Lecture 1: Getting started…

Olivier Liechti
AMT

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

Introduction

Multi-Tiered Applications

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

Frameworks

spring

EHCACHE

HIBERNATE

APIs

servlet

EJB

JDBC

MySQL

JSP

JPA

mongoDB

JAX-RS

Java EE
Application Server

JAX-WS

JMS

Tools

Apache maven

Jenkins

Apache JMeter™

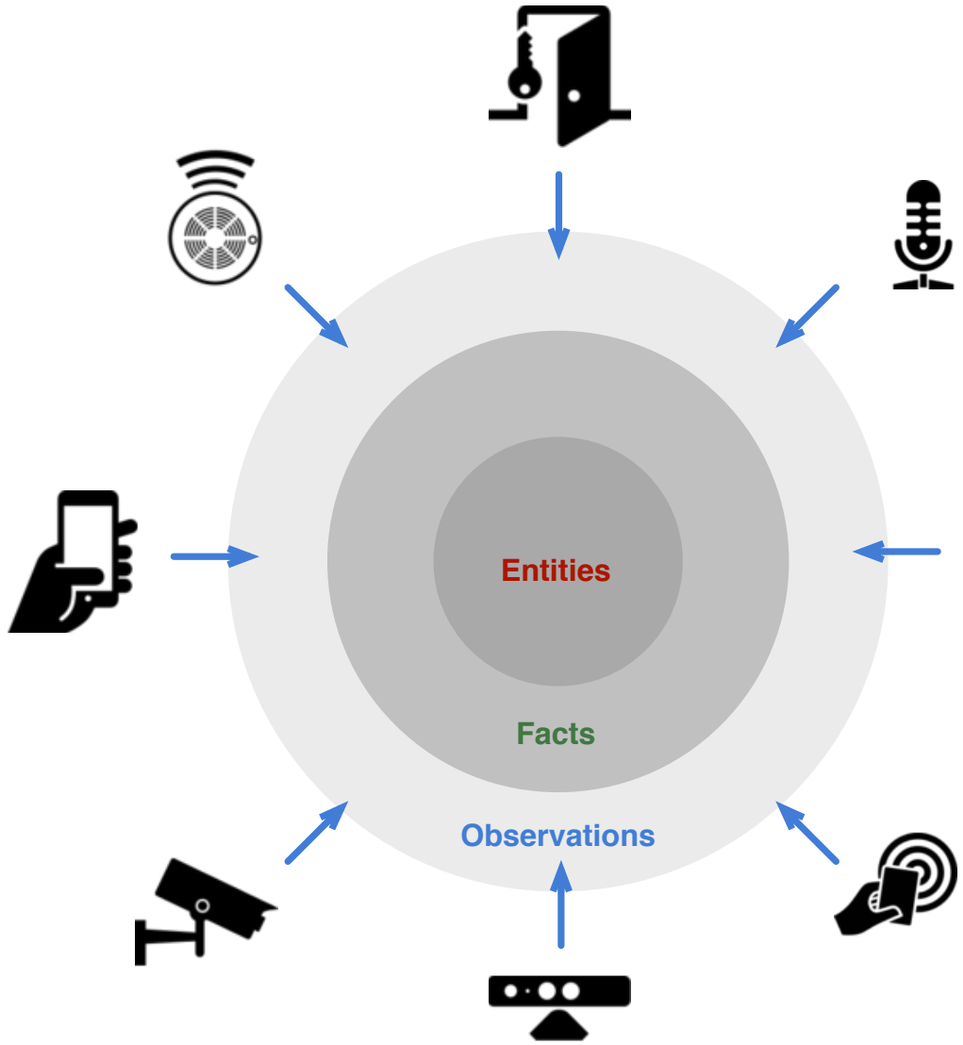
jbehave

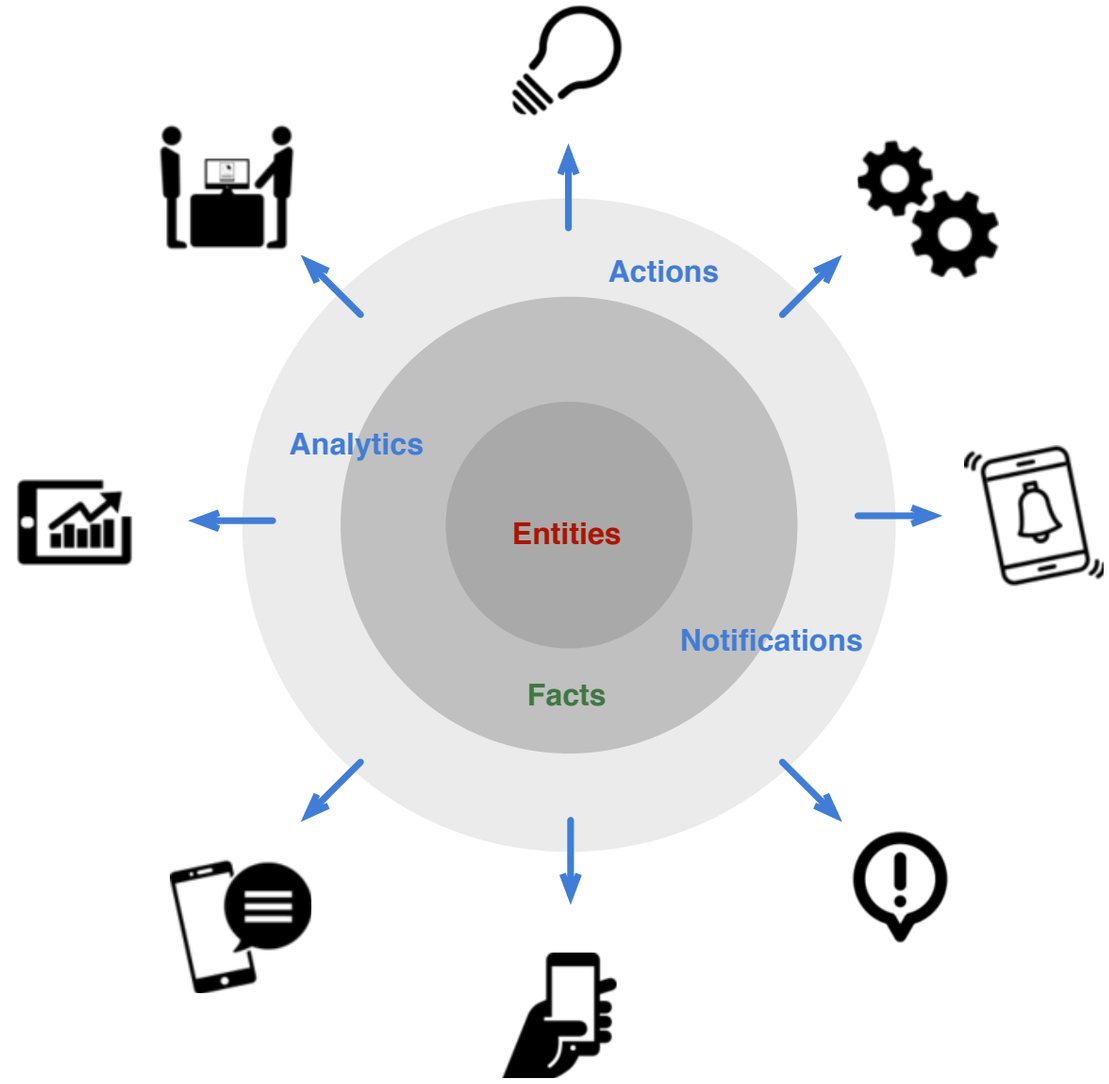
docker

Selenium Webdriver

Web of Things

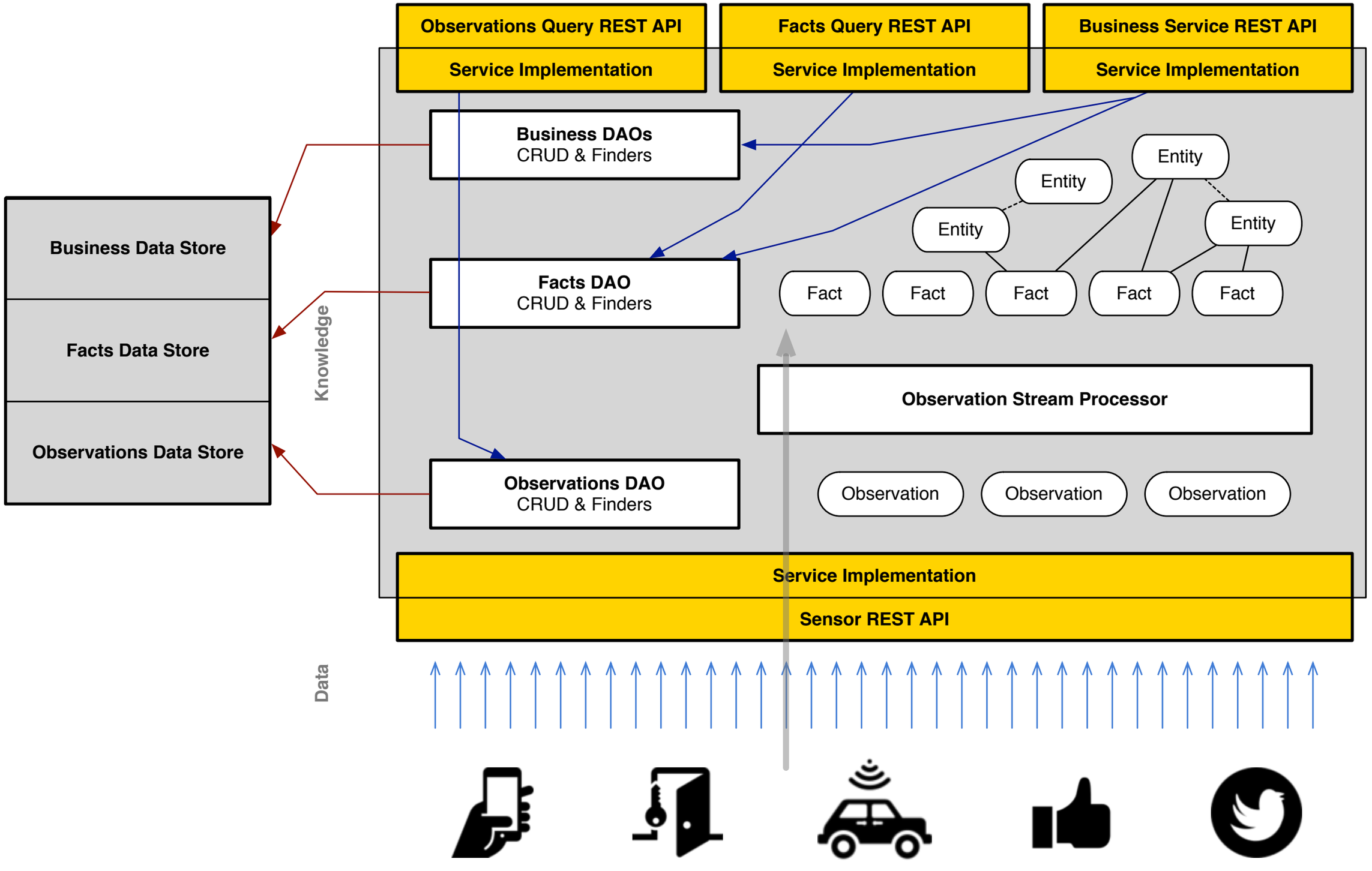
heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud





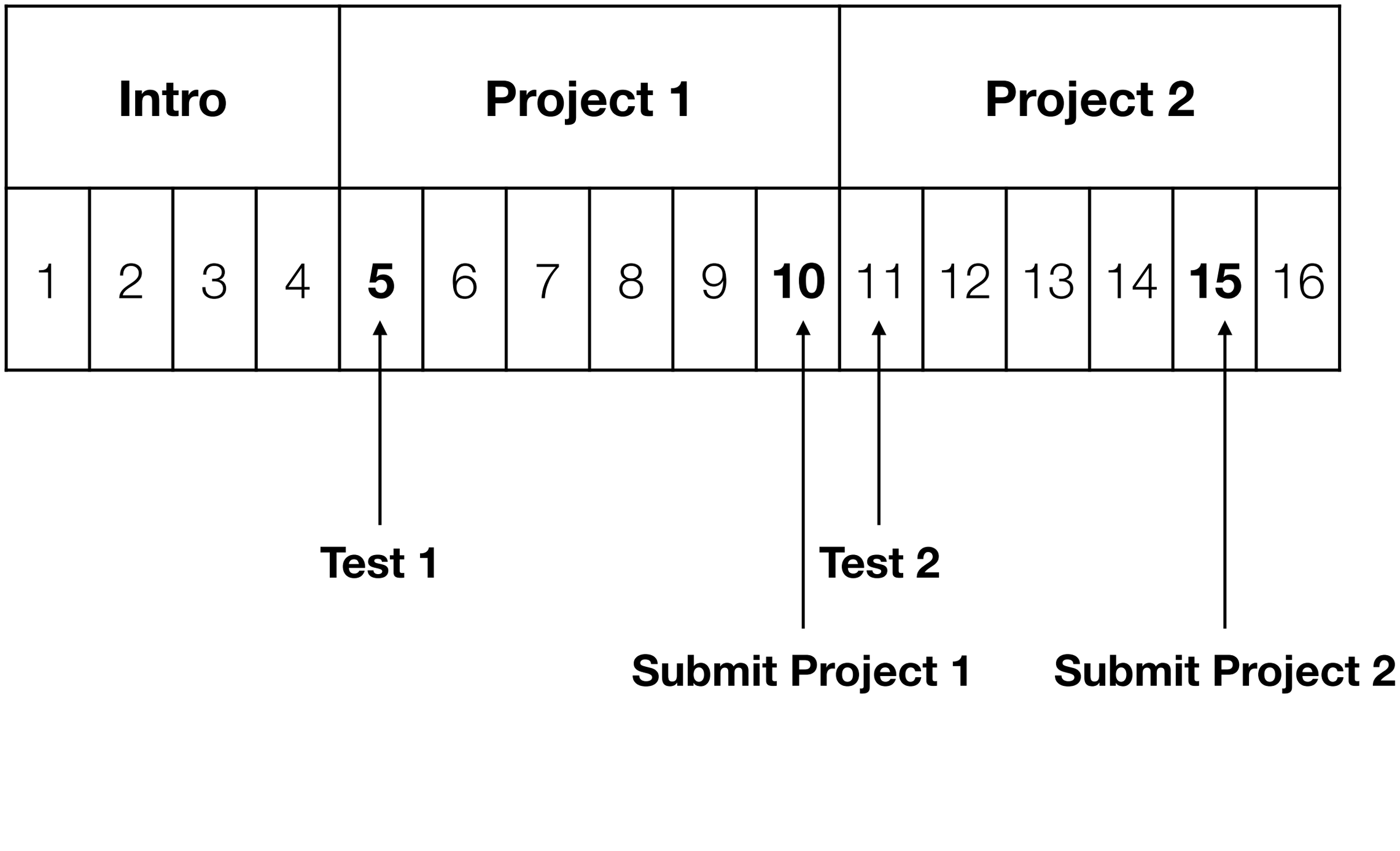
Web of Things

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



Planning

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

When you see me, **watch out**! I will make you aware of pitfalls and dangers!

I successfully passed the AMT tests and the exam. I remember **some** of the questions and important stuff.

When you see me, it's **practice time**! Follow instructions on the slide (during the class)

Quick Poll

Quick Poll: are you familiar with…

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

- Servlets and JSPs
- EJBs
- REST APIs and JAX-RS
- JDBC
- JPA
- JMS
- apache maven
- Jenkins
- Selenium

Agenda

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

13h00 - 14h00	60’	**Lecture** General introduction Architecture, Java EE, Containers, Glassfish, MVC
14h00 - 15h15	75’	**Lab 01** A first “multi-tiered” application with Java EE
15h15 - 15h45		Break
15h45 - 16h15	30’	**Lecture** Session management in the web tier and its impacts
16h15 - 17h45	90’	**Lab 02** Assessing and validating the impact of HTTP sessions
17h45 - 18h00	15’	**Status & discussion**

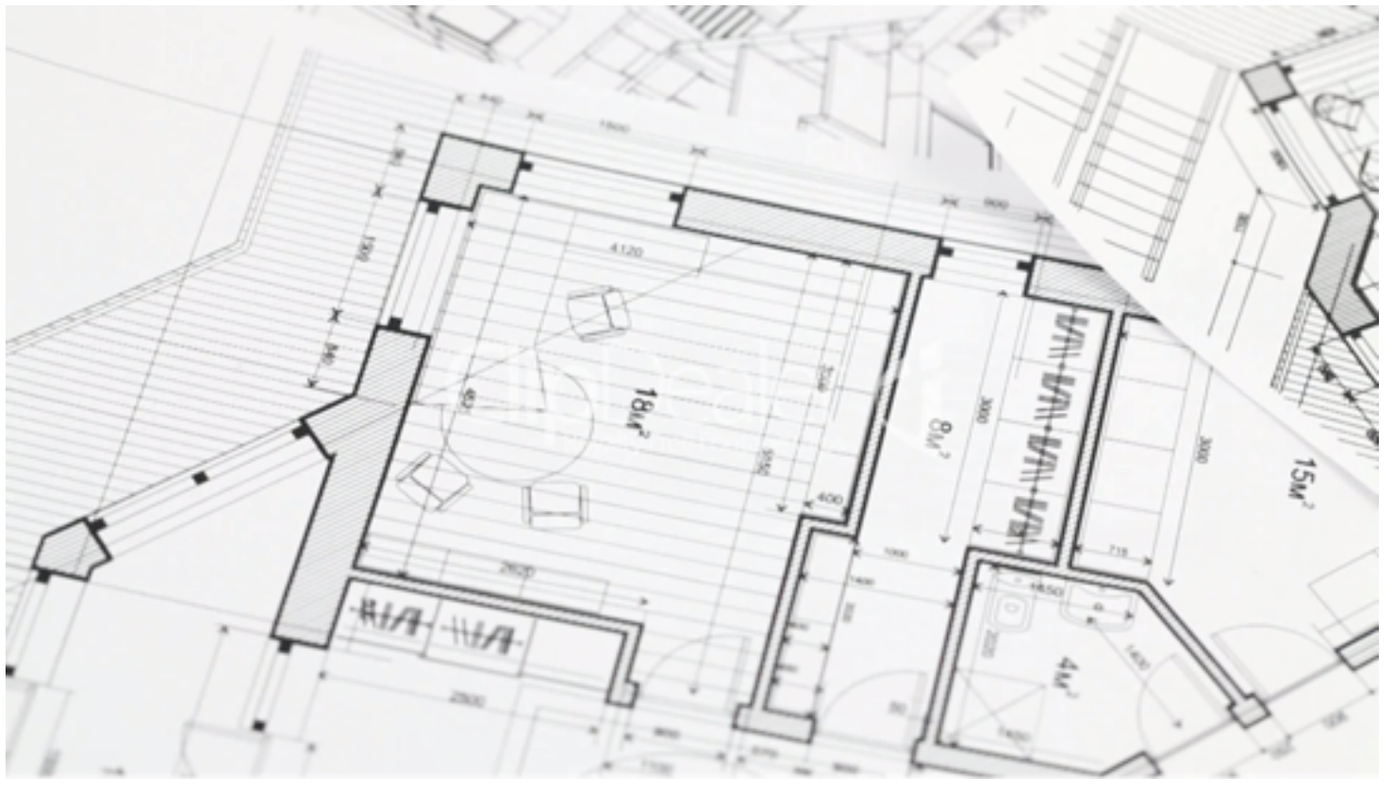
heig-vd

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

Lecture - Intro

heig-vd

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



Architecture

What do we mean by “**architecture**” “**architectural style**” in the context of software systems?

- There is **no single definition** for software architecture!

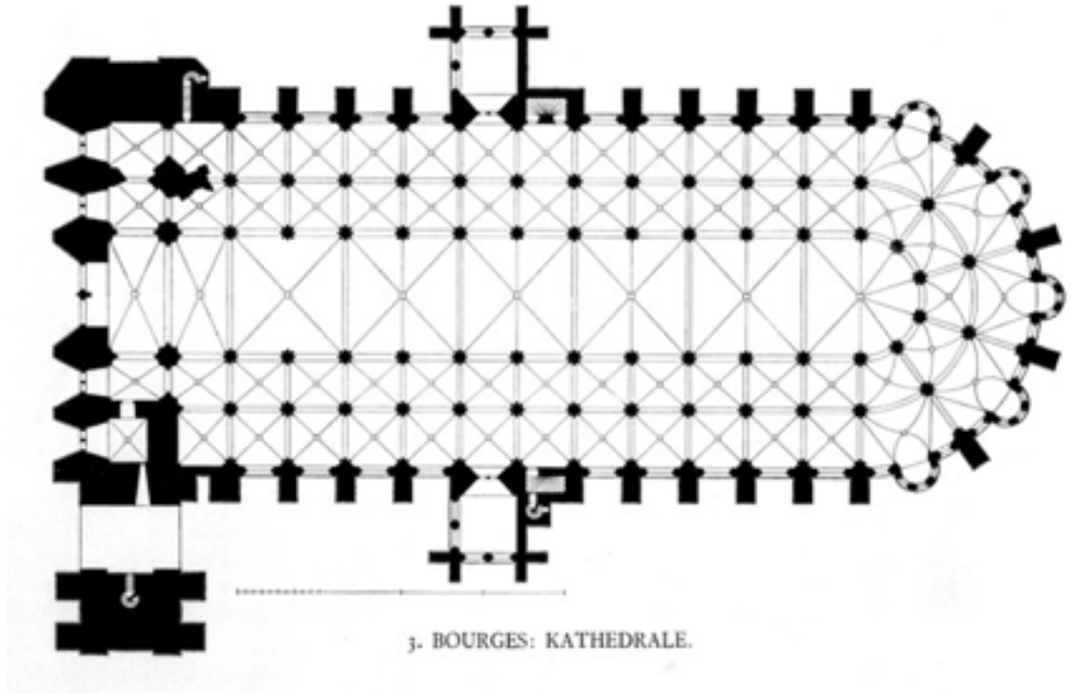
- The **architecture** of **a particular application** is a **description** of its **high-level structure** and **behavior**. It is a description of its **components**, how they **interact** with each other and with **external systems**.

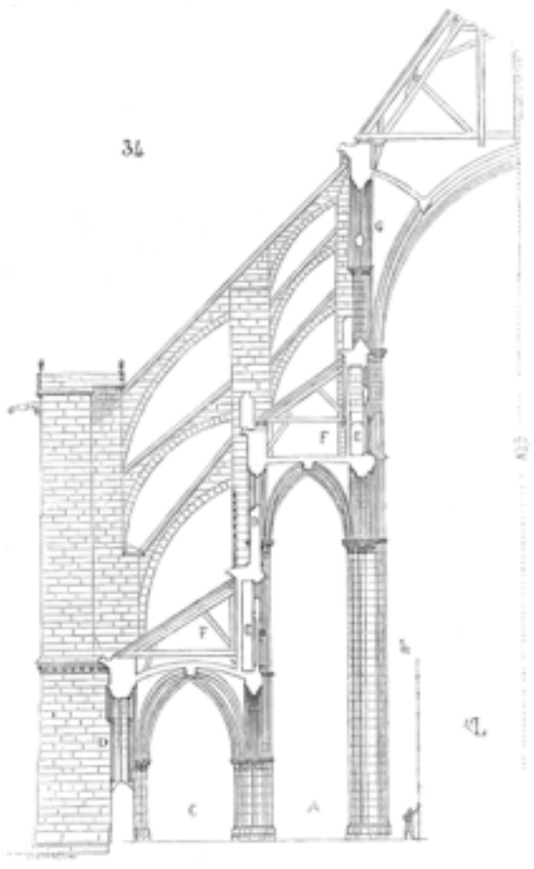
- The **architecture** of **one particular** application is **documented** with a **collection of models** (**UML** diagrams are examples of such models).

- An **architectural style** is a **set of core principles and rules** (**architectural patterns**), which are **common** to a large number of different applications.

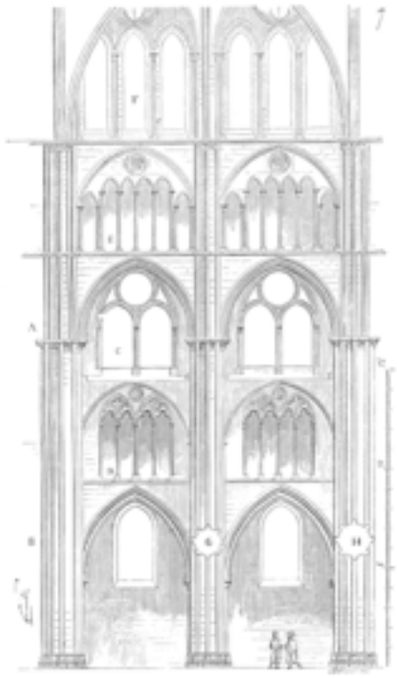
heig-vd

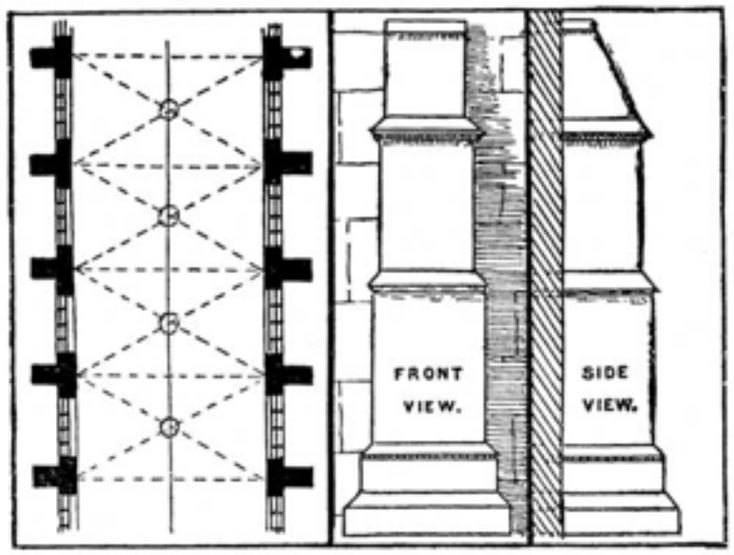
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

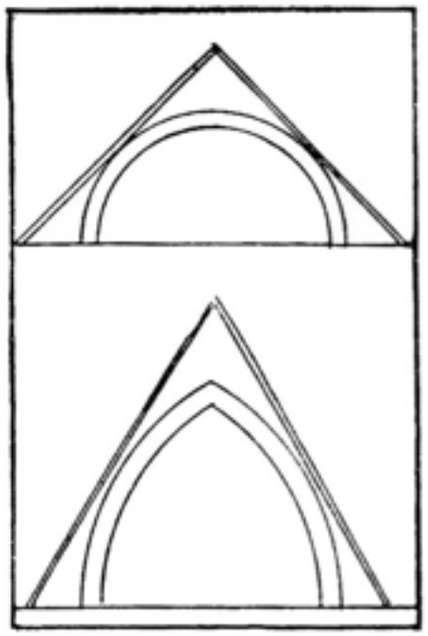


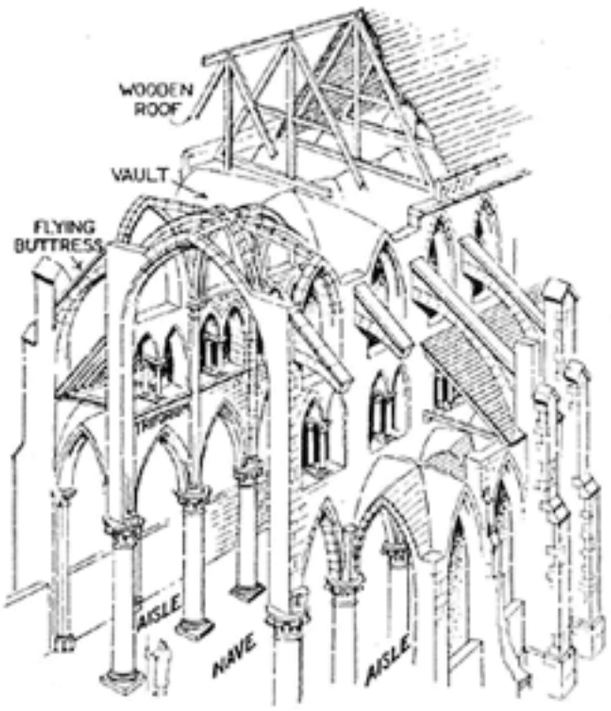


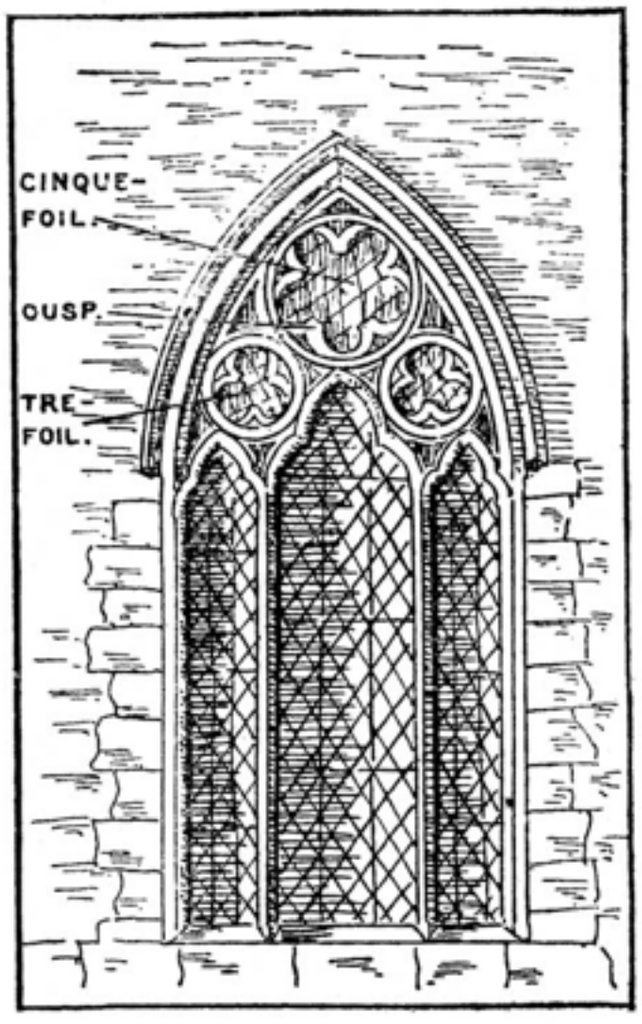
The Architecture of Bourges Cathedral



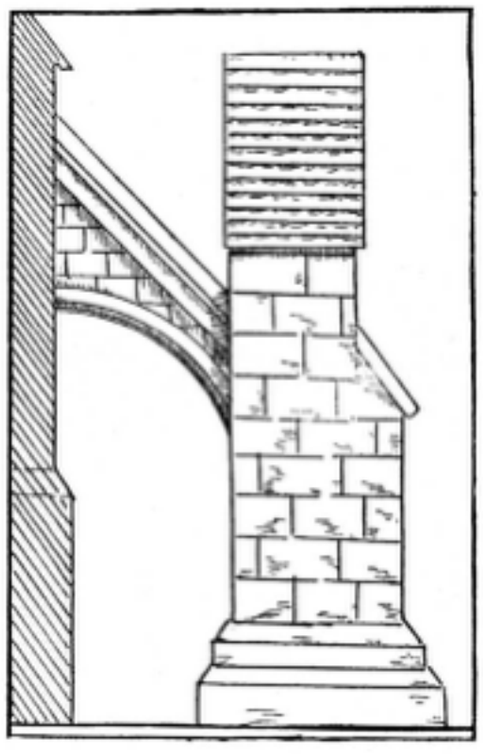








The Gothic Architectural Style



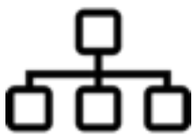
What are some of the well-known **architectural styles?**

- The client-server architectural style
- The 3-tiered architectural style
- **The multi-tiered architectural style**
- The RESTful architectural style
- The service oriented architectural style
- The event-driven architectural style
- The layered architectural style

Architectural styles are **not mutually exclusive**. Many applications combine patterns from several styles!

3-tiers, multi-tiers… **how many tiers** does a multi-tiered application have, and which ones?

human-readable
markup generation

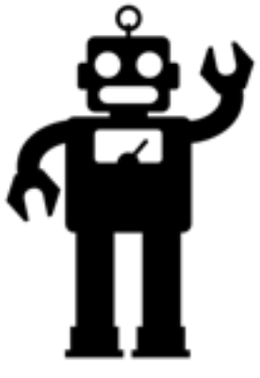


Page navigation

Data access

Shared business
logic and rules

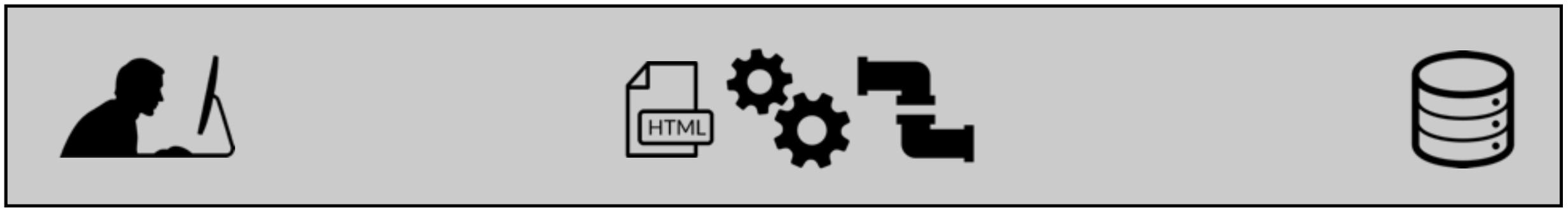
Message/
event bus



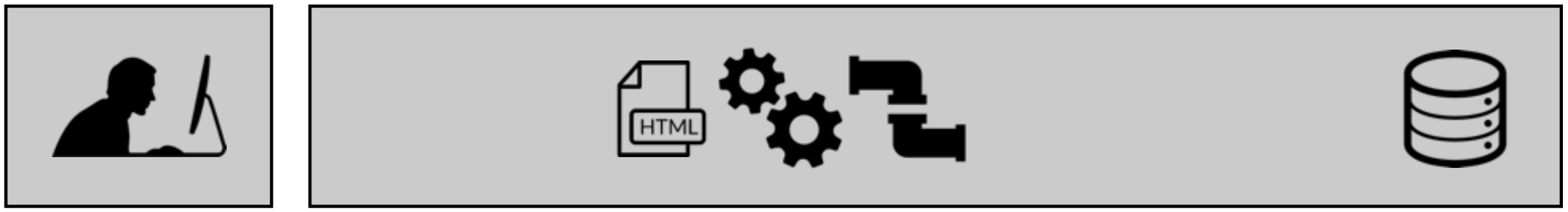
machine-readable
markup generation

User	Client	Presentation	Business	Integration	Resources

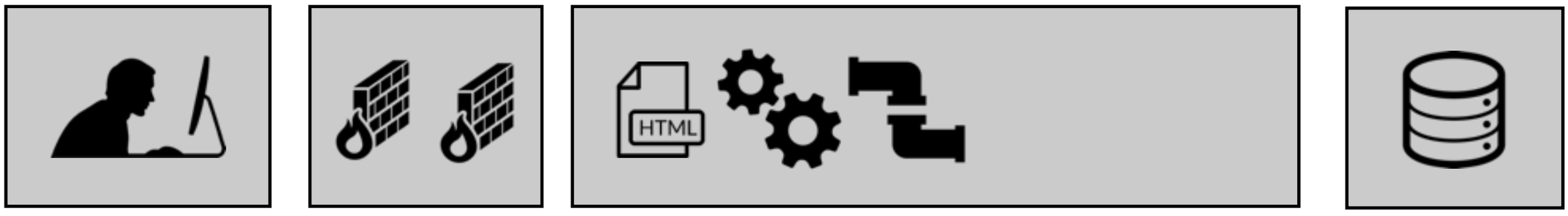
What is the mapping between tiers, **logical nodes** and **physical nodes**?



Dev environment



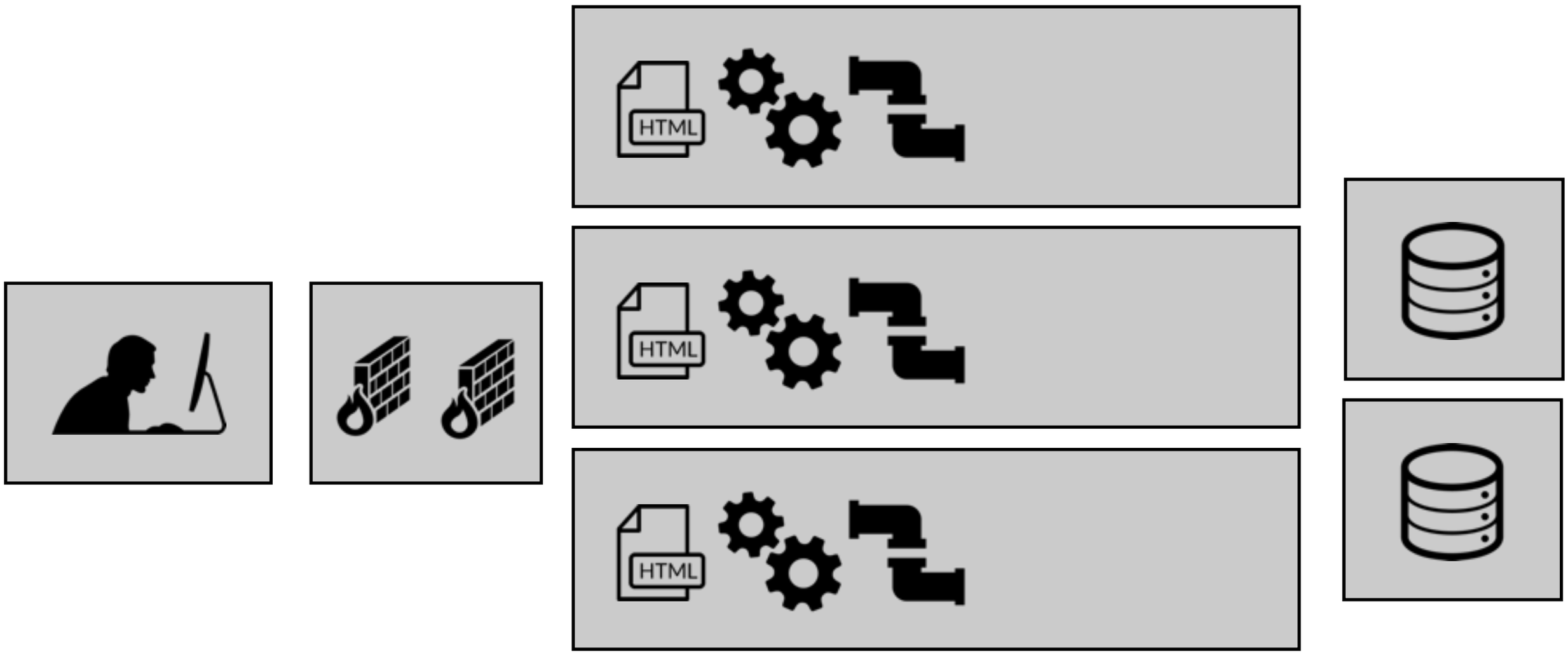
Minimal environment



Simple environment

User | **Client** | **Presentation** | **Business** | **Integration** | **Resources**

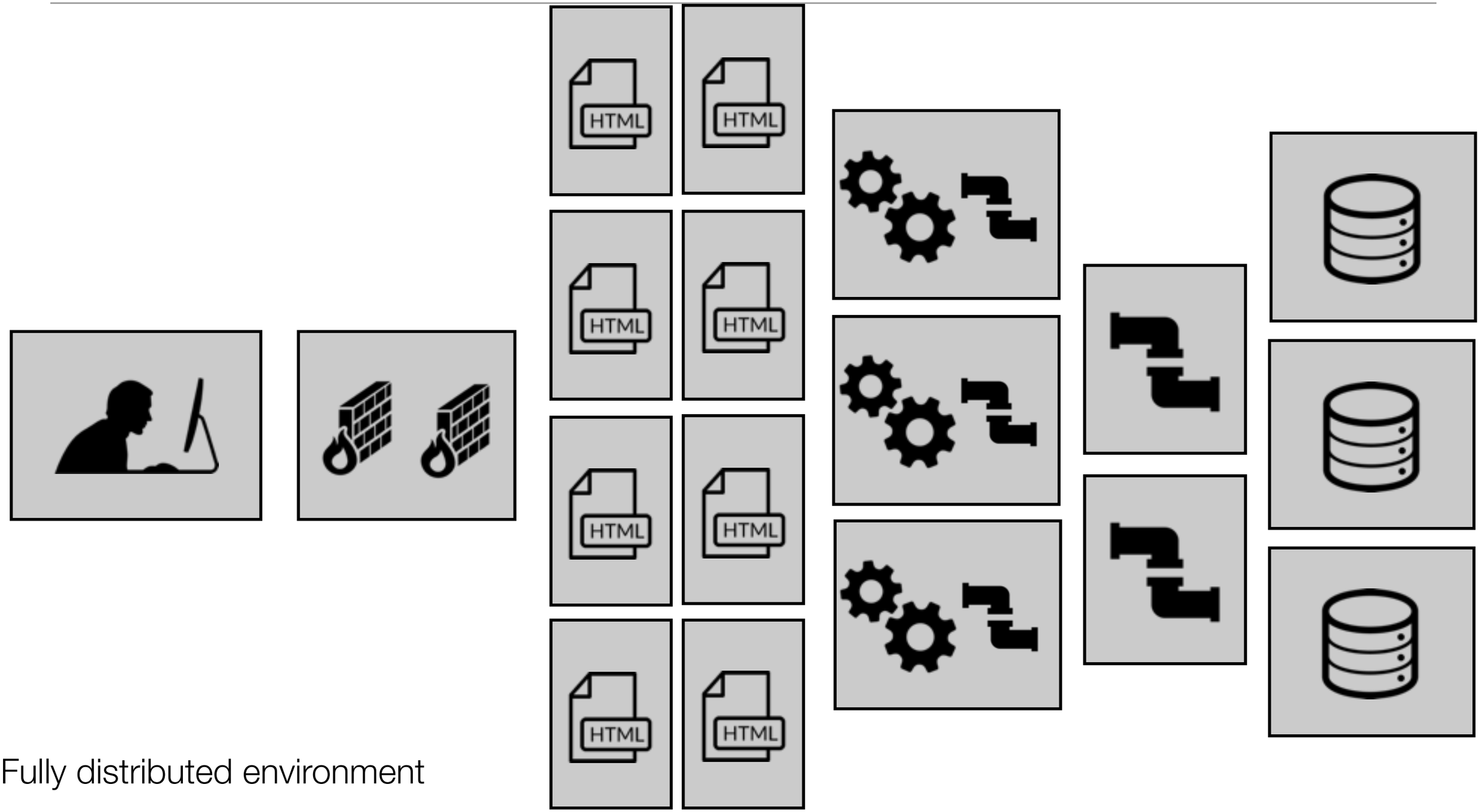
What is the mapping between tiers, **logical nodes** and **physical nodes**?



Typical environment for scalability and availability

User | **Client** | **Presentation** | **Business** | **Integration** | **Resources**

What is the mapping between tiers, **logical nodes** and **physical nodes**?



Fully distributed environment

User | **Client** | **Presentation** | **Business** | **Integration** | **Resources**

What is the mapping between tiers, **logical nodes** and **physical nodes**?

- Just because it is **possible to physically separate the presentation and the business tier** does not mean that it is always a good idea!
- One reason to keep them in the same node is that **invoking a remote service is a lot slower than invoking a local service**.
- Introduce a **reverse proxy** as soon as possible. In simple environments (e.g. dev.), it does not require a physical node.
- **Database nodes** are often shared by several applications. Optimization techniques are fairly different for database and application nodes. They are often physically separated.

(Rapidly evolving) virtualization technologies gives us more options. Think about **Docker**, which allows you to “run a data center” on your laptop.

heig-vd

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

Java Enterprise Edition

What is Java Enterprise Edition?

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

- 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	BEA Systems	Borland Software Corporation
Capgemini	Chandrasekaran, Muralidharan	Crawford, Scott
Dudney, Bill	E.piphany, Inc.	Genender, Jeff
Hewlett-Packard	IBM	Ironflare AB
Kohen, Elika S.	Leme, Felipe	Novell, Inc.
Oracle	OW2	Pramati Technologies
Pratap, Rama Murthy Amar	Raible, Matt	Red Hat Middleware LLC
Reinshagen, Dirk	SAP AG	SeeBeyond Technology Corp.
Shah, Suneet	Sun Microsystems, Inc.	Sybase
Tiwari, Ashish	Tmax Soft, Inc.	Trifork
Umapathy, Sivasundaram		

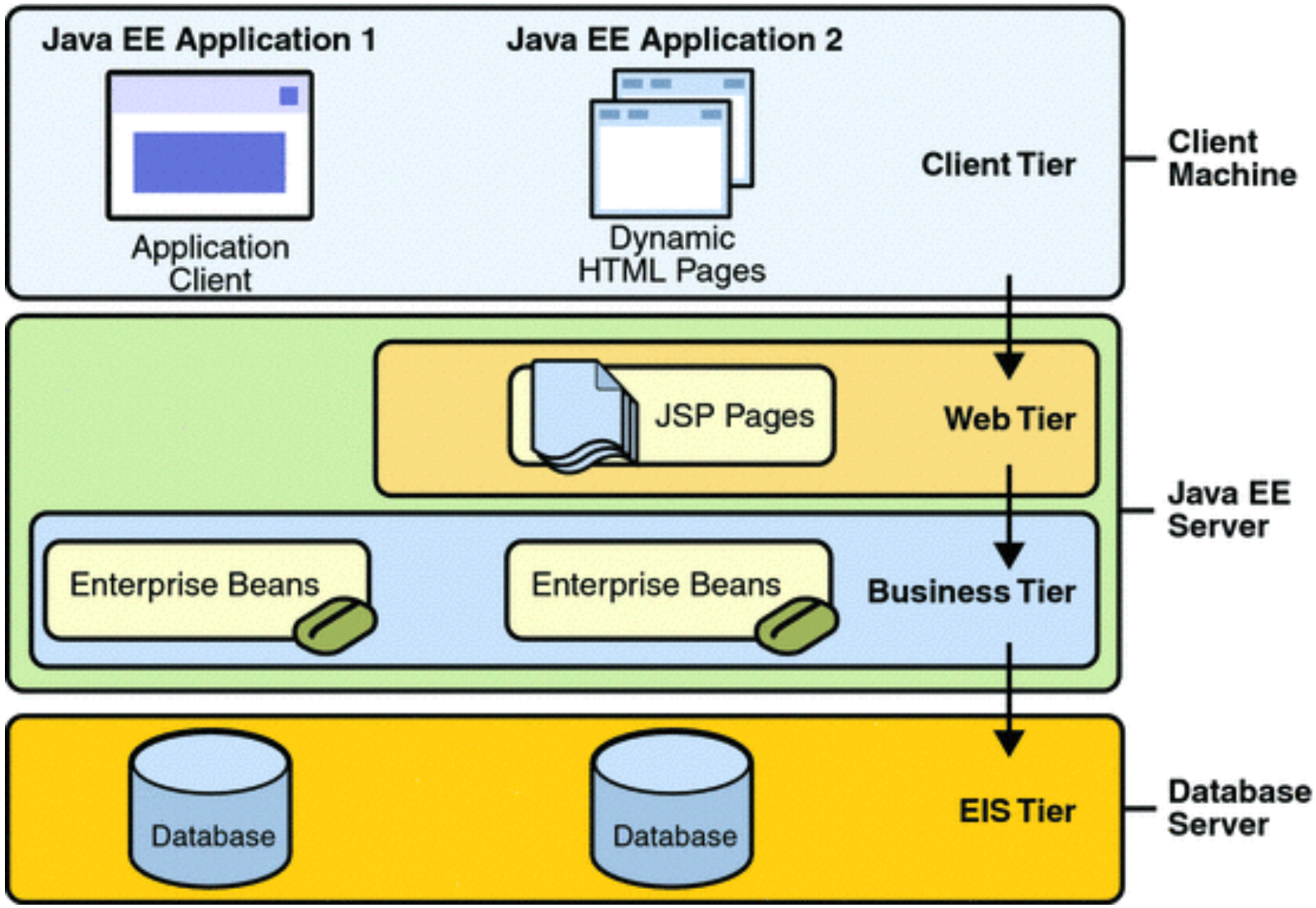
J2EE or Java EE?

- 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...
- Java EE 7 has been adopted on May 28th, 2013
- Java EE 8 is underway…

Watch out! Java EE (formerly known as J2EE) **has evolved a lot** over the years!

Architecture

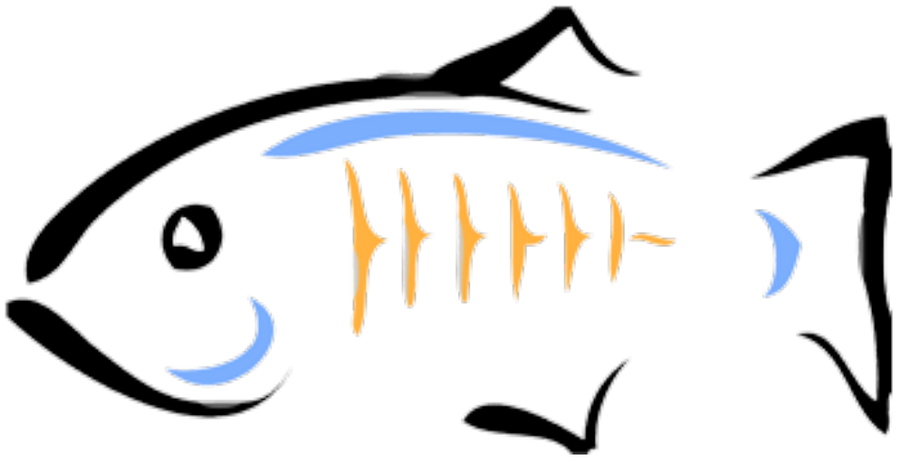
- The software that implements the Java EE specification is called an "**application server**"
  - There are **open source** and **proprietary** application servers.
  - Glassfish, JBoss, WebSphere, BEA WebLogic are examples of application servers.
    - Editors compete on aspects that are not defined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

heig-vd

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



Glassfish

WildFly

JBoss by Red Hat

IBM WebSphere software

ORACLE WebLogic

APACHE GERONIMO

Let's have a first look at Glassfish…

- **What do I find in the file system?**
  - binaries, domains, logs, deployed applications
- **How do I manage and configure my Glassfish environment?**
  - The web console (localhost:4848, if default port is used)
  - The mighty `asadmin` command line tool
  - The RESTful API: http://localhost:4848/management/domain, http://localhost:4848/monitoring/domain
  - Some things can be done via Netbeans too
- **Where is the documentation?**
  - https://glassfish.java.net/documentation.html
  - `asadmin help`

GlassFish Console – Comm ×

localhost:4848/common/index.jsf

Home | About... | Help

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

GlassFish™ Server Open Source Edition

Tree

- Common Tasks
  - Domain
  - server (Admin Server)
  - Clusters
  - Standalone Instances
  - Nodes
  - Applications
  - Lifecycle Modules
  - Monitoring Data
  - Resources
    - Concurrent Resources
    - Connectors
    - JDBC
    - JMS Resources
    - JNDI
    - JavaMail Sessions
    - Resource Adapter Configs
  - Configurations
    - default-config
    - **server-config**
  - Update Tool

GlassFish Console - Common Tasks

GlassFish News

- GlassFish News

Deployment

- List Deployed Applications
- Deploy an Application

Administration

- Change Administrator Password
- List Password Aliases

Monitoring

- Monitoring Data

Documentation

- Open Source Edition Documentation Set
- Quick Start Guide
- Administration Guide
- Application Development Guide
- Application Deployment Guide

Update Center

- Installed Components
- Available Updates
- Available Add-Ons

Resources

- Create New JDBC Resource
- Create New JDBC Connection Pool

Applications

localhost:4848/common/index.jsf

Home | About... | Help

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

GlassFish™ Server Open Source Edition

Tree

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

Applications

Applications can be enterprise or web applications, or various kinds of modules. Restart an application or module by clicking on the reload link, this action will apply only to the targets that the application or module is enabled on.

Deployed Applications (1)

Deploy... | Undeploy | Enable | Disable | Filter:

Select	Name	Deployment Order	Enabled	Engines	Action
☐	HelloWebApp	100	✓	web	Launch | Redeploy | Reload

You *can* deploy .war files via the web console. In practice, it's not really a common thing to do…

Monitoring Service

localhost:4848/common/index.jsf

Home | About... | Help

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

GlassFish™ Server Open Source Edition

Tree

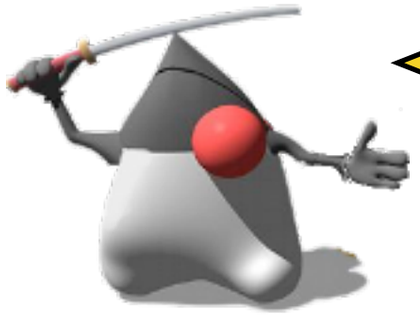
- Common Tasks
  - Domain
  - server (Admin Server)
  - Clusters
  - Standalone Instances
  - Nodes
  - Applications
  - Lifecycle Modules
  - Monitoring Data
  - Resources
    - Concurrent Resources
    - Connectors
    - JDBC
    - JMS Resources
    - JNDI
    - JavaMail Sessions
    - Resource Adapter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

Monitoring

View and manage the monitoring information for GlassFish Server instances.

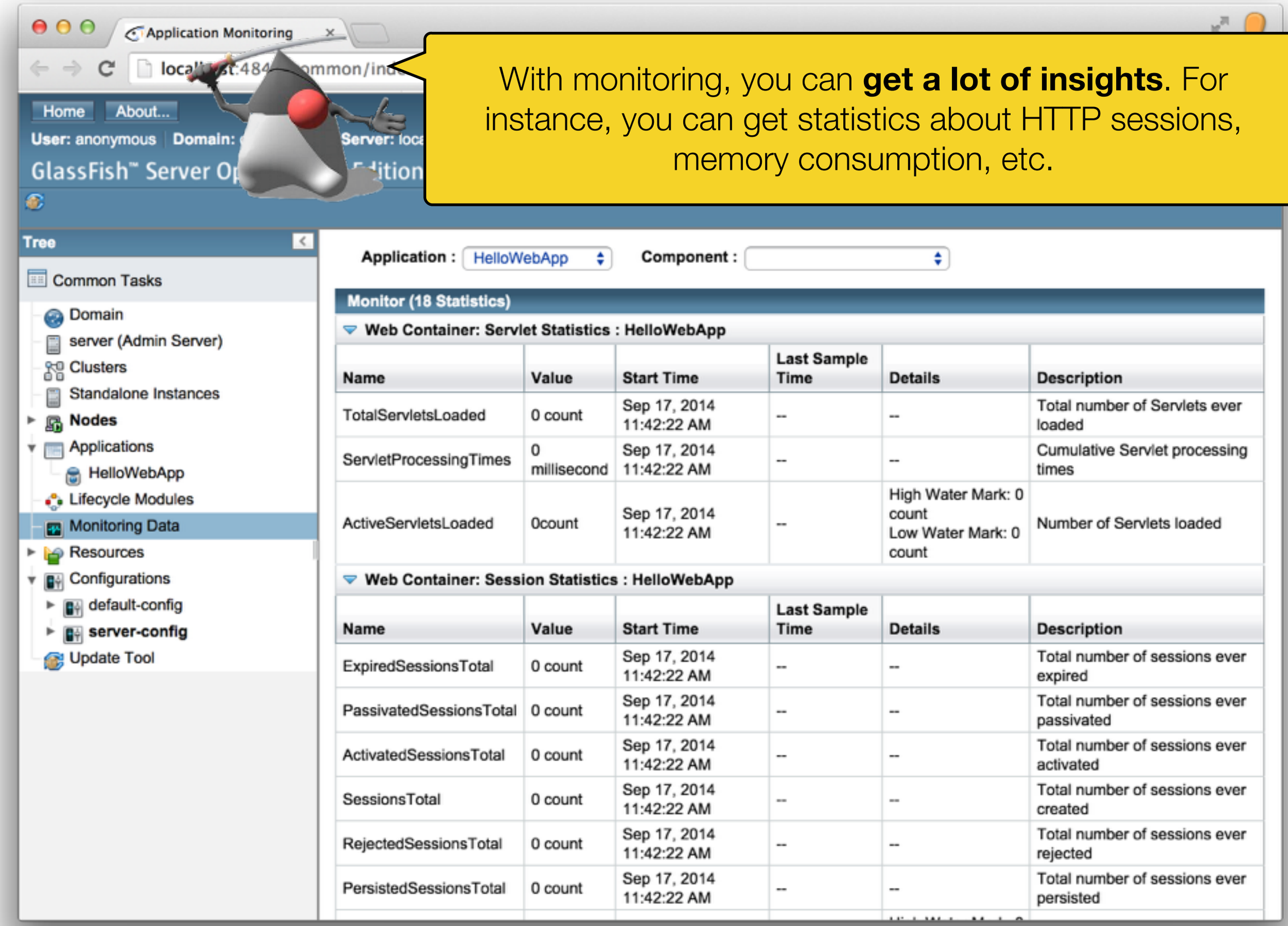
Monitoring (1)

Instance Name	Cluster Name	Action	View Monitoring Data
server	N/A	Configure Monitoring	Application , Server , Resources



If you want to **collect metrics** (performance, memory, etc.), you need to **activate monitoring**!

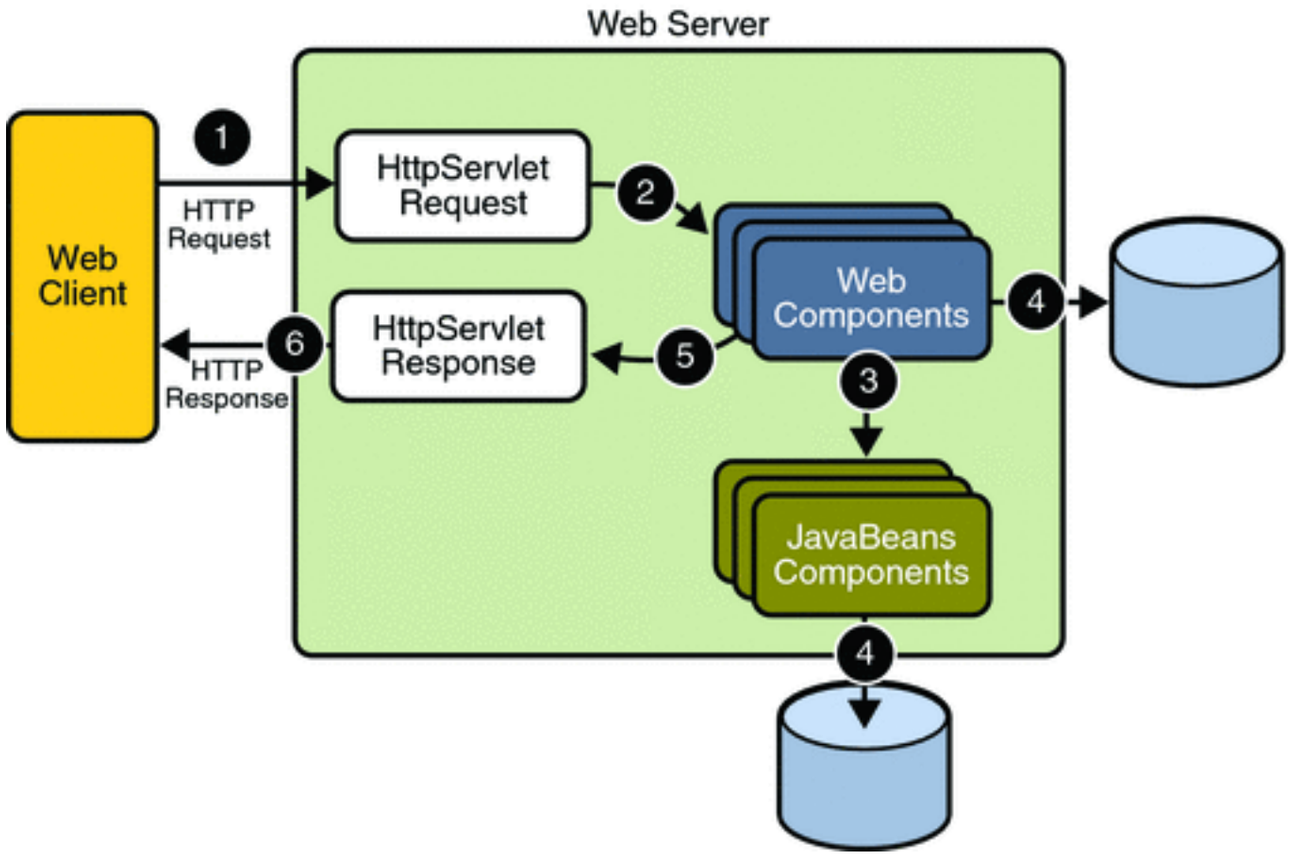
localhost:4848/common/monitor/monitoringPage.jsf?configName=server-config


Application Monitoring
Home
About...
User: anonymous
Domain:
Server:
GlassFish™ Server Op
dition
With monitoring, you can **get a lot of insights**. For instance, you can get statistics about HTTP sessions, memory consumption, etc.
Tree
Common Tasks
Domain
server (Admin Server)
Clusters
Standalone Instances
Nodes
Applications
HelloWebApp
Lifecycle Modules
Monitoring Data
Resources
Configurations
default-config
server-config
Update Tool
Application : HelloWebApp
Component :
Monitor (18 Statistics)
Web Container: Servlet Statistics : HelloWebApp
Name | Value | Start Time | Last Sample Time | Details | Description
TotalServletsLoaded | 0 count | Sep 17, 2014 11:42:22 AM | -- | -- | Total number of Servlets ever loaded
ServletProcessingTimes | 0 millisecond | Sep 17, 2014 11:42:22 AM | -- | -- | Cumulative Servlet processing times
ActiveServletsLoaded | 0count | Sep 17, 2014 11:42:22 AM | -- | High Water Mark: 0 count Low Water Mark: 0 count | Number of Servlets loaded
Web Container: Session Statistics : HelloWebApp
Name | Value | Start Time | Last Sample Time | Details | Description
ExpiredSessionsTotal | 0 count | Sep 17, 2014 11:42:22 AM | -- | -- | Total number of sessions ever expired
PassivatedSessionsTotal | 0 count | Sep 17, 2014 11:42:22 AM | -- | -- | Total number of sessions ever passivated
ActivatedSessionsTotal | 0 count | Sep 17, 2014 11:42:22 AM | -- | -- | Total number of sessions ever activated
SessionsTotal | 0 count | Sep 17, 2014 11:42:22 AM | -- | -- | Total number of sessions ever created
RejectedSessionsTotal | 0 count | Sep 17, 2014 11:42:22 AM | -- | -- | Total number of sessions ever rejected
PersistedSessionsTotal | 0 count | Sep 17, 2014 11:42:22 AM | -- | -- | Total number of sessions ever persisted

Watch out! You **will** encounter problems. It is a fact of life. Sometimes because of your code, and sometimes because of the environment (glassfish, etc.).

Your first reaction **must be** to check the Glassfish log.
More than this, you **must always** have a terminal window running a
`tail -f server.log`

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



Presentation Tier & MVC

What is the ~~ugliest~~ simplest way to process an HTTP request in Java EE?

- HTTP requests sent by clients are received by the application server.
- The application server looks at the first element in the URL path to figure out **which application** should process the request. A **servlet mapping** is then used to figure out **which servlet** should process the request.
- The servlet has access to a **request** and a **response objects**. It can access **HTTP data** (URL, headers, payload) via these objects. The servlet can generate an HTML and send it back via the response object.

```
@WebServlet("/greeting")                          GET /theApp/greeting HTTP/1.1
public class GreetingServlet extends HttpServlet {

  @Override
  public void doGet(HttpServletRequest request, HttpServletResponse response)
                    throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<html lang=\”en\”>”);
    out.println("It is a bad idea to write HTML in a servlet. ”);
    out.println("I will never, ever, do that.</html>”);
    out.close();
}
```

javax.servlet.http

Class HttpServlet

java.lang.Object
    javax.servlet.GenericServlet
        javax.servlet.http.HttpServlet

All Implemented Interfaces:

Serializable, Servlet, ServletConfig

```
public abstract class HttpServlet
extends GenericServlet
```

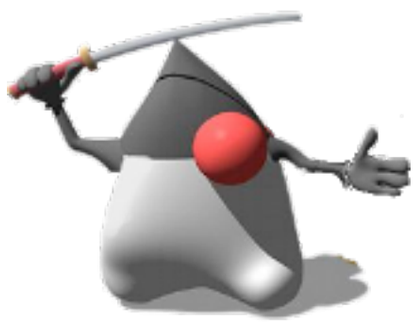
Provides an abstract class to be subclassed to create an HTTP servlet suitable for a Web site. A subclass of `HttpServlet` must override at least one method, usually one of these:

- `doGet`, if the servlet supports HTTP GET requests
- `doPost`, for HTTP POST requests
- `doPut`, for HTTP PUT requests
- `doDelete`, for HTTP DELETE requests
- `init` and `destroy`, to manage resources that are held for the life of the servlet
- `getServletInfo`, which the servlet uses to provide information about itself

There's almost no reason to override the `service` method. `service` handles standard HTTP requests by dispatching them to the handler methods for each HTTP request type (the `do`*XXX* methods listed above).

Likewise, there's almost no reason to override the `doOptions` and `doTrace` methods.

Servlets typically run on multithreaded servers, so be aware that a servlet must handle concurrent requests and be careful to synchronize access to shared resources. Shared resources include in-memory data such as instance or class variables and external objects such as files, database connections, and network connections. See the Java Tutorial on Multithreaded Programming for more information on handling multiple threads in a Java program.



What's wrong with this approach?

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

- There is **no separation of concerns**: the presentation and business logic is coded in the same module.
- The code is **hard to read** and to **maintain**.
- It is **not possible to share and reuse business logic** (the service) across different views. What happens when a business service needs to be accessed via a browser and a native mobile application?
- It is **not possible to distribute the work** between the back-end and the front-end developer.
- How do we implement and enforce **page navigation logic** in this scenario?

The Model-View-Controller (MVC) pattern

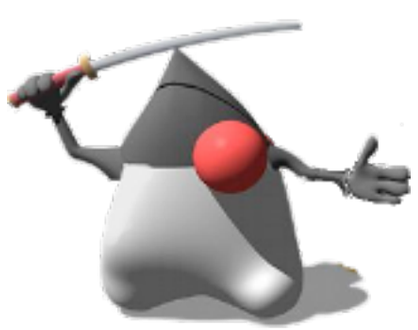
heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- The MVC design pattern has initially been developed in **native GUI toolkits**:
  - The **model** is responsible to capture the state and behavior of a business object.
  - The **view** is responsible for rendering the model and present it to the user.
  - The **controller** is responsible to **react to events** (mouse, keyboard, etc.). In reaction to events, it invokes operations on the model (which changes its state) and asks

What does MVC mean in Web Apps?

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

- The **model** is still responsible to capture the state (and behavior) of a business object.
- The **view** is responsible for rendering the model and present it to the user. Hence, the view is generating HTML, JSON, XML, PNG.
- The **controller** is responsible to **react to events**.
  - An event originates with a **user action** (clicking on a link, submitting a form, typing in a URL).
  - It is **encoded in an HTTP request** (with a URL, query string params, headers).



In web apps, **MVC can be implemented both on the server side and on the client side** (javascript frameworks). Here, we are talking about server-side MVC.

How are responsibilities shared in the MVC design pattern?

---

model

I am (domain) ***data***

view

I know how to ***present data*** *in a specific way.*

controller

I know ***what data*** *is needed.*

I know ***how*** *to generate data.*

I know ***which view*** *can present the data*

How can the MVC design pattern be implemented with Java EE technologies?

---

model

*I am a **JavaBean***

view

*I am a **JSP** page*

controller

*I am a **servlet***

*I can do the work myself or, better, **delegate** it to a **service**.*

*I know how to **delegate** work to a JSP*

How can the MVC design pattern be implemented with Java EE technologies?

---

client

GET /theApp/command

request

response

application server

request

response

request

response

servlet

req.setAttribute(“model”, bean)

request

response

JSP

req.getRequestDispatcher(“WEB-INF/views/home.jsp”).forward(req, res);

<h1>${requestScope.model}</h1>

bean = …

bean

What is the ~~ugliest~~ simplest way to process an HTTP request in Java EE?

Servlet <<controller>>

```
@WebServlet("/greeting")
public class GreetingServlet extends HttpServlet {

  @Override
  public void doGet(HttpServletRequest request, HttpServletResponse response)
                    throws ServletException, IOException {
    String message = "A First Java EE Web App";
    request.setAttribute("message", message);
    request.getRequestDispatcher("WEB-INF/views/home.jsp").forward(request, response);
}
```

JSP <<view>>

```
<div class="page-header">
  <h1>${requestScope.message}</h1>
</div>
```

Object <<model>>

“A First Java EE Web App”

If you read the Java EE 7 Tutorial, you will read about **JSF**… but where is the **JSP** section???

- **Java Server Pages** (JSP) is a core Java EE technology since the early days. It still is fully supported by application servers (backward compatibility).
- At some stage, **Java Server Faces** (JSF) was added as a complementary presentation tier API. The promise was to offer a component-oriented programming model.
- **JSF has never gained broad adoption** (issues in the first versions of the spec, complexity, alternatives, growing popularity of javascript frameworks combined with REST APIs, etc.).
- Nevertheless, **application server vendors** are still trying to push the standard. They **would like** JSF to be the technology that developers use in the web tier…
- You simply need to do a bit of “documentation archeology” (Java EE 5):
  - http://docs.oracle.com/javaee/5/tutorial/doc/bnagx.html
  - https://jstl.java.net/

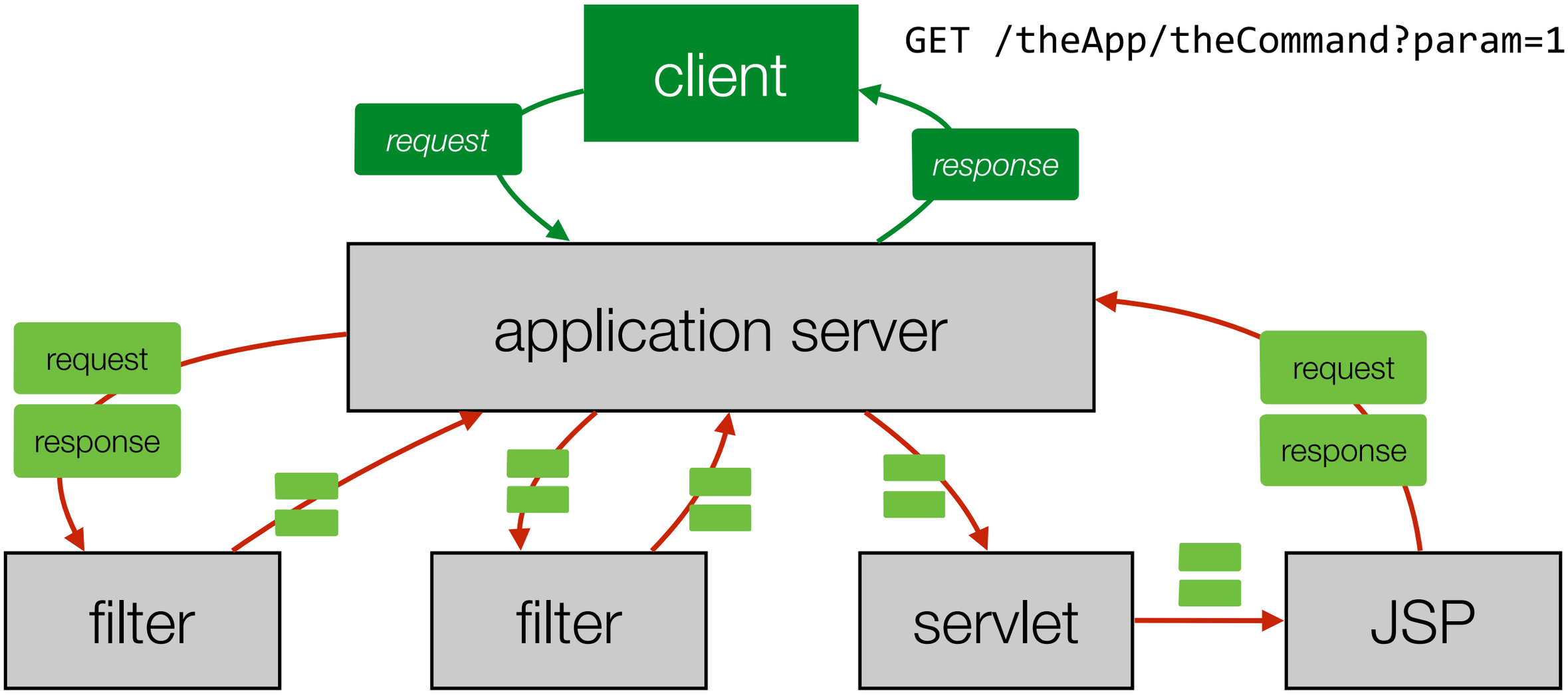
HTTP requests are processed in a **pipeline**. What does it mean?

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

- HTTP requests sent by clients are received by the application server.
- Remember that several applications may be deployed in the application server. Each application has an associated **context**, which is a URL prefix.
- The application server **inspects the URL**. Based on its prefix, it is responsible to find the first application component that will initiate the processing of the request. Typically, this will be a servlet or a servlet filter.
- In well structured applications, **several components are involved in the processing of each HTTP request**. At the very least, a controller is responsible to invoke the appropriate service (based on URL path and parameters) and a view is responsible to present the data in a particular format. In more complex scenarios, several components (**filters**) may apply some processing in sequence (security checks, logging, compression, etc.).
- The components involved in the processing are organized in a pipeline. The request and the response object are **passed** from one component to the other.

HTTP requests are processed in a **pipeline**. What does it mean?

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




```
public void doFilter(ServletRequest request,
ServletResponse response, FilterChain chain)
throws IOException, ServletException {

  chain.doFilter(request, response);

}
```

```
public void doGet(HttpServletRequest req,
HttpServletResponse res)
throws ServletException, IOException {

  req.getRequestDispatcher("WEB-INF/views/
home.jsp").forward(req, res);
}
```

javax.servlet

Interface Filter

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

`public interface` **`Filter`**

A filter is an object that performs filtering tasks on either the request to a resource (a servlet or static content), or on the response from a resource, or both.

Filters perform filtering in the `doFilter` method. Every Filter has access to a FilterConfig object from which it can obtain its initialization parameters, and a reference to the ServletContext which it can use, for example, to load resources needed for filtering tasks.

Filters are configured in the deployment descriptor of a web application.

Examples that have been identified for this design are:

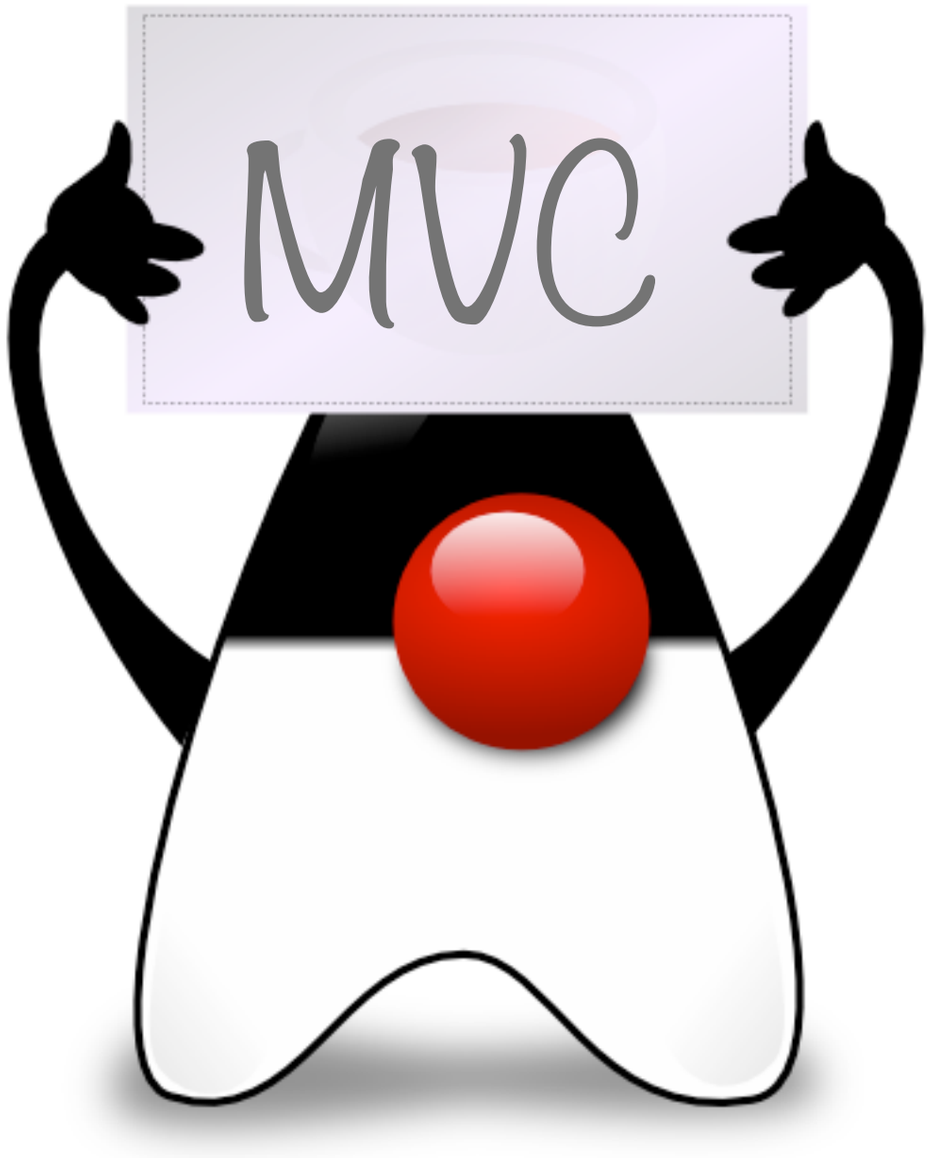
1. Authentication Filters
2. Logging and Auditing Filters
3. Image conversion Filters
4. Data compression Filters
5. Encryption Filters
6. Tokenizing Filters
7. Filters that trigger resource acces
8. XSL/T filters
9. Mime-type chain Filter

Method Summary

Methods

Modifier and Type	Method and Description
`void`	**`destroy()`** Called by the web container to indicate to a filter that it is being taken out of service.
`void`	**`doFilter(ServletRequest`** `request,` **`ServletResponse`** `response,` **`FilterChain`** `chain)` The `doFilter` method of the Filter is called by the container each time a request/response pair is passed through the chain due to a client request for a resource at the end of the chain.
`void`	**`init(FilterConfig`** `filterConfig)` Called by the web container to indicate to a filter that it is being placed into service.

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



Lab 01 - “Hello Tiers”

Create a **Java Web project** in Netbeans

- Select “**New Project**” in the File menu.
- Today, we will use the “**Java Web**” category (we will soon use maven projects).
- We will work with **Java EE 7**.

Create a **model** class (a JavaBean, a POJO)

- It is a **good practice** to keep all your model classes in the same package, so create a package `ch.heigvd.amt.lab1.model`.

- At this stage, a model class is a very simple one, with a collection of properties (private variables) and corresponding **constructor**, **getters** and **setters methods**. It is a **P**lain **O**ld **J**ava **O**bject (POJO).

- Create a model class that will represent a sensor measurement. A **`Measure`** has 3 properties: a **`sensorId`** (unique string ID), a **`timestamp`** (expressed in ms) and a **`value`** (numeric value).

NetBeans hint:
> Source > Insert code (CTRL-I)
gives you access to constructor and getter/setter wizards

Create a **service** class (a POJO)

- It is also a **good practice** to keep all your service classes in one same package, so create a package `ch.heigvd.amt.lab1.services`.

- In this first lab, the service will be very simple (it will merely **simulate** what a real service would do). Don't worry about storing data in a DB, this will come later.

- Add a **getMeasures()** method, which returns a list of Measure objects.

- Create a **dummy implementation**: generate a random number of measures and assign random values to their fields.

Create a **view** (a JSP page)

- Create a **views folder** in the WEB-INF directory (good for security: they will not be directly accessible, WEB-INF is protected!).
- Create a JSP page in the views folder and name it **measures.jsp**.
- **Add basic HTML content** in the JSP page. Copy the Bootstrap Sticky Footer source from http://getbootstrap.com/examples/sticky-footer/.
- **Download the linked CSS files** (bootstrap.min.css and ticky-footer.css) and place at the same level as the WEB-INF directory.

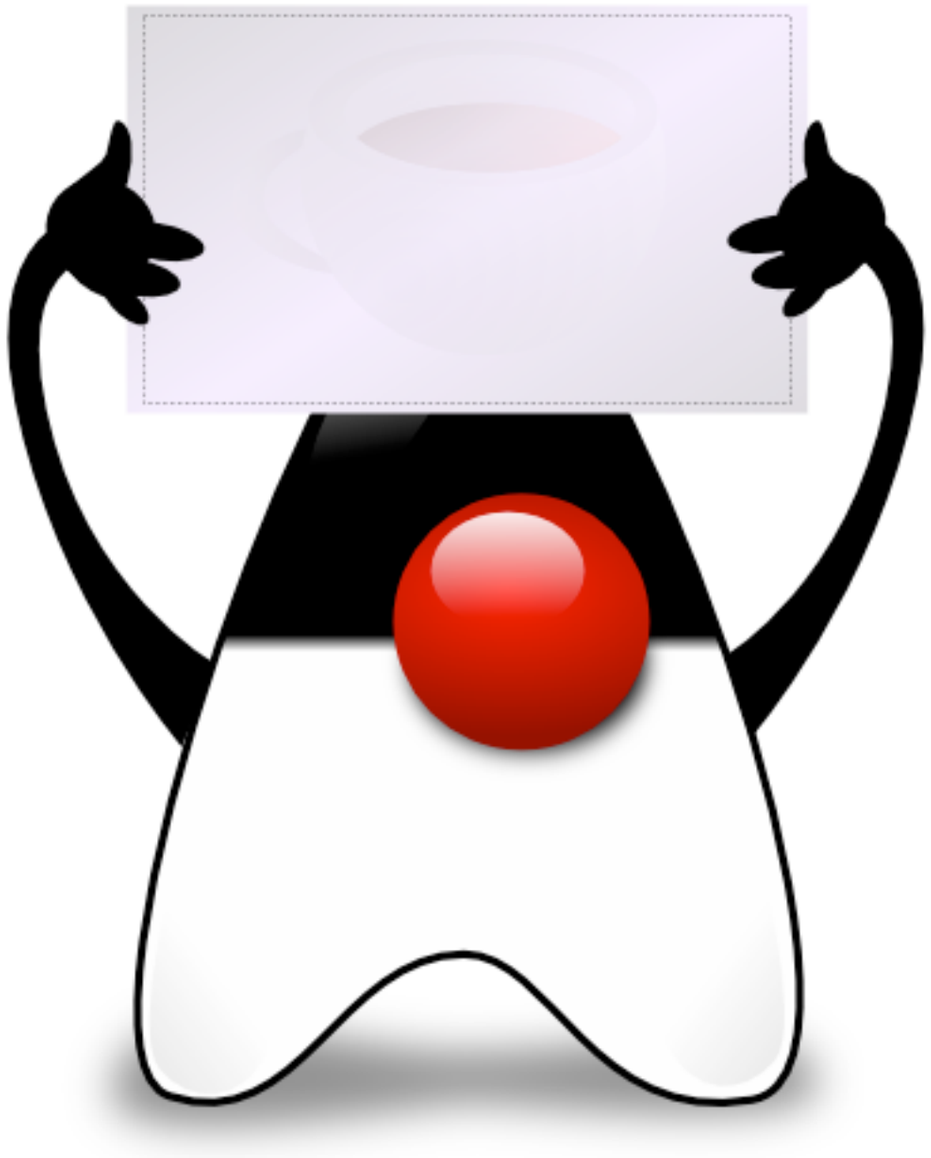
Create a **controller** class (a servlet)

- Create a package `ch.heigvd.amt.lab1.controllers` package in your project.
- Use the wizard to create a servlet and call it **FrontControllerServlet**.
- **Iteration 1**: modify the code, so that when the user invokes the corresponding URL, the servlet generates HTML directly (bad practice). Deploy and run.
- **Iteration 2**: modify the code, so that the servlet delegates view processing to your JSP page (without passing data). Deploy and run.
- **Iteration 3**: modify the code, so that the servlet stores a string message in the request scope. Modify the JSP page accordingly. Deploy and run.

Bind the controller, the service and the view

- In your servlet, create an instance of your service.
- In the request processing method, invoke the service to get a new model instance each time (i.e. a list of measures).
- Attach the model to the request scope.
- As before, delegate rendering to the JSP (no need to change the code).
- In your JSP, find a way to retrieve and iterate over the list of measures. Display them in a Bootstrap table.
  - http://docs.oracle.com/javaee/5/tutorial/doc/bnakh.html#bnakm
  - http://getbootstrap.com/css/#tables

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



Lecture - HTTP Sessions

How is it possible to use **HTTP sessions** in a Java EE application?

- Remember that HTTP is a **stateless protocol**, but that there are ways to implement stateful applications on top of it.
  - One technique is to rely on **cookies** (server receives a cookie from the client, and thereby recognizes him).
  - One technique is to rely on a session id **encoded in the URL**.
- The **Java EE API** hides the complexity of dealing with these issues. It exposes a notion of “session”, which works like magic.
- The **Java EE implementation** (i.e. the app server) deals with HTTP protocol details and handles the cookies, URL rewriting, etc.

How is it possible to use **HTTP sessions** in a Java EE application?

- The HTTP session is available both in servlets and in JSPs.
- With the API, it is possible to attach and retrieve attributes to the user session.
- With the API, it is also possible to get information about the current session.

Methods

Modifier and Type	Method and Description
`Object`	**`getAttribute(String name)`** Returns the object bound with the specified name in this session, or `null` if no object is bound under the name.
`Enumeration<String>`	**`getAttributeNames()`** Returns an `Enumeration` of `String` objects containing the names of all the objects bound to this session.
`long`	**`getCreationTime()`** Returns the time when this session was created, measured in milliseconds since midnight January 1, 1970 GMT.
`String`	**`getId()`** Returns a string containing the unique identifier assigned to this session.
`long`	**`getLastAccessedTime()`** Returns the last time the client sent a request associated with this session, as the number of milliseconds since midnight January 1, 1970 GMT, and marked by the time the container received the request.
`int`	**`getMaxInactiveInterval()`** Returns the maximum time interval, in seconds, that the servlet container will keep this session open between client accesses.

void	**invalidate**() Invalidates this session then unbinds any objects bound to it.
boolean	**isNew**() Returns `true` if the client does not yet know about the session or if the client chooses not to join the session.
void	**putValue**(**String** name, **Object** value) **Deprecated.** *As of Version 2.2, this method is replaced by* ***setAttribute(java.lang.String, java.lang.Object)***
void	**removeAttribute**(**String** name) Removes the object bound with the specified name from this session.
void	**removeValue**(**String** name) **Deprecated.** *As of Version 2.2, this method is replaced by* ***removeAttribute(java.lang.String)***
void	**setAttribute**(**String** name, **Object** value) Binds an object to this session, using the name specified.
void	**setMaxInactiveInterval**(int interval) Specifies the time, in seconds, between client requests before the servlet container will invalidate this session.

Why do we need to be **careful** when using HTTP sessions?

- HTTP sessions are **useful**. They are easy to **use** and **abuse**.
- Let's imagine an online shop application. I could fetch all product descriptions in the database and store it in the HTTP session.
  - If I do this, whenever the user sends an HTTP request, I have access to the data. No need to go back to the DB. Sounds like a smart thing to do?
  - If I implement this and test in "on my machine" with a single user, it is likely to work well (especially if my test DB contains 3 products…).
  - What happens if I get 100 **concurrent sessions** and my product descriptions represent 10 MB of memory?

Who needs to care about HTTP sessions?

- **The developers must care:**
  - They are responsible for **writing the code** and for deciding when it makes sense to store data in the session or not (it is always a **tradeoff**!).
  - They always need to **think about concurrency and scale**. What will happen under load and over time

- **The system administrators must care:**
  - They are responsible for **monitoring the app server and the app**. Monitoring makes you aware of potential issues with your app.
  - They are responsible for **configuring the application server**. One thing they can control is the **session timeout**. If a user has not sent any request within a given timeframe, then session is discarded and memory is reclaimed.
  - They are responsible for understanding the (proprietary) tuning parameters of the app server. For instance, Glassfish has a notion of **session passivation**. If available memory falls below a threshold, active sessions are transferred to disk (and brought back to RAM when the user sends a new request). Cool, but they must understand the performance impact.

HTTP sessions **consume memory**. How does Java EE **prevents memory exhaustion**? What about **Glassfish**?

7.5 Session Timeouts

In the HTTP protocol, there is no explicit termination signal when a client is no longer active. This means that the only mechanism that can be used to indicate when a client is no longer active is a time out period.

The default time out period for sessions is defined by the servlet container and can be obtained via the `getMaxInactiveInterval` method of the `HttpSession` interface. This time out can be changed by the Developer using the `setMaxInactiveInterval` method of the `HttpSession` interface. The time out periods used by these methods are defined in seconds. By definition, if the time out period for a session is set to -1, the session will never expire. The session invalidation will not take effect until all servlets using that session have exited the service method. Once the session invalidation is initiated, a new request must not be able to see that session.

count

Web Container: Session Statistics : HelloWebApp

Name	Value	Start Time	Last Sample Time	Details	Description
ExpiredSessionsTotal	0 count	Sep 17, 2014 11:42:22 AM	--	--	Total number of sessions ever expired
PassivatedSessionsTotal	0 count	Sep 17, 2014 11:42:22 AM	--	--	Total number of sessions ever passivated
ActivatedSessionsTotal	0 count	Sep 17, 2014 11:42:22 AM	--	--	Total number of sessions ever activated
SessionsTotal	0 count	Sep 17, 2014 11:42:22 AM	--	--	Total number of sessions ever created
RejectedSessionsTotal	0 count	Sep 17, 2014 11:42:22 AM	--	--	Total number of sessions ever rejected
PersistedSessionsTotal	0 count	Sep 17, 2014 11:42:22 AM	--	--	Total number of sessions ever persisted
ActiveSessions	0count	Sep 17, 2014 11:42:22 AM	--	High Water Mark: 0 count Low Water Mark: 0 count	Number of active sessions

Web Container: Request Statistics : HelloWebApp

Name	Value	Start Time	Last Sample Time	Details	Description
RequestCount	0 count	Sep 17, 2014 11:42:22 AM	Sep 18, 2014 8:22:45 AM	--	Cumulative number of requests processed so far
MaxTime	0 millisecond	Sep 17, 2014 11:42:22 AM	Sep 18, 2014 8:22:45 AM	--	Longest response time for a request; not a cumulative value, but the largest response time from among the response times
ErrorCount	0 count	Sep 17, 2014 11:42:22 AM	--	--	Cumulative value of the error count, with error count representing the number of cases where the response code was greater than or equal to 400
ProcessingTime	0 millisecond	Sep 17, 2014 11:42:22 AM	Sep 18, 2014 8:22:45 AM	--	Average request processing time

General | Session Properties | Manager Properties | Store Properties

Session Properties

Load Defaults

Save

Configuration Name: server-config

Session Timeout: 1800 Seconds

Maximum number of seconds that an inactive session remains valid. A value of 0 or less means that sessions never expire.

Additional Properties (0)

Add Property | Delete Properties

Select	Name	Value	Description
No items found.			

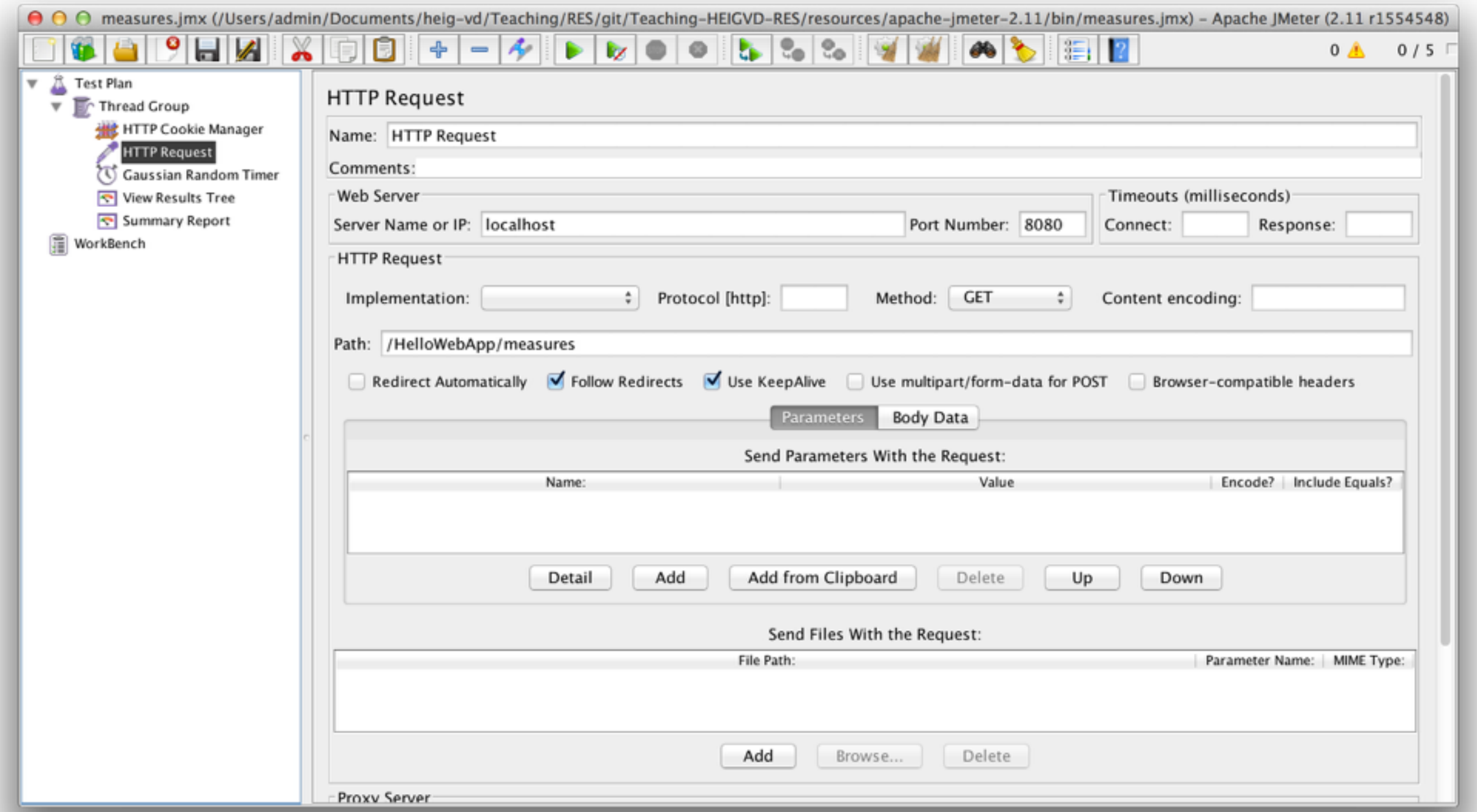
What do I need to to if I want to evaluate how my application behaves under load.

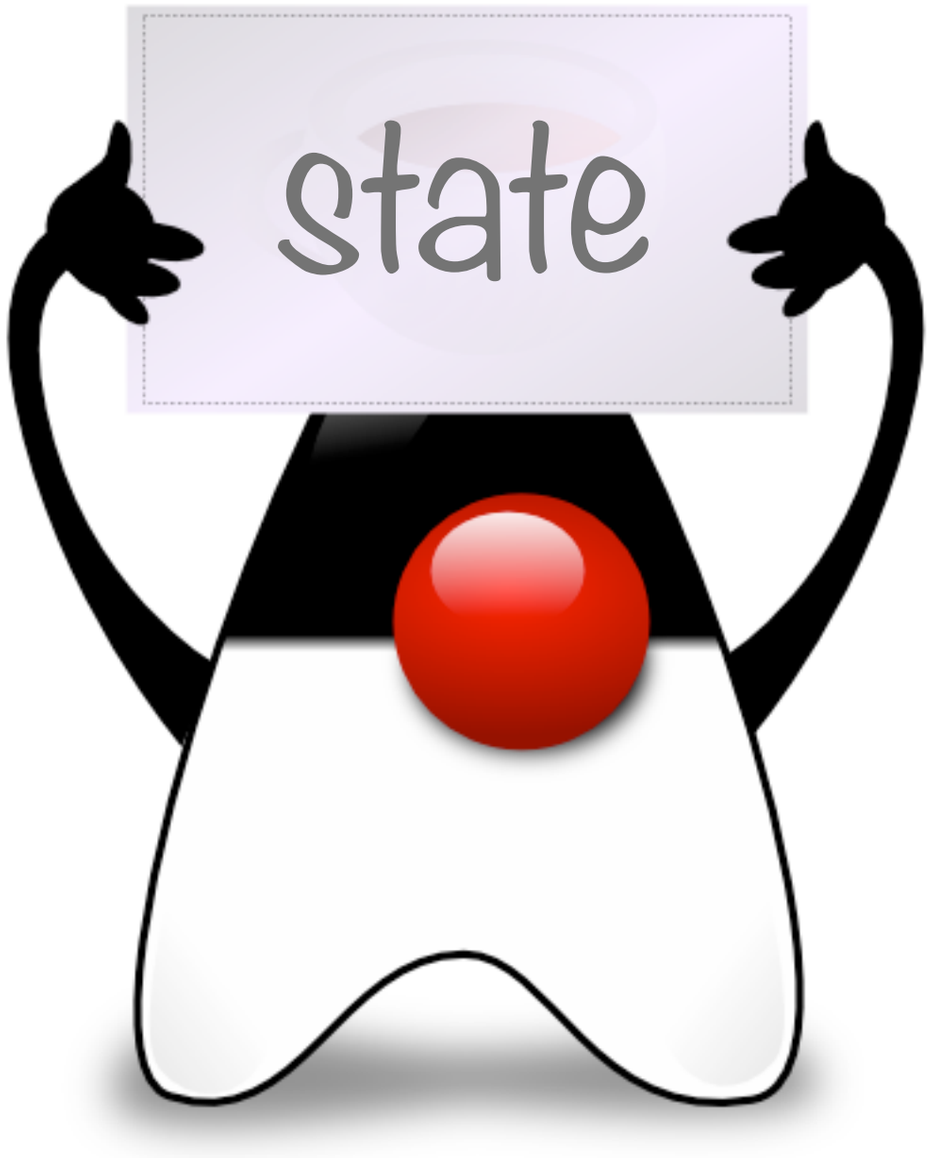
- **You need a test environment:**
  - For basic tests, you can work on your (single) machine.
  - For anything serious, you will need **at least** two machines. One for hosting the app, one for simulating the clients.
  - For real test campaigns, you should rent VMs from a cloud provider (e.g. Amazon Web Services, etc.)
- **You need a tool to inject load in your system:**
  - You can write your own scripts.
  - You can buy expensive load testing tools.
  - You can work with open source load testing tools, such as Apache JMeter.
- **You need tools / interfaces to collect metrics:**
  - On the client side (in the load testing tool)
  - On the server side, at the system level (e.g. `top`, etc.)
  - On the server side, at the application server level (e.g. Glassfish monitoring interfaces)

How do you setup JMeter to test how sessions behave in your web application?

- You create a **basic test plan**.
- You specify the number of virtual users in the **Thread Group**. You always start with 1, to validate your plan.
- You add an **HTTP Request sampler** to your test plan. You specify the target URL (e.g. you target a servlet that stores data in the session).
- If you run the test in a loop, you add a **Timer**, so avoid issues with your poor CPU.
- You add a couple of **listeners**. The View Tree listener will allow you to inspect the response of the server (and validate your script). The Summary report will give your performance metrics.
- You add a **HTTP Cookie Manager** to your testplan.

How do you setup JMeter to test how sessions behave in your web application?





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

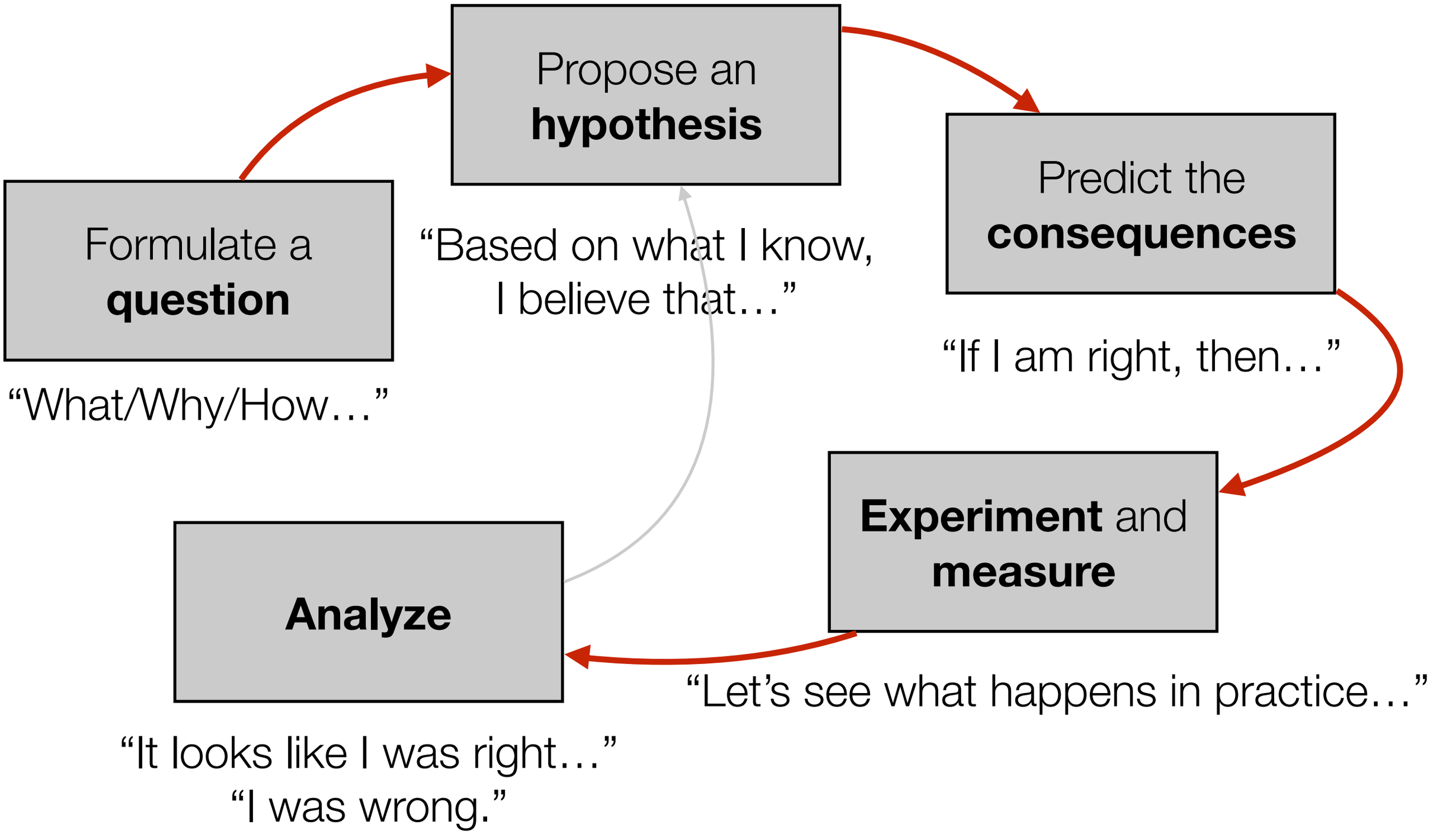
Lab 02 - HTTP Sessions

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**.

Scientific Method



Scientific Method

```
public void doSomething() {
  while (true) {
    Thread.sleep(1);
  }
}
```

Propose an **hypothesis**

"I believe that **this loop** is causing the issue."

Predict the **consequences**

"**If** I am right, **then** sleeping for 1000 ms should reduce CPU consumption."

Experiment and **measure**

"Let's run the program in the **2 configurations** and use the top command to **measure** CPU load."

Analyze

"CPU load is **still high**… even if this loop is a concern, it does not **seem** to be the cause of my problem."

Formulate a **question**

"**Why** is my program consuming all the CPU?"

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?

- Formulate your hypothesis. Based on what you have heard in this lecture, what do you think will happen as the number of sessions varies?
- Be precise.

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

- To verify if your hypothesis holds, you need to define what are would be the measurable effects of the experiment.
- What phenomenon will you observe, what will you measure and what do you expect this measures
- What are you going to measure? What do you expect these measures to show?

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

- To run the experiment, you will need to create an environment (remember what we said about having a test application, a load generator and monitoring tools/interfaces).
- Create a diagram to show how these elements interact with each other.
- Prepare a task list of the setup of your environment.
- Prepare a task list of the experiment itself.

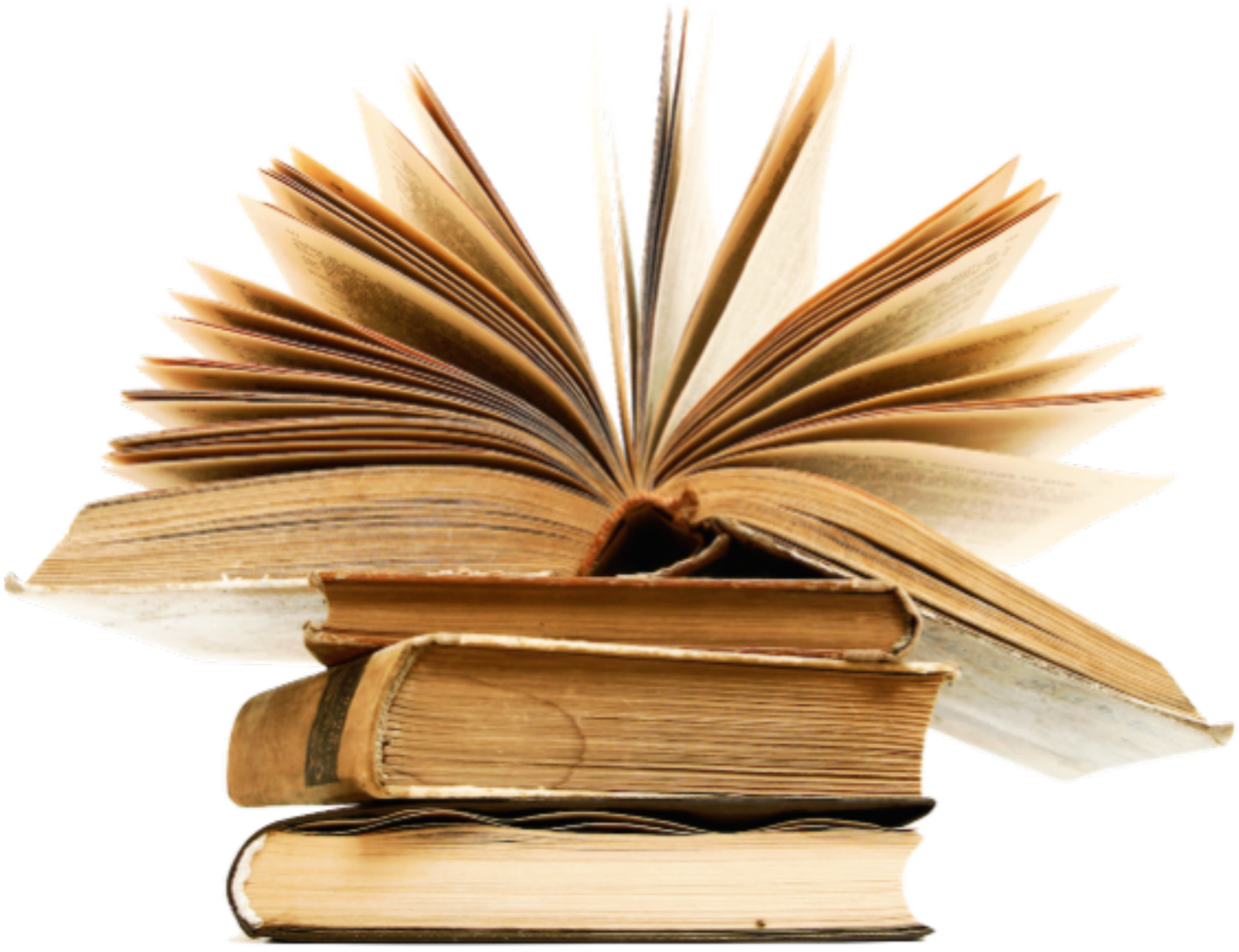
Run the experiment and collect the data

- Use your task list to setup and run the environment.
- Make sure that you collect and archive the collected metrics!

Analyze the data, draw your **conclusions** and **document** them!

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/overview002.htm#BNAAX
  - http://docs.oracle.com/javaee/7/tutorial/doc/overview003.htm#BNAAY
  - http://docs.oracle.com/javaee/7/tutorial/doc/overview004.htm#BNABO
  - http://docs.oracle.com/javaee/7/tutorial/doc/webapp001.htm#GEYSJ
  - http://docs.oracle.com/javaee/7/tutorial/doc/webapp004.htm#BNAEO
- **Selected sections from the Java EE 5 tutorial**
  - http://docs.oracle.com/javaee/5/tutorial/doc/bnafo.html
  - http://docs.oracle.com/javaee/5/tutorial/doc/bnagy.html
  - http://docs.oracle.com/javaee/5/tutorial/doc/bnahe.html
  - http://docs.oracle.com/javaee/5/tutorial/doc/bnahl.html
  - http://docs.oracle.com/javaee/5/tutorial/doc/bnahq.html
- **Design patterns**
  - http://www.oracle.com/technetwork/java/frontcontroller-135648.html
  - http://www.oracle.com/technetwork/java/interceptingfilter-142169.html

In a servlet, what **code** do you write to delegate request processing to a JSP page?

How do you make sure that a JSP page (home.jsp) cannot be fetched directly by a client (http://app/home.jsp is **refused**)

How does the application server know which servlet it should forward a request to?

How can different components (filters, servlets, JSPs) involved in the processing of an HTTP request **share data**? Describe the different **scopes**.

req is an instance of `HttpRequest`. Explain the difference between `req.getParameter()` and `req.getAttribute()`

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

What is the role of a **servlet mapping**? Give two examples of servlet mappings and explain how they work.

What is the difference between **architecture** and **architectural style**? What are 3 popular architectural styles?

Describe two ways to **manage and configure a Glassfish** environment.

What is the difference between a **servlet** and a **JSP**? Are there things that can only be implemented in a servlet? only in a JSP? Justify.

Java EE is an **umbrella specification**. What does it mean?

Alice is stating that **Apache Tomcat is a an application server**. Bob disagrees. Why are they both right?

Why is it important to **be aware of older Java EE versions** (and of the fact that APIs have evolved significantly?)

Why are many companies not immediately **adopting the latest edition** of the Java EE platform?

What is the difference between the **3-tiered** and the **multi-tiered architectural style**?