HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD

www.heig-vd.ch

02 - Presentation tier

MVC, IoC, Pipes and Filters patterns.
Servlets and JSPs.

AMT 2020

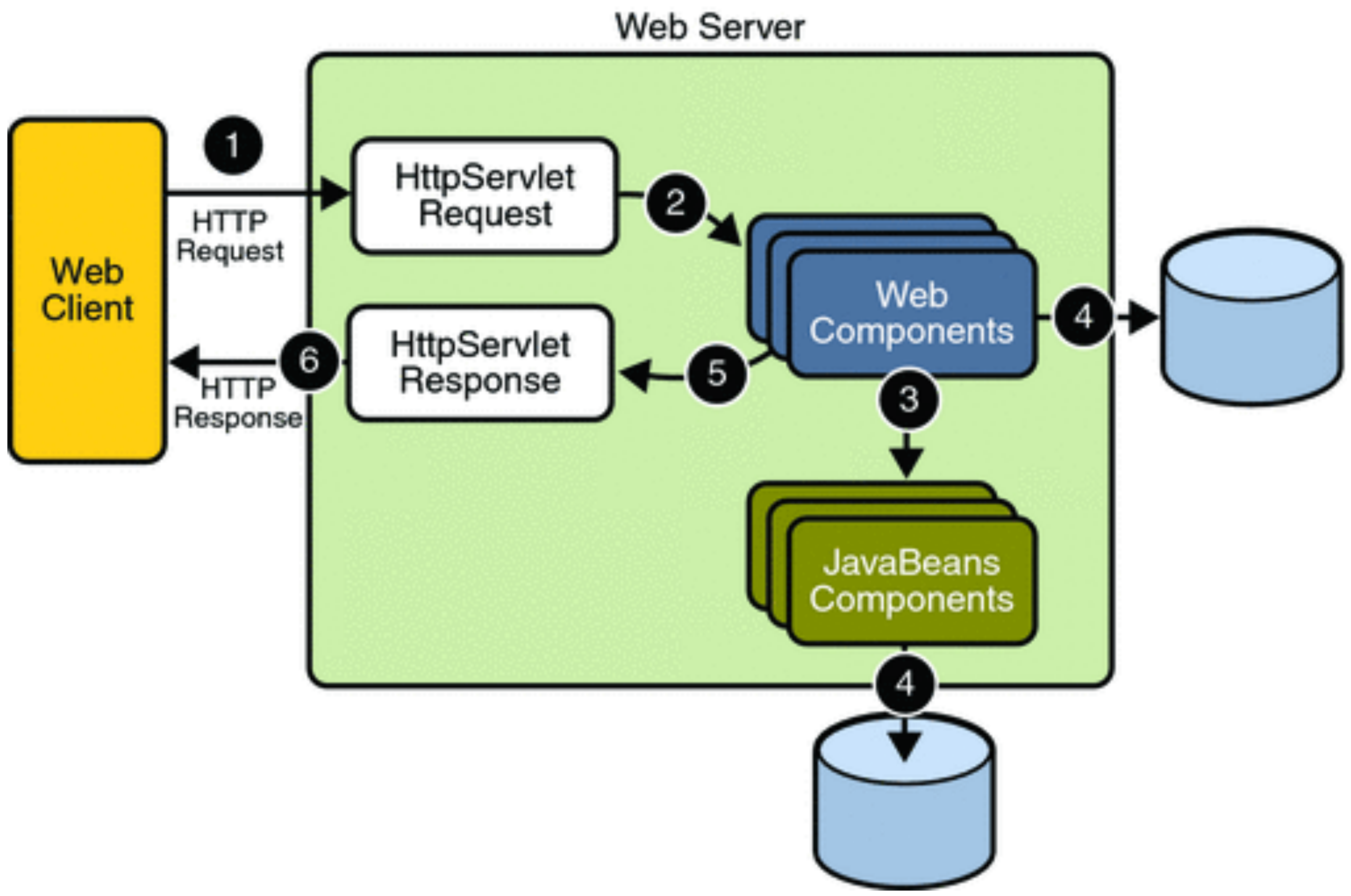
Olivier Liechti

Key takeaways

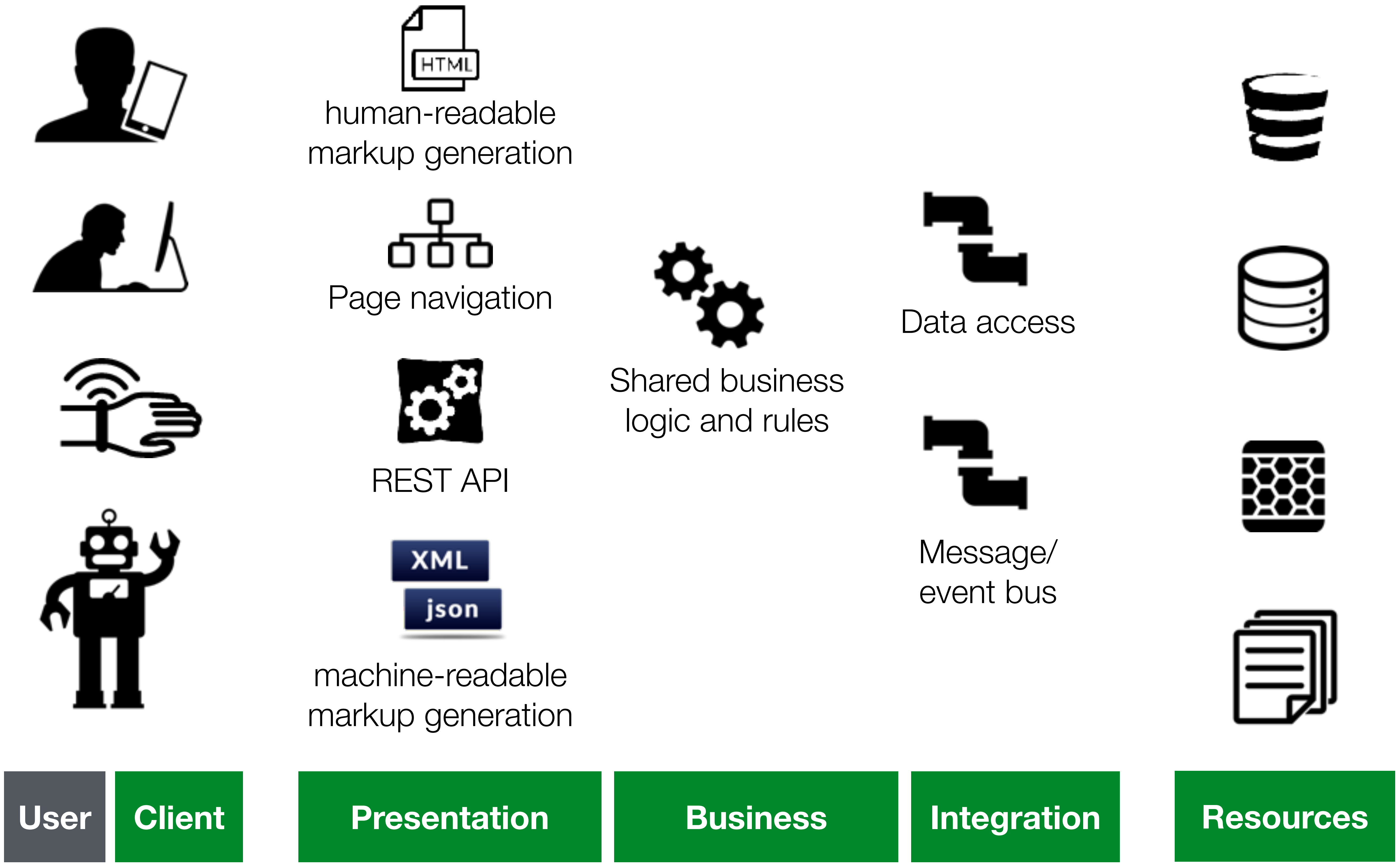
- MVC pattern
- IoC pattern
- Pipes and Filters pattern

Describe them in generic terms.
Apply them with Servlets and JSPs.

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch



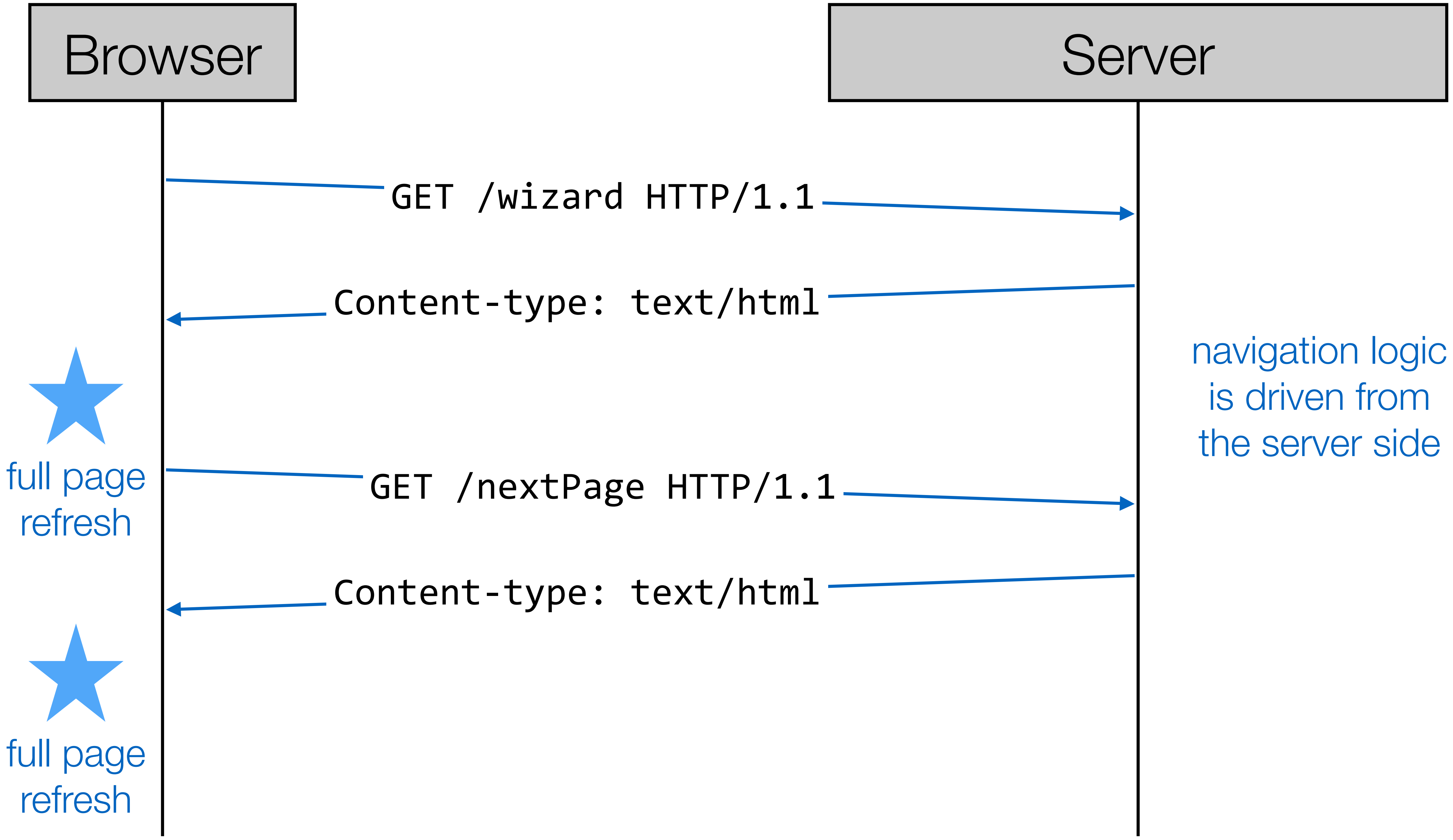
Client and presentation tiers


HTML
human-readable markup generation
Page navigation
REST API
XML
json
machine-readable markup generation
Shared business logic and rules
Data access
Message/ event bus
User
Client
Presentation
Business
Integration
Resources

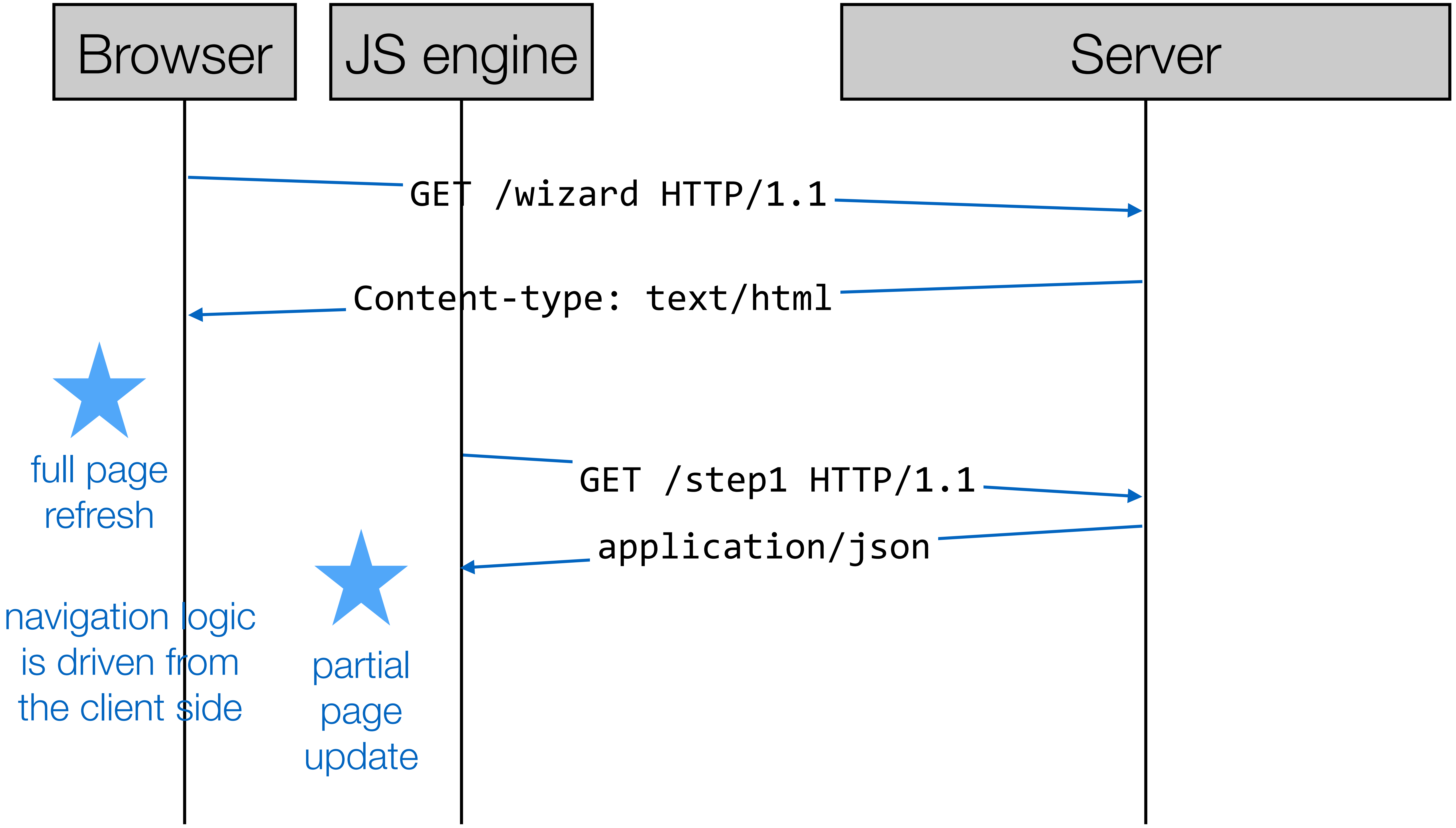
What is the relation between the **client** and the **presentation** tiers?

- The **client tier** is located on the **user device** (laptop, mobile phone, etc.).
- The **presentation tier** is located on the **server**. It generates content (HTML, JSON, PNG, etc.) that is used in the client tier (it also consumes content sent by the client tier).
- Components in the client and presentation tier use a **communication protocol**. Today, it is most often HTTP, but other protocols could be used.
- In many applications, the client tier consists of a **web browser**, a **JavaScript engine** and **scripts executed locally**.
- The client tier can also consist of a **native application** (it is often the case for mobile applications), making requests to the presentation tier. You could implement a native application in Java, with Swing.
- Some client applications **don't have a GUI** (command line tools, robots, etc.).

How do clients and server interact in the "traditional" MVC model?



How do clients and server interact in the "Single Page Application" model?

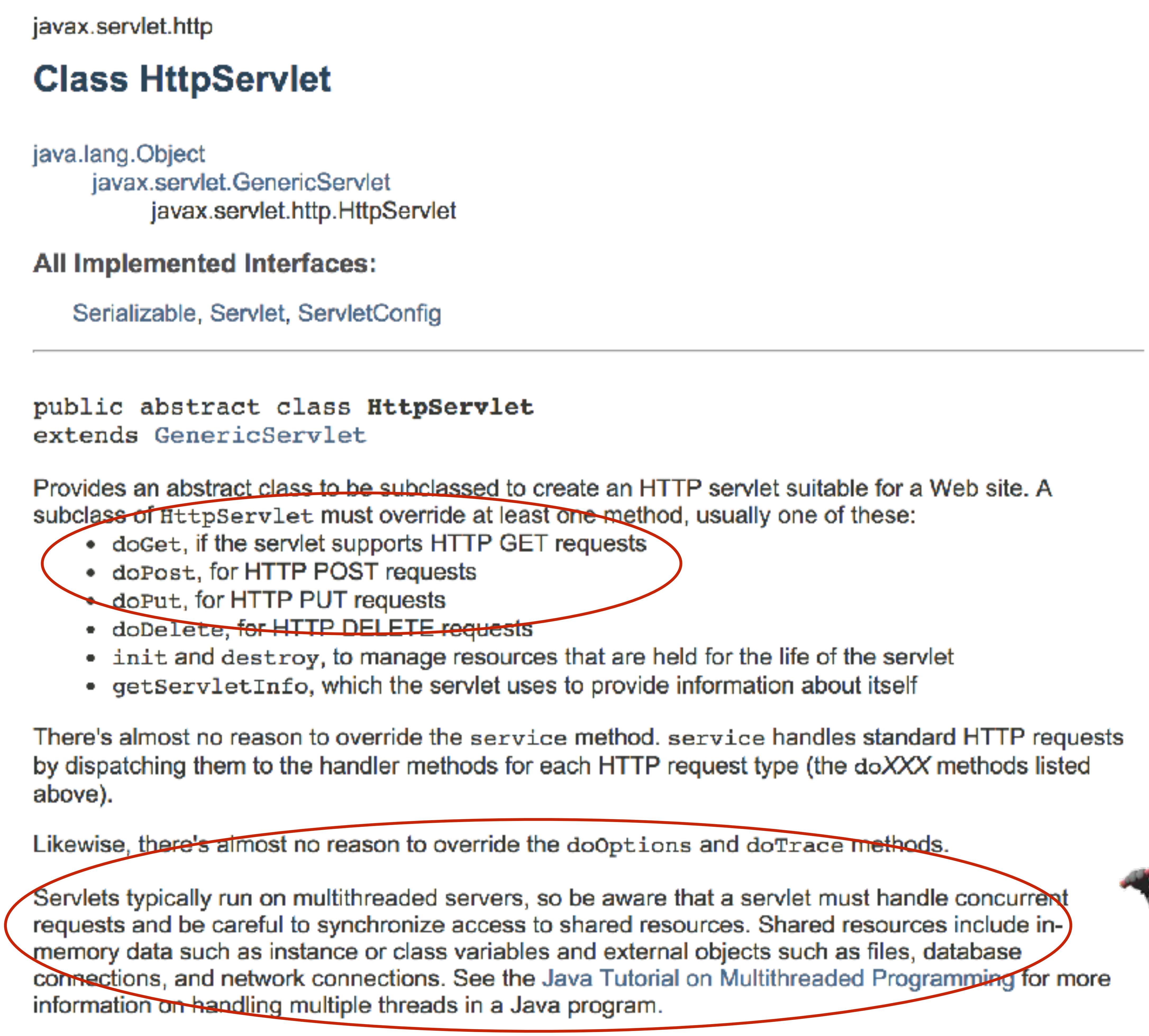


What is the ~~ugliest~~ simplest way to process an HTTP request in Java EE?

- 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.

```
                              GET /theApp/greeting HTTP/1.1
@WebServlet("/greeting")
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
```

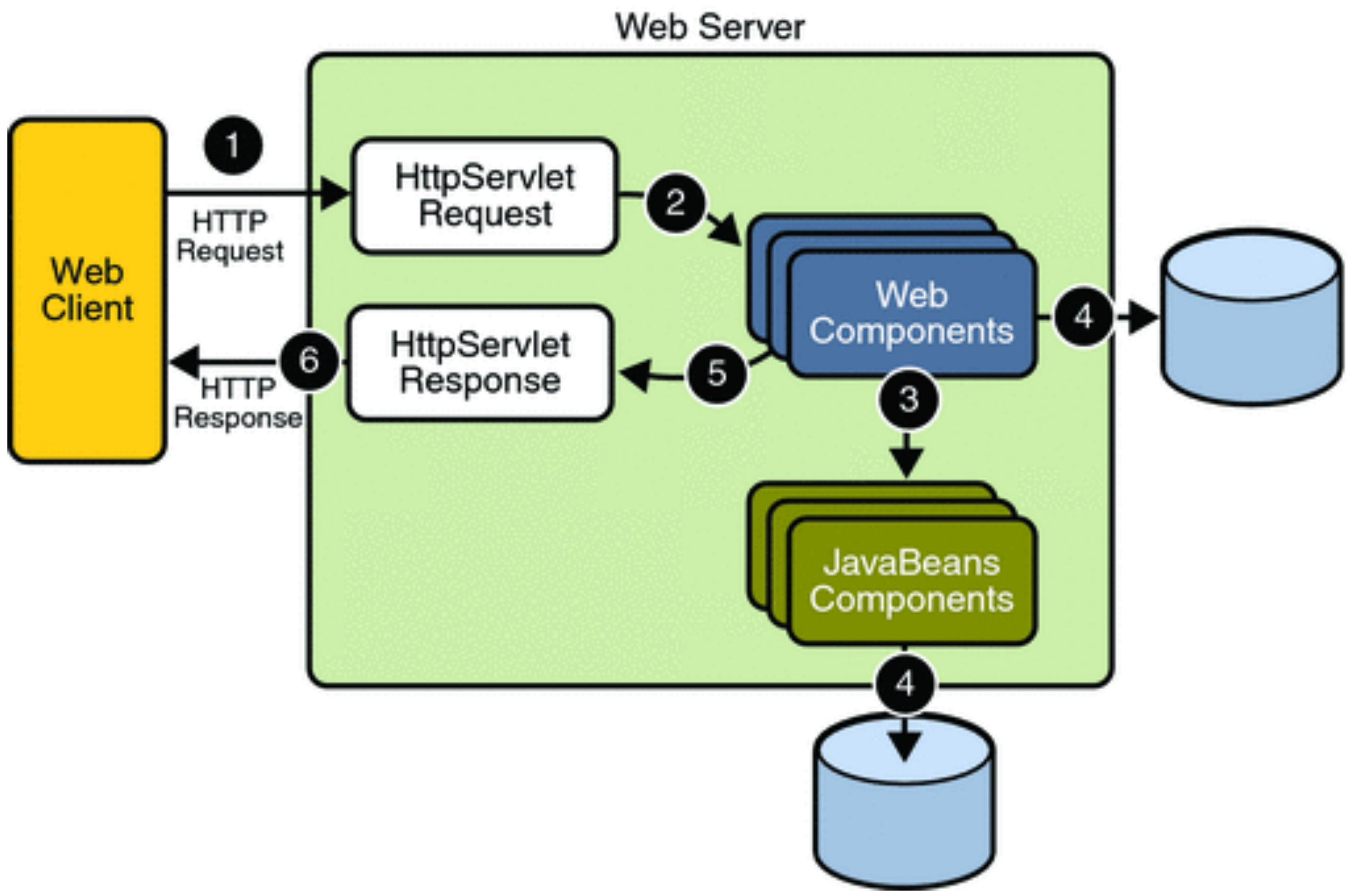
Provides an abstract class to be subclassed to create an HTTP servlet suitable for a Web site. A subclass of `HttpServlet` must override at least one method, usually one of these:

- `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.



Let's put it in practice…

Let’s put it in practice

HAUTE ÉCOLE
D’INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

- Set up the **local environment** (in addition to the docker-compose setup)
- Configure IntelliJ
- Open the simple mvc repo: https://github.com/SoftEng-HEIGVD/Teaching-HEIGVD-AMT-MVC-simple-example
- Let’s create a `DummyServlet` and quickly hack it together.

What’s wrong with this approach?

- 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

- 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 the view to render the model.

What does MVC mean in Web Apps?

- 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

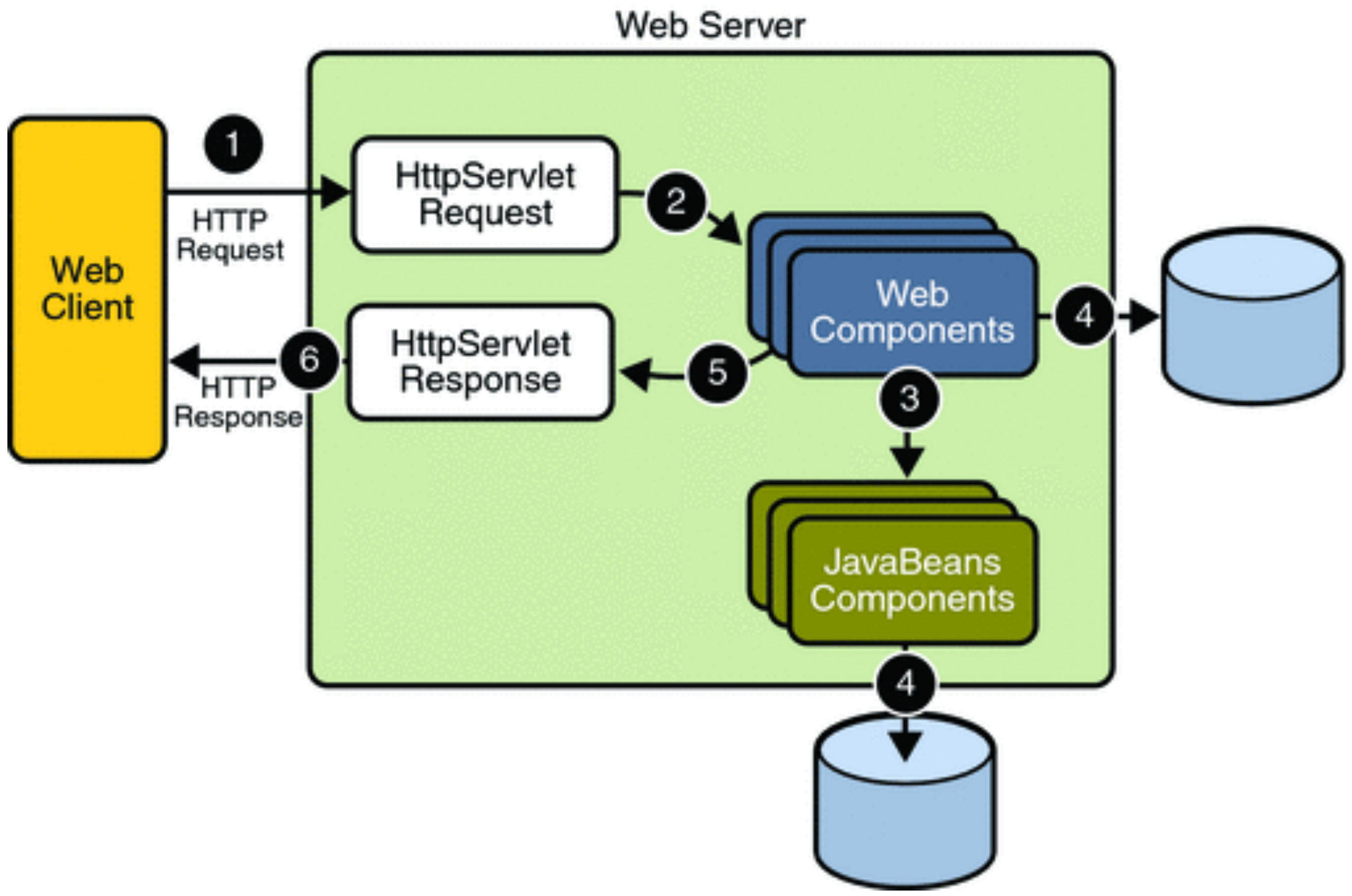
*I am a **JSP** page*

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*



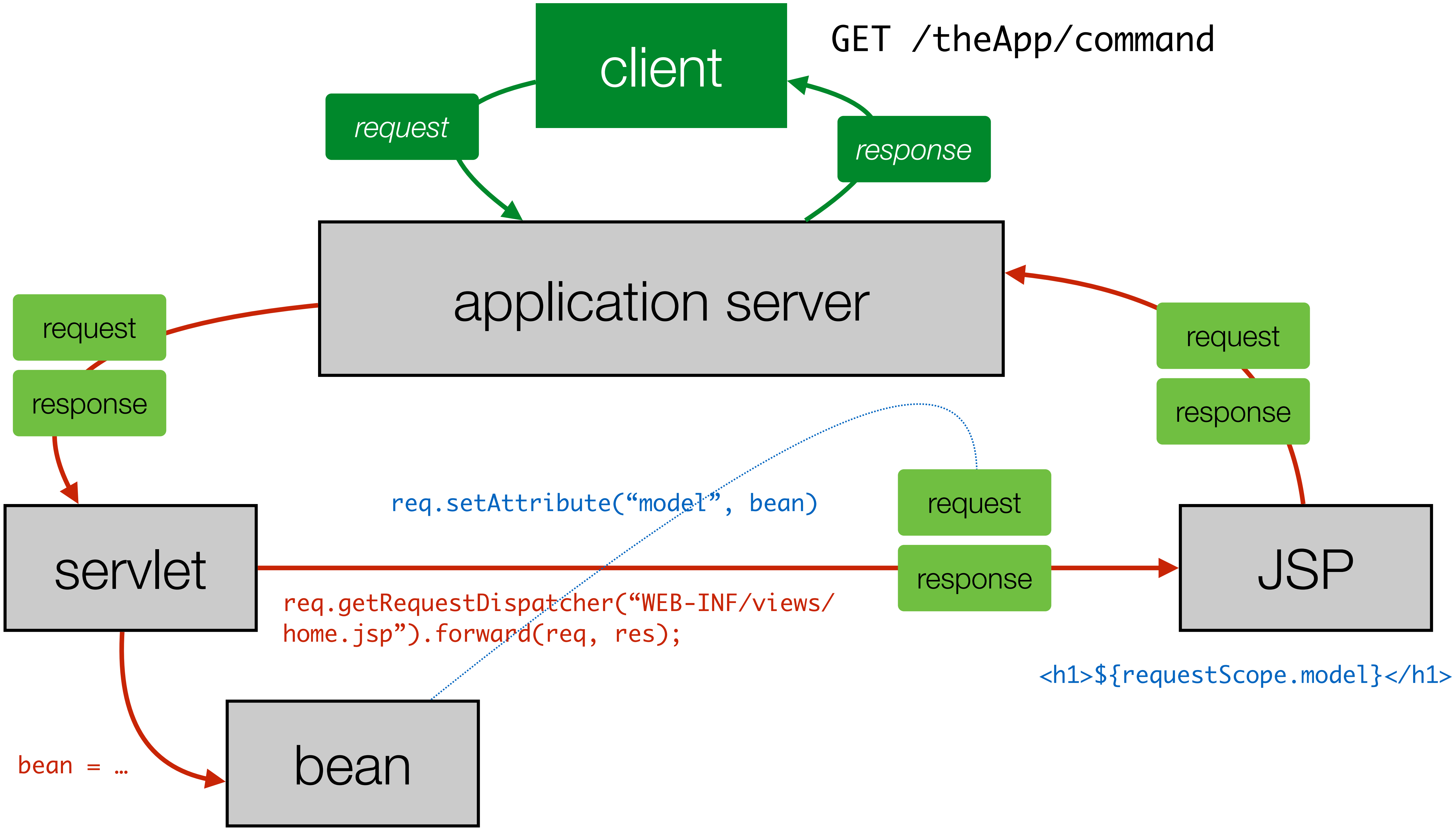
Let's put it in practice…

Let’s put it in practice

HAUTE ÉCOLE D’INGÉNIERIE ET DE GESTION DU CANTON DE VAUD
www.heig-vd.ch

- Let’s go back to the Simple MVC project and
    - Analyze the structure of packages
    - Study the model class
    - Study the service class
    - Study the controller class
    - Study the view template

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


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

What is the proper way to process an HTTP request in Java EE?

1 servlet mappé s/ 1 route

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);
}
```

Un servlet p/ opération
=> bonne pratique

JSP <<view>>

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

<div class="page-header">
  <h1>${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. $\longrightarrow$ HORRIBLE !
- **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?

HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD
www.heig-vd.ch

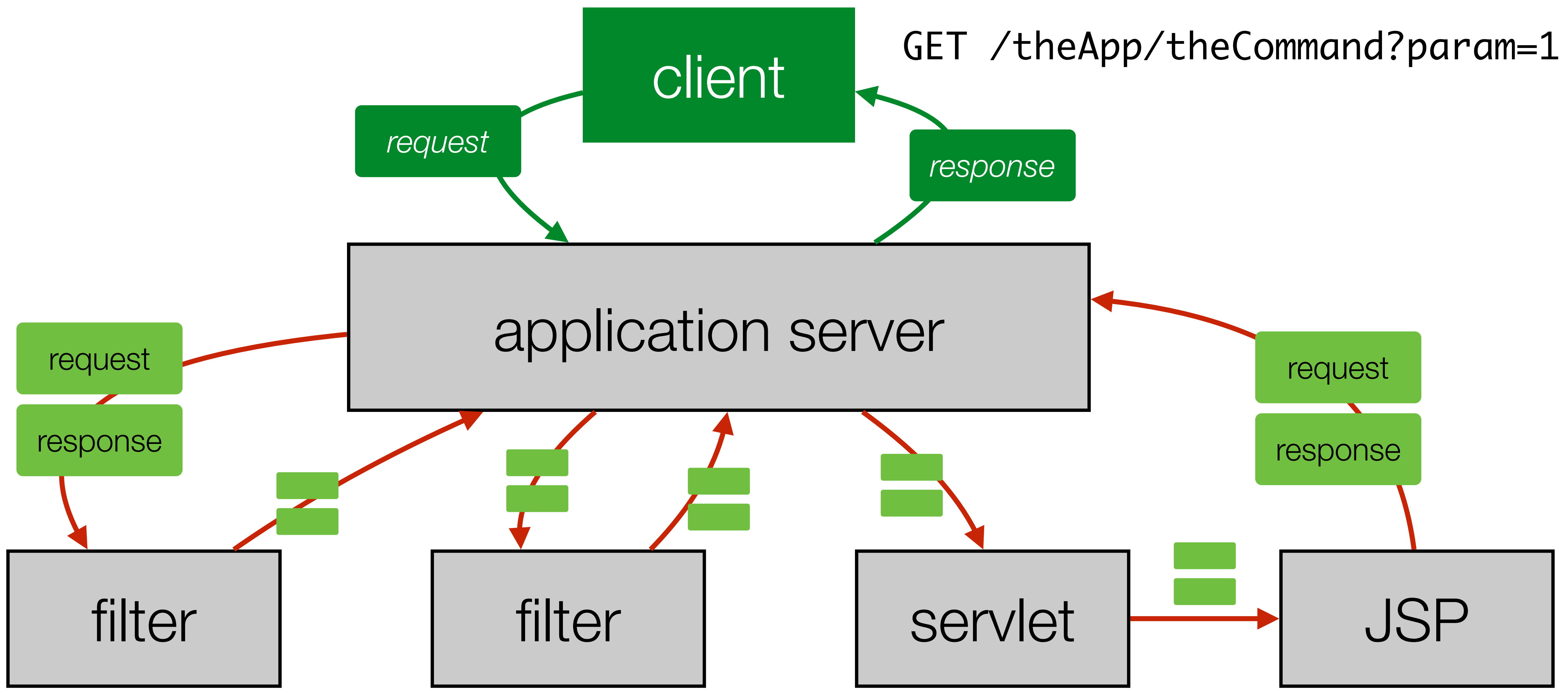
- HTTP requests sent by clients are received by the application server.
- Remember that several applications may be deployed in the application server. Each application has an associated **context**, which is a URL prefix.
- The application server **inspects the URL**. Based on its prefix, it is responsible to find the first application component that will initiate the processing of the request. Typically, this will be a servlet or a servlet filter.

Architecture pipes & filters

- 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?

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch



```
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

```
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.

Key takeaways

HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD
www.heig-vd.ch

- ✓ MVC pattern
- ● IoC pattern = Inversion of Control
- ✓ Pipes and Filters pattern

What did he mean by that?

IoC

HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD
www.heig-vd.ch

Inversion of Control → What ever of?
↓
the program flow

Question: who controls the program flow in the "normal" case?

IoC

HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD
www.heig-vd.ch

The developer controls the flow. In this code, ***he makes calls*** *to functions and methods.*

The developer can use a ***library****. He still decides when a third-party function should be invoked.*

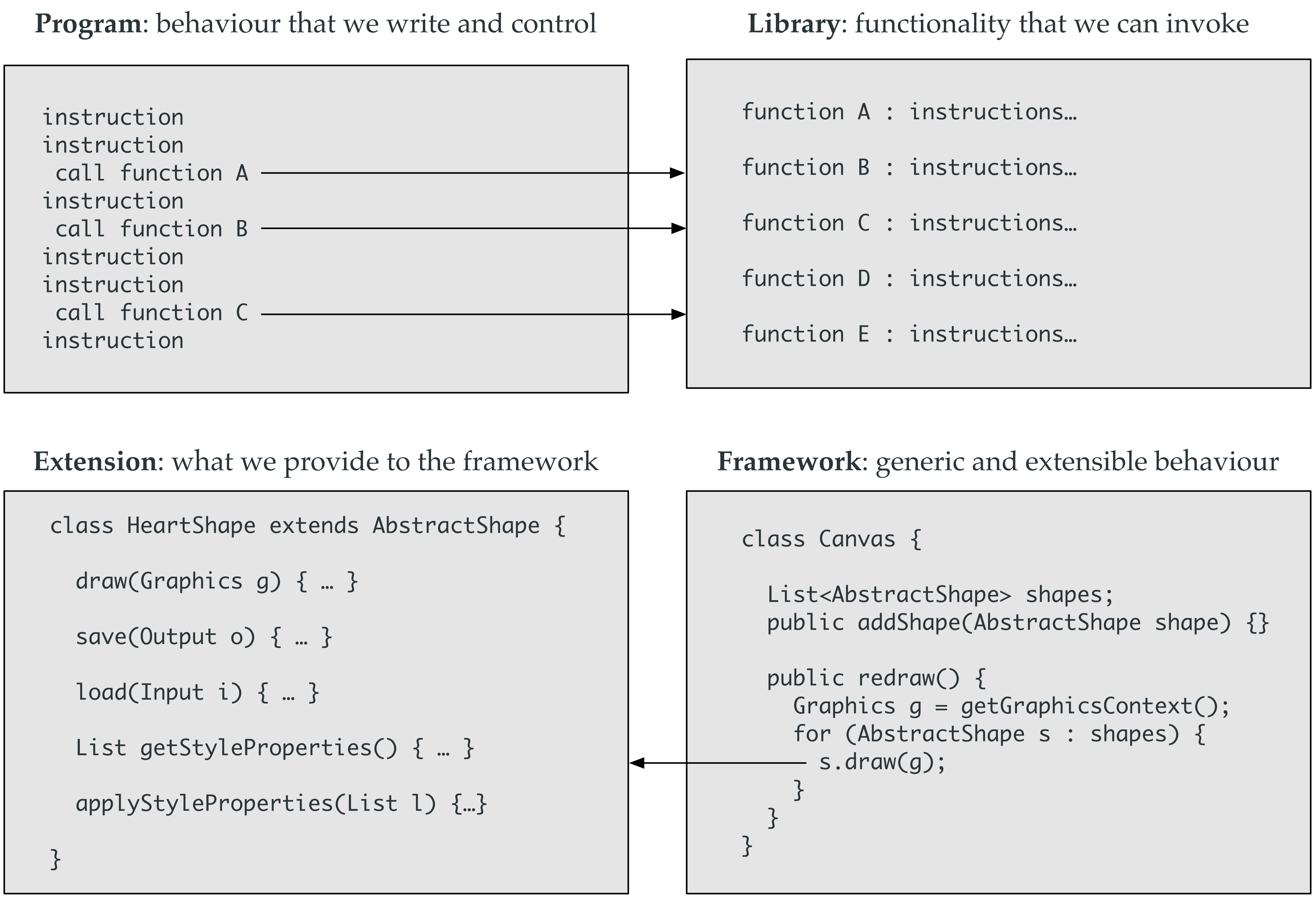
IoC

What does it mean to invert the control of the flow?

Coquille vide traitant des requêtes HTTP

It happens when the developer uses an **application framework**. *He provides extensions (typically sub-classes of abstract classes defined in the framework).* *The* ***framework calls*** *these extensions, when it makes sense.* ⚠

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch



Program: behaviour that we write and control

```
instruction
instruction
 call function A
instruction
 call function B
instruction
instruction
 call function C
instruction
```

Library: functionality that we can invoke

```
function A : instructions…

function B : instructions…

function C : instructions…

function D : instructions…

function E : instructions…
```

Extension: what we provide to the framework

```
class HeartShape extends AbstractShape {

  draw(Graphics g) { … }

  save(Output o) { … }

  load(Input i) { … }

  List getStyleProperties() { … }

  applyStyleProperties(List l) {…}

}
```

Framework: generic and extensible behaviour

```
class Canvas {

  List<AbstractShape> shapes;
  public addShape(AbstractShape shape) {}

  public redraw() {
    Graphics g = getGraphicsContext();
    for (AbstractShape s : shapes) {
      s.draw(g);
    }
  }
}
```

IoC in Java EE

HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD
www.heig-vd.ch

How is IoC applied in the presentation tier, when using servlets and JSP?

In the example, we never created instances of our servlets. We never called doGet or doPost ourselves. The Java EE app server did (it is the framework).

When is the "right time" to call our methods? This is why we mapped our servlets to URL patterns!

Questions?

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch