception** in my stateless session bean and have a look at the stack trace"

Analyze

"In the stack trace, I can confirm that the servlet is not directly calling my Stateless Session Bean implementation class."

```
Caused by: java.lang.RuntimeException: just kidding
     at ch.heigvd.amt.lab1.services.CollectorService.submitMeasure(CollectorService.java:15)
     at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
     at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
     at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
     at java.lang.reflect.Method.invoke(Method.java:483)
     at org.glassfish.ejb.security.application.EJBSecurityManager.runMethod(EJBSecurityManager.java:1081)
     at org.glassfish.ejb.security.application.EJBSecurityManager.invoke(EJBSecurityManager.java:1153)
     at com.sun.ejb.containers.BaseContainer.invokeBeanMethod(BaseContainer.java:4786)
     at com.sun.ejb.EjbInvocation.invokeBeanMethod(EjbInvocation.java:656)
     at com.sun.ejb.containers.interceptors.AroundInvokeChainImpl.invokeNext(InterceptorManager.java:822)
     at com.sun.ejb.EjbInvocation.proceed(EjbInvocation.java:608)
     at
org.jboss.weld.ejb.AbstractEJBRequestScopeActivationInterceptor.aroundInvoke(AbstractEJBRequestScopeActivationInte
ceptor.java:46)
     at org.jboss.weld.ejb.SessionBeanInterceptor.aroundInvoke(SessionBeanInterceptor.java:52)
     at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
     at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
     at sun.reflect.DelegatingMeth
     at java.lang.reflect.Method.i
     at com.sun.ejb.containers.int
     at com.sun.ejb.containers.int
     at com.sun.ejb.EjbInvocation.
     at com.sun.ejb.containers.int
     at com.sun.ejb.containers.int
     at sun.reflect.NativeMethodAc
     at sun.reflect.NativeMethodAc
     at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
     at java.lang.reflect.Method.invoke(Method.java:483)
     at com.sun.ejb.containers.interceptors.AroundInvokeInterceptor.intercept(InterceptorManager.java:883)
     at com.sun.ejb.containers.interceptors.AroundInvokeChainImpl.invokeNext(InterceptorManager.java:822)
     at com.sun.ejb.containers.interceptors.InterceptorManager.intercept(InterceptorManager.java:369)
     at com.sun.ejb.containers.BaseContainer.__intercept(BaseContainer.java:4758)
     at com.sun.ejb.containers.BaseContainer.intercept(BaseContainer.java:4746)
     at com.sun.ejb.containers.EJBLocalObjectInvocationHandler.invoke(EJBLocalObjectInvocationHandler.java:212)
     ... 34 more
```

```java
@Stateless
public class CollectorService implements CollectorServiceLocal {

  @Override
  public void submitMeasure(Measure measure) {
    throw new RuntimeException("just kidding");
  }
```

Scientific Method

Formulate a **question**

"How is the container able to **intercept** method calls on EJBs?"

Propose an **hypothesis**

"I believe that it generates **dynamic proxies**"

Predict the **consequences**

"**If** I am right, then I should see these proxies in the call stack."

Experiment and **measure**

"Let's **inspect the call stack in the debugger**"

Analyze

"In the debugger console, I can confirm that there are container generated classes between my servlet and my service implementation."

Projects | Files | Services | Debugging

'http-listener-1(5)' at line breakpoint Collecto

- **CollectorService.submitMeasure:15**
- Hidden Source Calls
- Method.invoke:483
- Hidden Source Calls
- Method.invoke:483
- Hidden Source Calls
- Method.invoke:483
- Hidden Source Calls
  - AroundInvokeInterceptor.intercept:883
  - AroundInvokeChainImpl.invokeNext:82
  - InterceptorManager.intercept:369
  - BaseContainer.__intercept:4758
  - BaseContainer.intercept:4746
  - EJBLocalObjectInvocationHandler.invok
  - EJBLocalObjectInvocationHandlerDeleg
  - $Proxy347.submitMeasure
  - FrontController.processRequest:86
  - FrontController.doGet:103
- Hidden Source Calls
- Thread.run:745

At some point, the method call is forwarded to my implementation.

The reference actually points to a proxy generated by the container. The container performs tasks that are visible in a **long call stack**!

My servlet invokes the method on its **reference** to the EJB.

*An HTTP request has arrived; GF invokes the doGet callback on my servlet (**IoC**). GF has also **injected** a **reference** to the EJB into the servlet.*

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

Introduction to JMeter

JMeter

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

- Open source project, apache foundation
- http://jmeter.apache.org/index.html

"The Apache JMeter™ desktop application is open source software, a 100% pure Java application designed to ***load test functional behavior*** *and* ***measure performance****.*
It was originally designed for ***testing Web Applications*** *but has since* ***expanded to other*** *test functions."*

JMeter

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

*"Apache JMeter may be used to **test performance** both on static and dynamic resources (files, Servlets, Perl scripts, Java Objects, Data Bases and Queries, FTP Servers and more).*

*It can be used to **simulate a heavy load** on a server, network or object to test its strength or to analyze overall performance under **different load types**. You can use it to make a **graphical analysis of performance** or to test your server/script/object behavior under heavy **concurrent** load."*

Types of tests (1)

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

- **Functional tests**
  - Is the system doing what it is supposed to do?
  - Does its behavior comply with functional requirements (use cases)?
  - Selenium is a tool for automating functional testing of web applications (http://seleniumhq.org/)
- **Performance, load and stress tests**
  - What is the response time? What is the consumption of resources? Are there issues (e.g. concurrency issues) that happen under load?
  - Relevant both for interactive and batch use cases.

What to install?

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

- **Main project**
  - http://jmeter.apache.org/download_jmeter.cgi
- **Add-ons**
  - http://jmeter-plugins.org/

Standard Set

Basic plugins for everyday needs. Does not require additional libs to run.

Download | Installation | Package Contents

Extras Set

Additional plugins for extended and complex testing. Does not require additional libs to run.

Download | Installation | Package Contents

Extras with Libs Set

Additional plugins that *do require* additional libs to run.

Download | Installation | Package Contents

WebDriver Set

Selenium/WebDriver testing ability.

Download | Installation | Package Contents

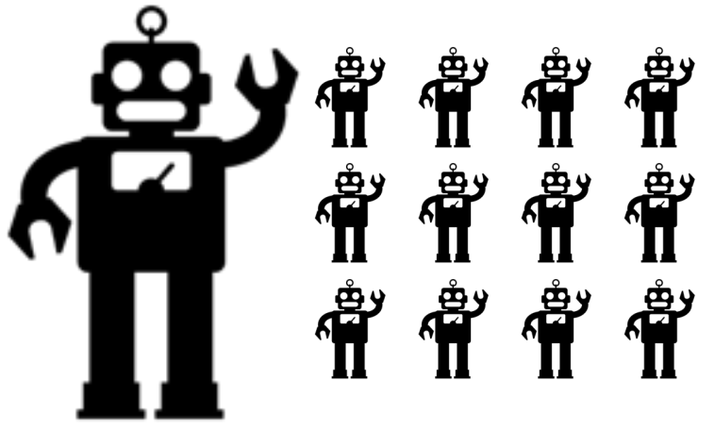
Hadoop Set

Hadoop/HBase testing plugins.

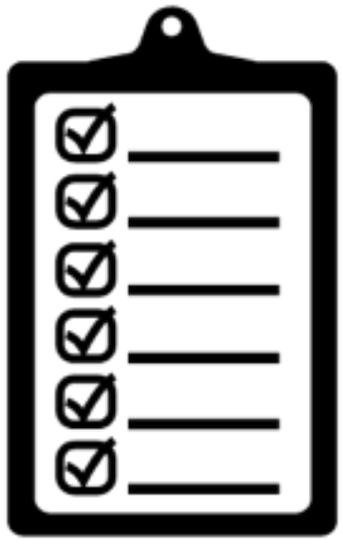
Download | Installation | Package Contents

JMeter Building Blocks

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



ThreadGroup



Test Plan

Listeners
(results & stats)

Samplers
(actions)

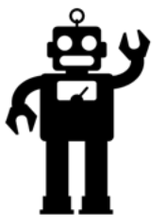
Logic Controllers
& Assertions

Concepts

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

- Test Plan
- ThreadGroup
- Samplers
- Logic Controllers
- Listeners
- Timers
- Assertions
- Configuration Elements
- Pre-Processor Elements
- Post-Processor Elements

Thread Group



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

*"Thread group elements are the **beginning points of any test plan**. All controllers and samplers must be under a thread group. [...]. As the name implies, the thread group element controls the number of threads JMeter will use to execute your test. The controls for a thread group allow you to:*

- *Set the **number of threads***
- *Set the **ramp-up** period*
- *Set the **number of times** to execute the test*

*Each thread will execute the test plan in its entirety and completely independently of other test threads. **Multiple threads are used to simulate concurrent connections to your server application.**"*

Sampler 

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

---

*"Samplers tell JMeter to **send requests to a server and wait for a response**. They are processed in the order they appear in the tree. Controllers can be used to modify the number of repetitions of a sampler.*

- *FTP Request*
- ***HTTP Request***
- *JDBC Request*
- *Java object request*
- *LDAP Request*
- *SOAP/XML-RPC Request*
- *WebService (SOAP) Request*

*Each sampler has several **properties** you can set. You can further customize a sampler by adding one or more Configuration Elements to the Test Plan."*

Logic Controllers

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

"Logic Controllers let you customize ***the logic that JMeter uses to decide when to send requests****. Logic Controllers can change the order of requests coming from their child elements. They can modify the requests themselves, cause JMeter to repeat requests, etc."*

- *Loop Controller*
- *Once Only Controller*
- *Interleave Controller*
- *Random Controller*
- *Random Order Controller*
- *Throughput Controller*
- *Runtime Controller*
- *If Controller*
- *etc.*

Listeners

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

*"Listeners provide **access to the information JMeter gathers about the test cases** while JMeter runs. The **Graph Results** listener plots the response times on a graph. The "**View Results Tree**" Listener shows details of sampler requests and responses, and can display basic HTML and XML representations of the response. Other listeners provide **summary or aggregation information**.*

*Additionally, listeners can **direct the data to a file** for later use.*

Listeners can be added anywhere in the test, including directly under the test plan. They will collect data only from elements at or below their level."

Timers

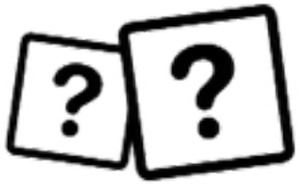
heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

"By default, a JMeter thread sends requests without pausing between each request. We recommend that you specify a delay by adding one of the available timers to your Thread Group. If you do not add a delay, JMeter could ***overwhelm your server*** *by making too many requests in a very short amount of time.*
The timer will cause JMeter to delay a certain amount of time before each sampler which is in its scope .

If you choose to add more than one timer to a Thread Group, JMeter takes the sum of the timers and pauses for that amount of time before executing the samplers to which the timers apply. Timers can be added as children of samplers or controllers in order to restrict the samplers to which they are applied.

To provide a pause at a single place in a test plan, one can use the ***Test Action Sampler.****"*

Assertions



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

---

*"Assertions allow you to **assert facts about responses received** from the server being tested.*

*Using an assertion, you can essentially **"test" that your application is returning the results you expect it to**.*

*For instance, you can assert that the response to a query will **contain some particular text**. The text you specify can be a Perl-style regular expression, and you can indicate that the response is to contain the text, or that it should match the whole response.*

*You can add an assertion to any Sampler. For example, you can add an assertion to a HTTP Request that checks for the text, "</HTML>". JMeter will then check that the text is present in the HTTP response. If JMeter cannot find the text, then it will **mark this as a failed request.**"*

Configuration Elements

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

"A configuration element works closely with a Sampler. Although it does not send requests (except for HTTP Proxy Server), it can add to or modify requests.

A configuration element is accessible from only inside the tree branch where you place the element. For example, if you place an ***HTTP Cookie Manager*** *inside a Simple Logic Controller, the Cookie Manager will only be accessible to HTTP Request Controllers you place inside the Simple Logic Controller"*

- *HTTP Authorization Manager*
- *HTTP Cache Manager*
- *HTTP Cookie Manager*
- *HTTP Request Defaults*
- *HTTP Header Manager*

How to Create Test Scenarios?

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

- **Option 1 : manually**
  - Create a Test Plan
  - Add a Thread Group
  - Add HTTP samplers and specify HTTP request parameters
- **Option 2 : recording with JMeter configured as an HTTP proxy**
  - http://jmeter.apache.org/usermanual/jmeter_proxy_step_by_step.pdf
  - Do manual adjustments

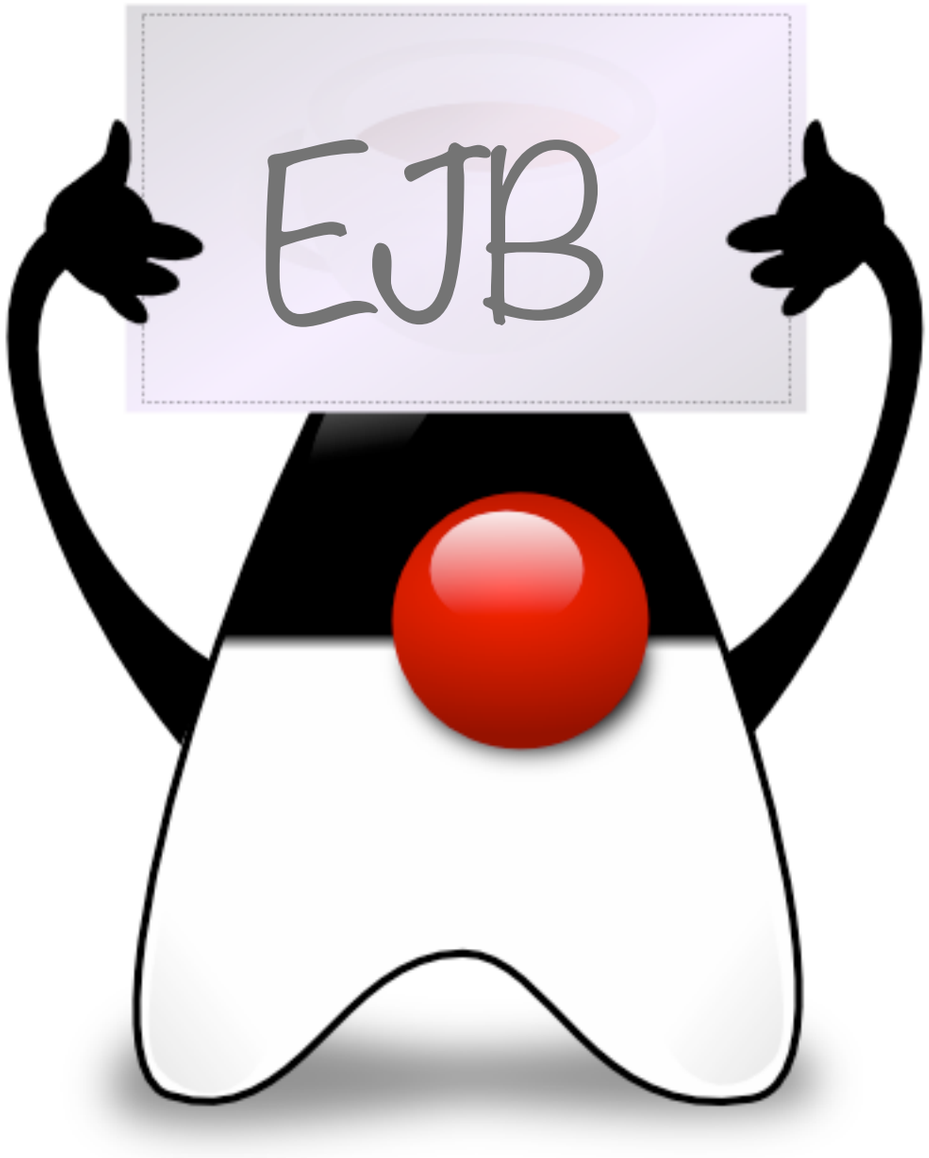
Advanced Usage

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

- **Use variables:**
  - Very often, it is needed to use parts of an HTTP response into follow-up requests (e.g. session ids).
  - `http://jmeter.apache.org/usermanual/test_plan.html#properties`
- **Use more than one machine:**
  - With JMeter running on a single machine, it is common to “exhaust” the client before the server (especially when testing a “real” infrastructure with multiple nodes).
  - For real performance tests, it is therefore recommended to use multiple machines for injecting load into the network. One way to do it is use use virtual machines in a cloud environment (e.g. on Amazon EC2).
  - Multiple JMeter clients can be coordinated by a master and results can be collected and aggregated (`http://jmeter.apache.org/usermanual/jmeter_distributed_testing_step_by_step.pdf`)

heig-vd

Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud



Demo - Servlets are not thread-safe

Description

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

- 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**.

Code: ConcurrencyServlet

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

```
@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);
  }
```

add 10 threads

concurrency.jmx (/Users/admin/Documents/heig-vd/Teaching/AMT/repos/Teaching-HEIGVD-AMT-Example-MVC/MVCDemo/jmeter-scripts/concurr...

- Test Plan
  - jp@gc - Ultimate Thread Grou
    - HTTP Request Defaults
    - HTTP Cookie Manager
    - Access login page
    - Send credentials
    - Access concurrency page
    - Regular Expression Extrac
    - Regular Expression Extrac
    - View Results Tree
    - Summary Report
    - Debug Sampler
  - WorkBench

jp@gc - Ultimate Thread Group

Name: jp@gc - Ultimate Thread Group

Comments:

Help on this plugin v1.3.0

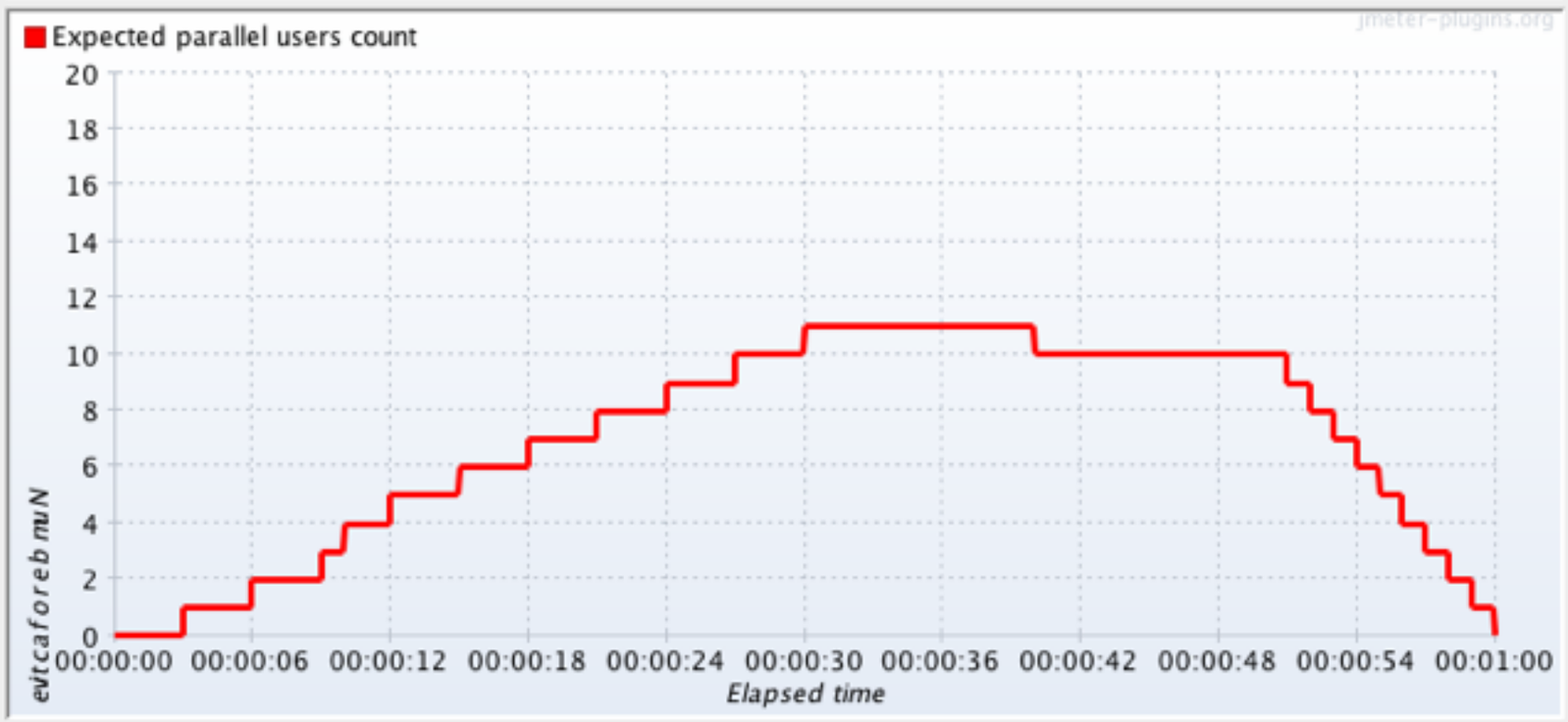
Action to be taken after a Sampler error

(•) Continue ( ) Start Next Thread Loop ( ) Stop Thread ( ) Stop Test ( ) Stop Test Now

Threads Schedule

Start Threads Count	Initial Delay, sec	Startup Time, sec	Hold Load For, sec	Shutdown Time
1	0	10	20	10
10	0	30	20	10

Add Row | Copy Row | Delete Row



concurrency.jmx

concurrency.jmx (/Users/admin/Documents/heig-vd/Teaching/AMT/repos/Teaching-HEIGVD-AMT-Example-MVC/MVCDemo/jmeter-scripts/concurr...

- Test Plan
  - jp@gc - Ultimate Thread Grou
    - HTTP Request Defaults
    - HTTP Cookie Manager
    - Access login page
    - Send credentials
    - Access concurrency page
    - Regular Expression Extrac
    - Regular Expression Extrac
    - View Results Tree
    - Summary Report
    - Debug Sampler
  - WorkBench

POST

HTTP Request

Name: Send credentials

Comments:

Web Server

Server Name or IP: Port Number:

Timeouts (milliseconds)

Connect:

HTTP Request

Implementation: Protocol [http]: Method: POST Content

Path: /MVCDemo/auth

Redirect Automatically | Follow Redirects | Use KeepAlive | Use multipart/form-data for POST | Browser

Parameters | Body Data

Send Parameters With the Request:

Name:	Value
action	login
email	a@a.com
password	toto

Detail | Add | Add from Clipboard | Delete | Up | Dov

Send Files With the Request:

File Path:

Add | Browse... | Delete

Proxy Server

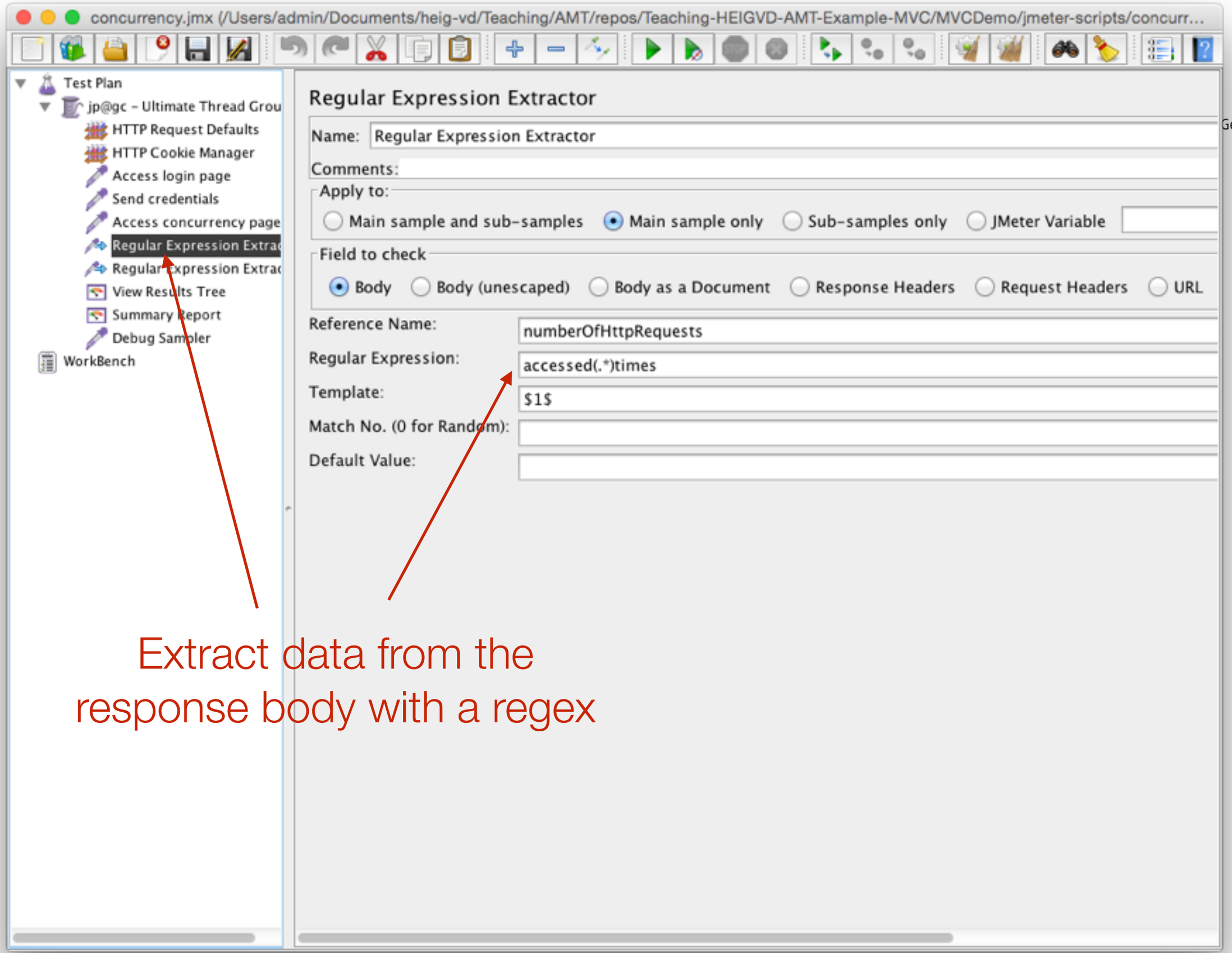
Server Name or IP: Port Number: Username

Embedded Resources from HTML Files

Retrieve All Embedded Resources | Use concurrent pool. Size: 4 | URLs must match:

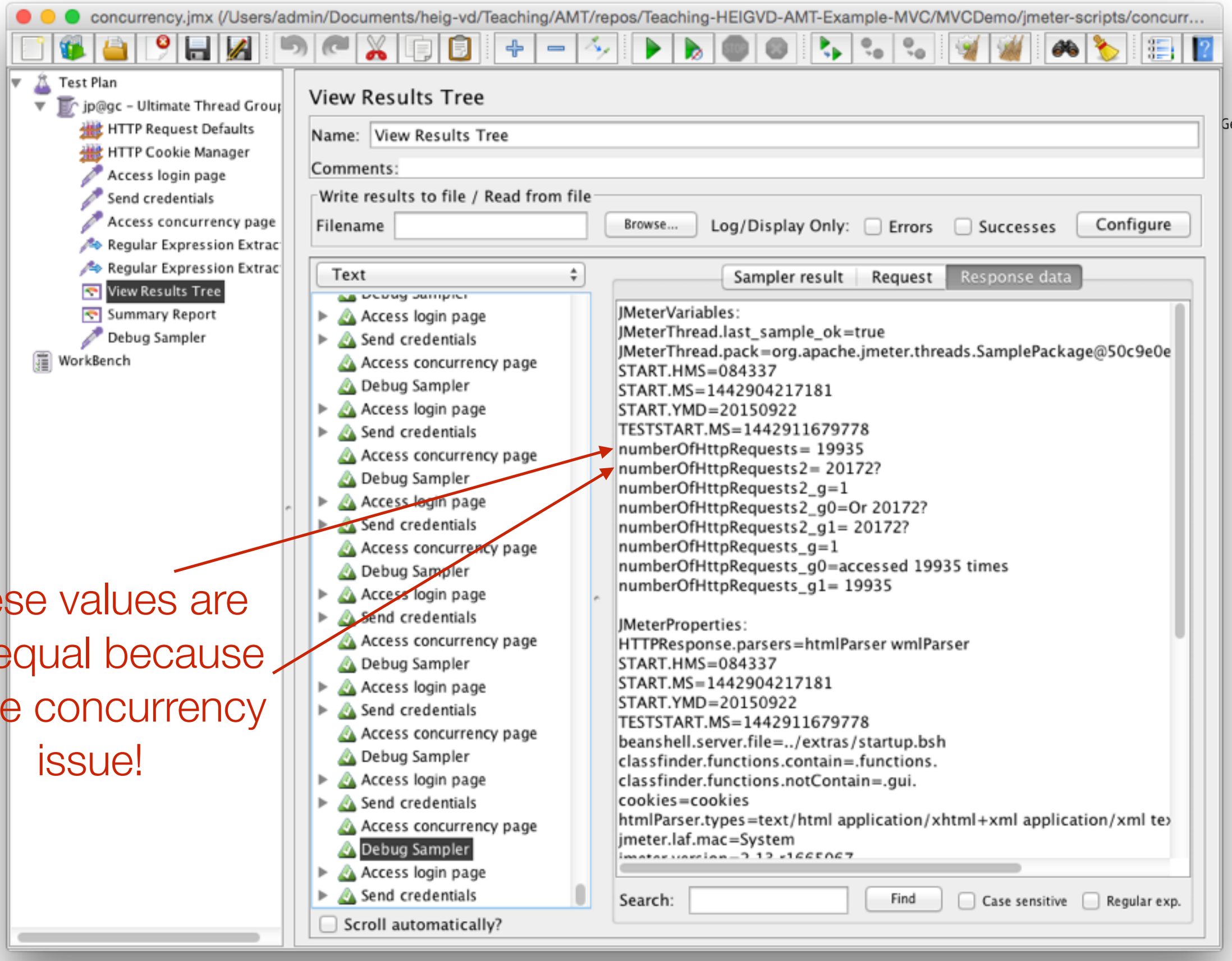
Content of the form

Gestion



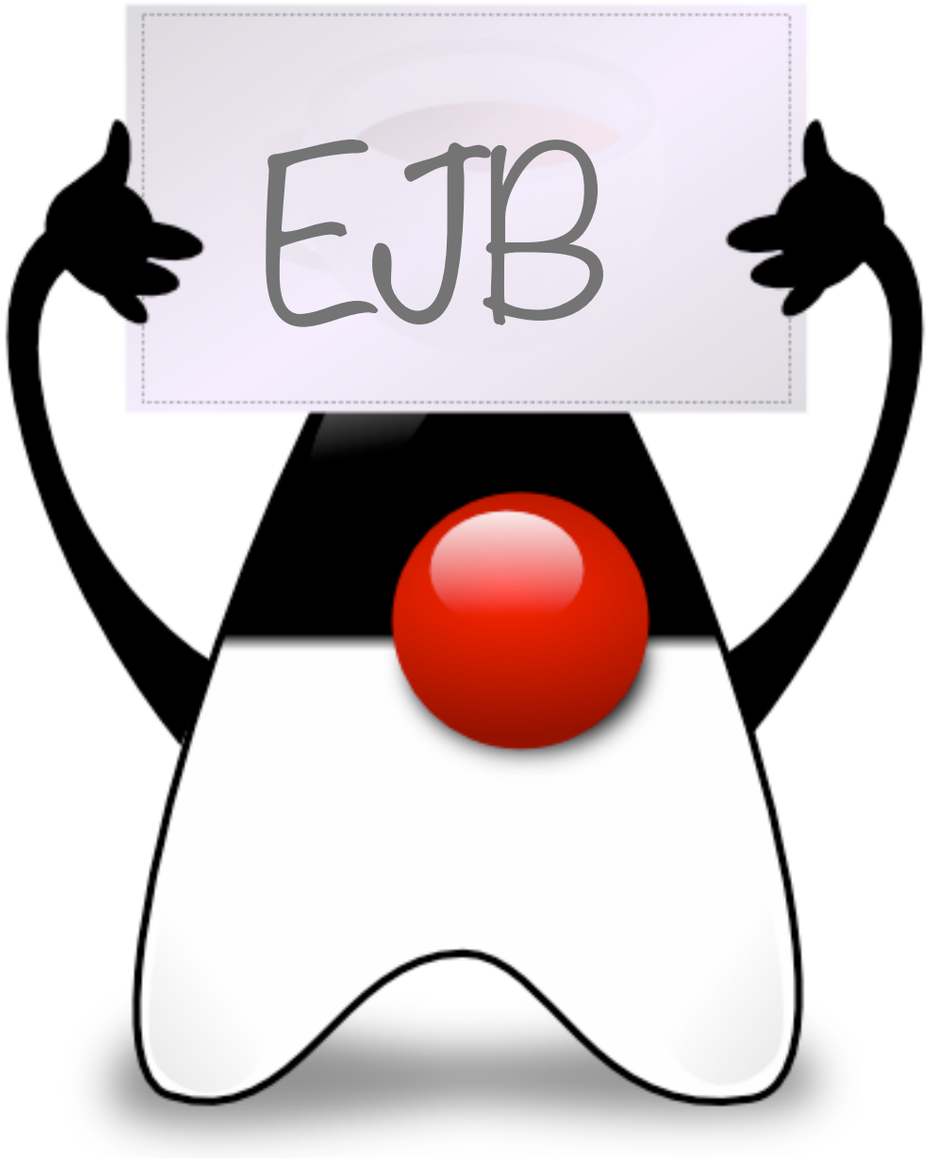
Gestion

Extract data from the
response body with a regex



These values are not equal because of the concurrency issue!

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



Business Services & EJB

The app server **manages a pool** of EJB instances. What does it mean?

Imagine you are a bar owner…
How do you organize your staff?

"Hire only one staff member. If a customer arrives, wait until staff is done with the previous one."

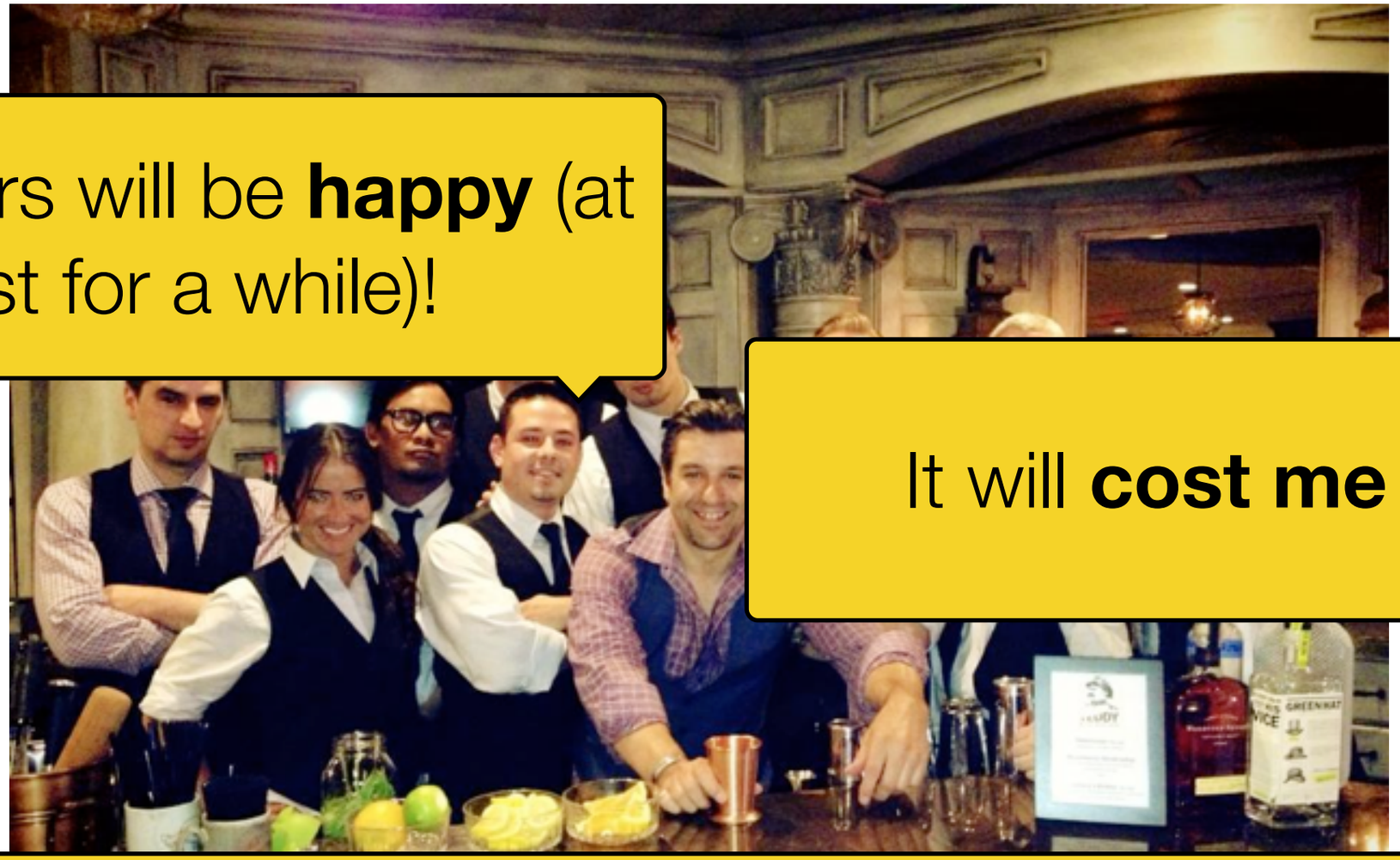
Customers will **wait too long** on Saturday evening!

Some people will not even get into the **queue**!

"Hire only one staff member. If a customer arrives, ask him to serve him at the same time as the previous one."

If a bar tender doing too many things at the same time, he will make **mistakes**!

"As soon as a customer arrives, if no staff member is available, hire a new one."



Customers will be **happy** (at least for a while)!

It will **cost me a lot**.

On Saturday evening, I will have 100 bar tenders. They will **compete** for the shaker and space. They will be **inefficient** and customers will have to **wait for a very long time** before getting their drink!

"Hire a pool of staff members. Ask each staff member to serve his customers one after the other."

I can **adjust the size of the pool** to **keep a balance** between the length of the queue and the contention on shared resources.
I will be able to **deal with affluence** on Saturday evening!"

Handling requests **without concurrency** does not scale, because of **queuing**.

A single **servlet** instance handles requests concurrently, but is **not thread-safe**.

Using a **new POJO instance** for every request does not scale, as it **exhausts resources**.

Using a **pool of EJBs** is a way to throttle requests in a **thread-safe** manner.

How does the EJB Container implement **resource pooling**?

- Remember that EJBs are **managed components** and are instantiated by the container.
- When the container creates an EJB instance, it **puts it in a pool**. The instance is "**ready to serve**", waiting for some job.
- When a client component (e.g. a servlet) invokes a method on a Stateless Session Bean, the container **checks if there is an available instance in the pool**:
  - If **yes**, then the instance is removed from the pool (it will not be available for other requests until done) and method is forwarded to it. When the method returns, the instance goes back into the pool.
  - If **no**, then **either** the request is **put into a queue** (until a instance is ready to serve), **or** the container **resizes the pool** by creating new EJB instances.

Monitoring Service

localhost:4848/common/index.jsf

Home | About... | Help

User: anonymous | **Domain:** domainAMT | **Server:** localhost

GlassFish™ Server Open Source Edition

- JDBC
- JMS Resources
- JNDI
- JavaMail Sessions
- Resource Adapter Configs
- **Configurations**
- default-config
  - Admin Service
  - Availability Service
  - Connector Service
  - EJB Container
  - Group Management Service
  - HTTP Service
  - JVM Settings
  - Java Message Service
  - Logger Settings
  - Monitoring
  - Network Config
  - ORB
  - Security
  - System Properties
  - Thread Pools
  - Transaction Service
  - Virtual Servers
  - Web Container
  - -config

Deploy all MBeans needed for monitoring

Component Level Settings (16)

Level: Change Level

Select	Module	Monitoring Level
☐	Jvm	OFF
☐	Transaction Service	OFF
☐	Connector Service	OFF
☐	Jms Service	OFF
☐	Security	OFF
☐	Web Container	HIGH
☐	Jersey(RESTful Web Services)	OFF
☐	Web Services Container	OFF
☐	Java Persistence	OFF
☐	Jdbc Connection Pool	OFF
☐	Thread Pool	OFF
☐	Ejb Container	HIGH
☐	ORB (Object Request Broker)	OFF
☐	Connector Connection Pool	OFF

If we want to monitor the EJB pool, we first need **activate the monitoring**.

Application Monitoring

localhost:4848/common/index.jsf

Home | About... | Help

User: anonymous | **Domain:** domainAMT | **Server:** localhost

GlassFish™ Server Open Source Edition

- Common Tasks
  - Domain
  - server (Admin Server)
  - Clusters
  - Standalone Instances
  - Nodes
    - localhost-domainAMT
  - Applications
  - Lifecycle Modules
  - Monitoring Data
  - Resources
    - Concurrent Resources
    - Connectors
    - JDBC
    - JMS Resources
    - JNDI
    - JavaMail Sessions
    - Resource Adapter Configs
  - **Configurations**
    - default-config
      - Admin Service
      - Availability Service
      - Connector Service
      - EJB Container
      - Group Management
      - ...

General | **Resources** | **Properties** | **Monitor** | **Batch** | **JMS Physical Destinations**

Applications | **Server** | **Resources**

Application Monitoring

Refresh

Click Configure Monitoring and enable monitoring for a component or service by selecting either LOW or HIGH. See the Online Help for more information.

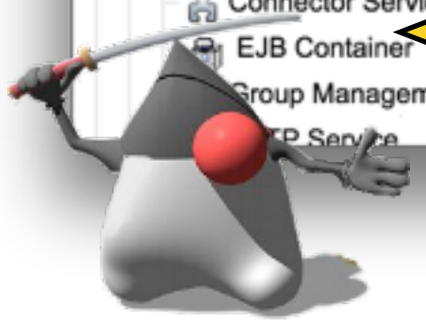
Instance Name: server

Application : SimpleApp **Component :** default

Monitor (23 Statistics)

Servlet Instance Statistics : SimpleApp/server/default

Name	Value	Start Time	Last Sample Time	Details	Description
RequestCount	0 count	Sep 24, 2014 5:26:03 PM	Sep 24, 2014 5:38:09 PM	--	Number of requests processed
ErrorCount	0 count	Sep 24, 2014 5:26:03 PM	--	--	Number of error responses (that is, responses with a status code greater than or equal to 400)
ProcessingTime	0 millisecond	Sep 24, 2014 5:26:03 PM	Sep 24, 2014 5:38:09 PM	--	Average response time



Even then, it seems that **no EJB monitoring data** is available in the web console…?

GlassFish Server Open Sou ×

localhost:4848/monitoring/domain/server/applications/SimpleApp/CollectorService

GlassFish Server Open Source Edition 4.1 REST Interface

- createcount
  - unit : count
  - lastsampletime : -1
  - name : CreateCount
  - count : 0
  - description : Number of times EJB create method is called or 3.x bean is looked up
  - starttime : 1411572510481
- methodreadycount
  - unit : count
  - current : 1
  - lastsampletime : 1411572807901
  - lowwatermark : 0
  - lowerbound : 0
  - upperbound : 32
  - name : MethodReadyCount
  - description : Number of stateless session beans in MethodReady state
  - highwatermark : 1
  - starttime : 1411572510481
- removecount
  - unit : count
  - lastsampletime : -1
  - name : RemoveCount
  - count : 0
  - description : Number of times EJB remove method is called
  - starttime : 1411572510481

Child Resources

bean-methods
bean-pool

/monitoring/domain/server/applications/**$APP**/**$EJB**

Using the **monitoring REST API** gives access to a huge quantity of precious information (in a **tree**)!!

Scientific Method

Formulate a **question**

"How does Glassfish manages EJB pools?"

Propose an **hypothesis**

"**I believe** that it increases the pool size if no bean is "ready to work" when a new request comes in.

Predict the **consequences**

"**If I am right**, then by generating sufficient load, I should be able to trigger a "pool resize operation"

Experiment and **measure**

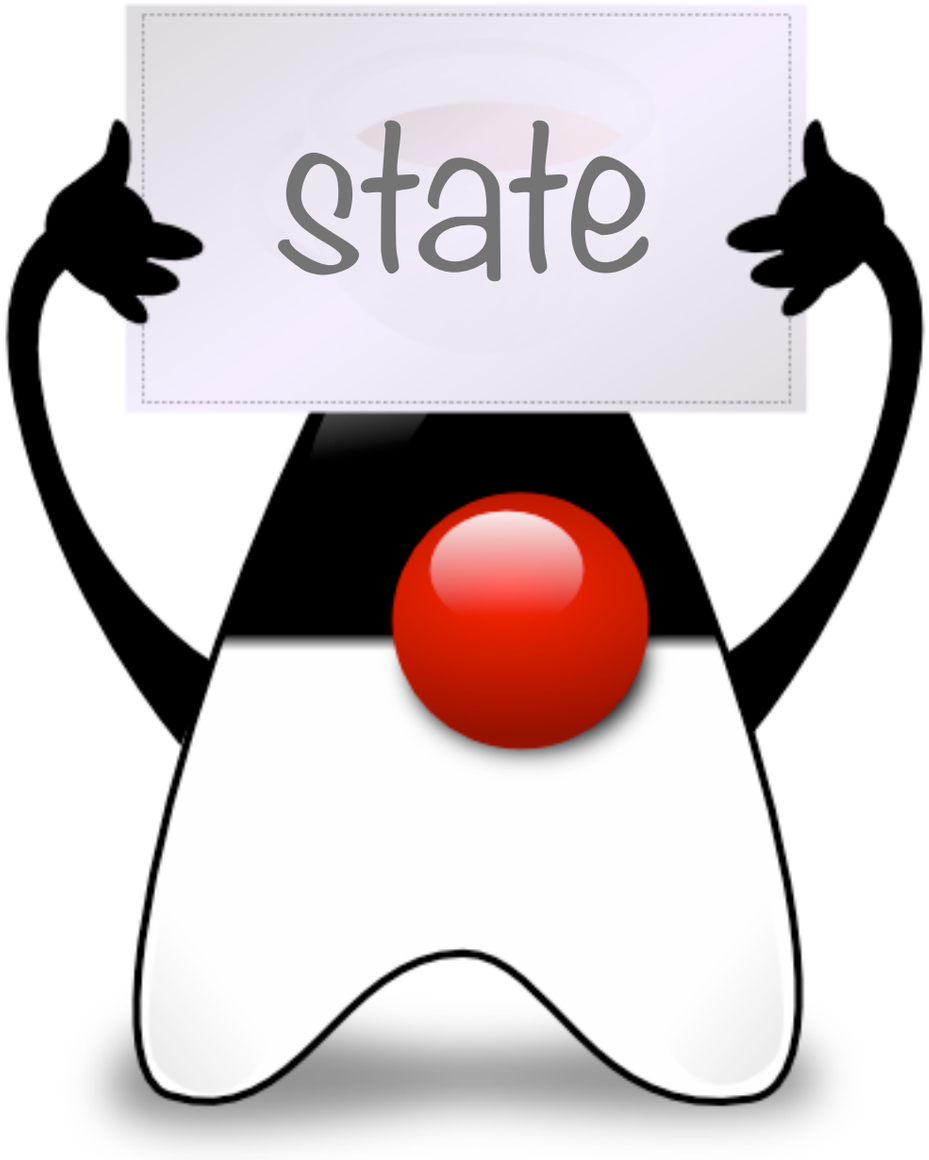
"Let's add a Thread.sleep(4000) statement in a EJB method to make it easier to exhaust the pool. Let's collect data via GF monitoring."

Analyze

"In the REST Monitoring API, I can validate that the pool has been resized."

heig-vd

Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud



Experiment - HTTP Sessions

Lab Objective

What is the impact of storing objects in **HTTP sessions** using the Servlet API, both in terms of **memory consumption** and of **performance**?

Answer this question in a systematic and structured way and provide supporting **evidence**.

Two questions, two experiments!

Question 1
What is the impact of storing objects in **HTTP sessions** using the Servlet API in terms of **memory consumption**?

Propose an **hypothesis**

Predict the **consequences**

Experiment and **measure**

Analyze

Two questions, two experiments!

Question 2
What is the impact of storing objects in **HTTP sessions** using the Servlet API in terms of **performance**?

Propose an **hypothesis**

Predict the **consequences**

Experiment and **measure**

Analyze

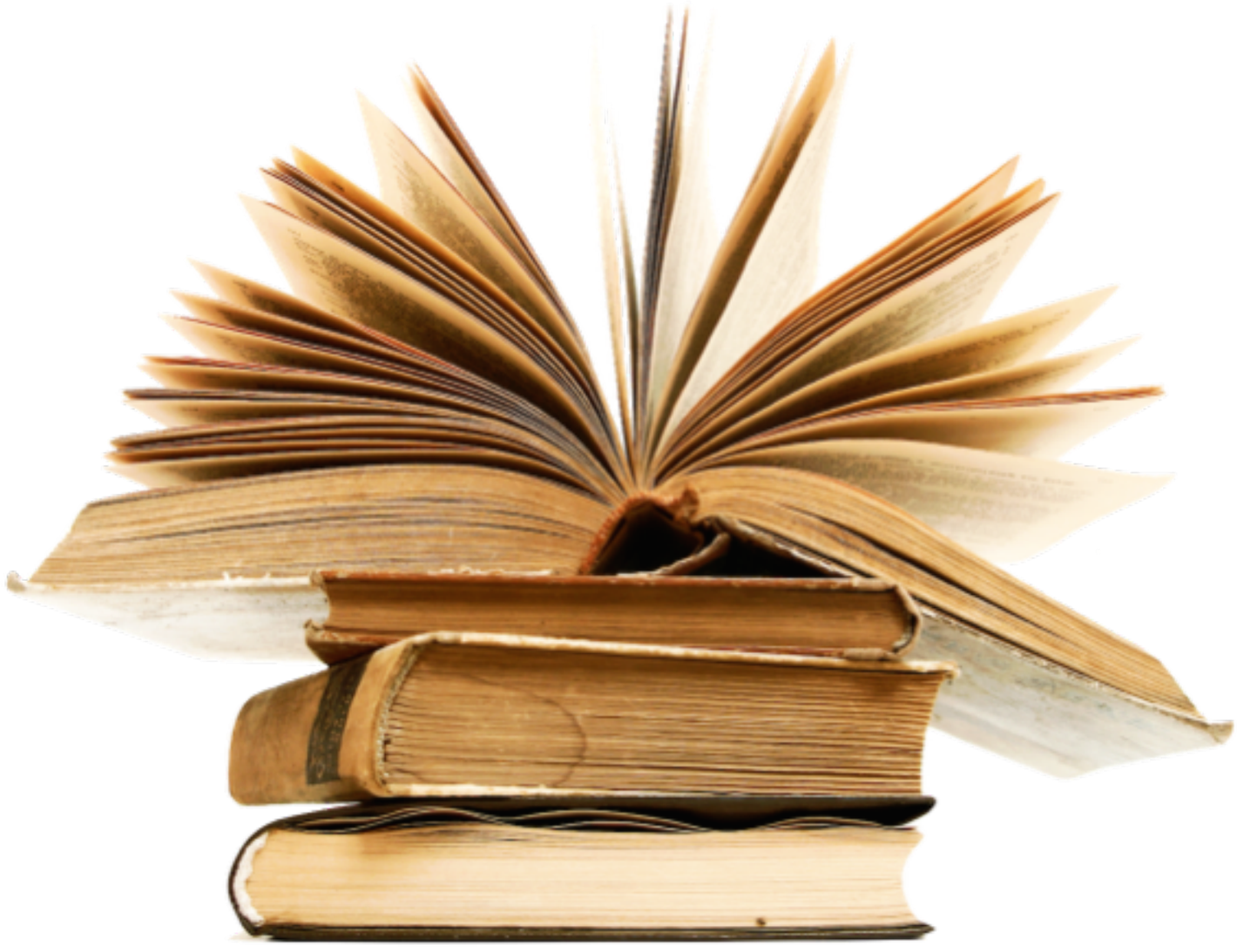
Let's start with **memory consumption**. What is your hypothesis?

If your hypothesis holds, **then** what are the **measurable consequences**?

How would you **design an experiment** to get evidence?

heig-vd

Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud



References

MUST READ for the Tests

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

- **Selected sections from the Java EE 7 tutorial**
  - http://docs.oracle.com/javaee/7/tutorial/doc/ejb-intro.htm#GIJSZ

Why is it very important to **code in english** (i.e. to use english for variable, class and method names, to use english in your comments)?

What is the difference between **servlets** and **stateless session beans** in terms of **concurrency**?

How is **dependency injection** applied in the context of EJBs? Illustrate your answer with code examples.

In the context of servlet to EJB interactions, what is the **alternative to dependency injection**?

What is the role of the **EJB pool** managed by the application server?

How is **inversion of control** applied in the Java EE platform? Give one example and illustrate it with a code example.

How is the EJB container able to perform **security checks** when a method is invoked on a stateless session bean? Illustrate your answer with a UML **sequence diagram**.

What are **four** types of EJBs?

What code do you write in a servlet in order to use the naming service provided by Glassfish, in order to **lookup** an EJB by name?

What code do you write in a servlet in order to **inject a dependency** on an EJB? Why is this code not working in a POJO?