Fachhochschule Nordwestschweiz
Hochschule für Technik

Software Architecture

Exercise – Servlet Engine

BSc

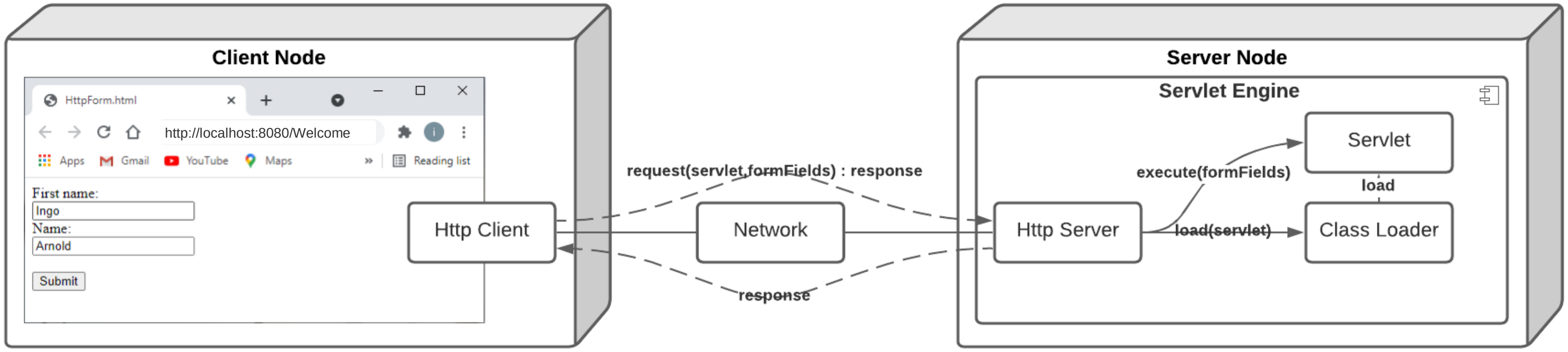
Ingo Arnold

Exercise Opening
Overall Motivation

The goal of this exercise is to create a **simplified version of a dynamic web application including the operational building blocks typical for it**.

The task is of comparatively more technical nature than previous exercises, which were more in the area of business domains.

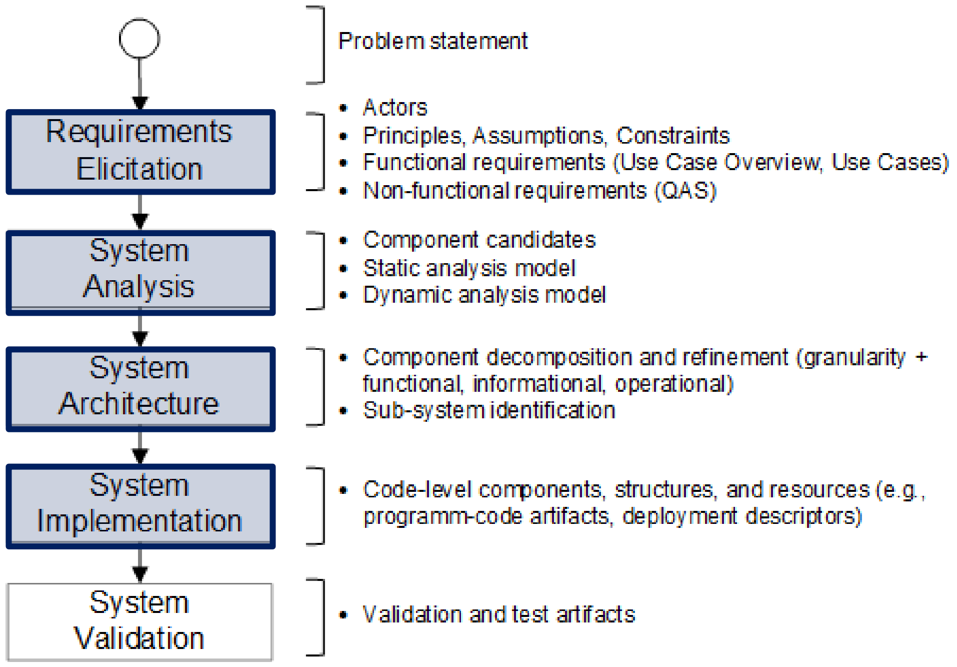
I have expressed the scenario you are to develop in a high-level sketch.



Exercise Opening

Motivation

You **create an initial design** for a **servlet engine** by incrementally following the process outlined, below. Note that your design should focus less on the algorithmic and more on the structural solution aspects.



- Actors
- Principles, Assumptions, Constraints
- Functional requirements (Use Case Overview, Use Cases)
- Non-functional requirements (QAS)

- Component candidates
- Static analysis model
- Dynamic analysis model

- Component decomposition and refinement (granularity + functional, informational, operational)
- Sub-system identification

- Code-level components, structures, and resources (e.g., programm-code artifacts, deployment descriptors)

- Validation and test artifacts

Exercise Opening
Motivation

A servlet engine is to be developed—i.e. a platform which receives http requests from clients and hosts java servlets to execute these requests.

- Your goal is to develop a servlet engine and demonstrate it based on a given scenario.
- A scenario that describes the entire process was already developed and provided to you as input.
- The scenario is divided into two parts because of the required client-server architecture (i.e., there is a client-side and a server-side part of the desired solution).
- Your task is to derive an activity diagram from the scenario, first.
- Based on the activity diagram, you elaborate an initial design in the form of a class diagram. Do not worry, if your classes sound very technical – the scenario is very technical, too.
- Finally, you implement this scenario in Java.
- A rudimentary implementation will suffice, but you should implement the client-server architecture based on a socket connection between client and server.

Exercise Agenda

- Requirements Elicitation
- System Analysis
- System Architecture
- System Implementation

Requirements Elicitation
Scenario

Requirements elicitation was already conducted by the team so that you receive the below scenario as an input for further elaboration.

1. A user on the client side enters data into the fields of an Http-form hosted by an Http client.
2. After completing the input, the user submits the form for further processing. In the process, the data required for sending is available in the form itself (i.e., protocol, server address, port, the resource that is to process the request (i.e., servlet name), and the form fields including the submitted values).
3. The Http client collects the data from the form and creates an Http-post-request. Http-post-requests are submitted via networks in text form.
4. The Http client establishes a connection to the Http server based on server address and port information and sends the Http-post-request into the established connection.

   Nota bene: it is assumed that the Http server is already started and waiting for requests from Http clients.
5. The Http server receives the Http-post request and extracts the contained data (i.e., resource name, form fields and values).
6. The Http server loads the requested resource (i.e., servlet-class).
7. The Http server establishes an Http-request object representing the sent request. After this object is created, the form field data is stored in the form of <key, value> pairs in the Http-request object.

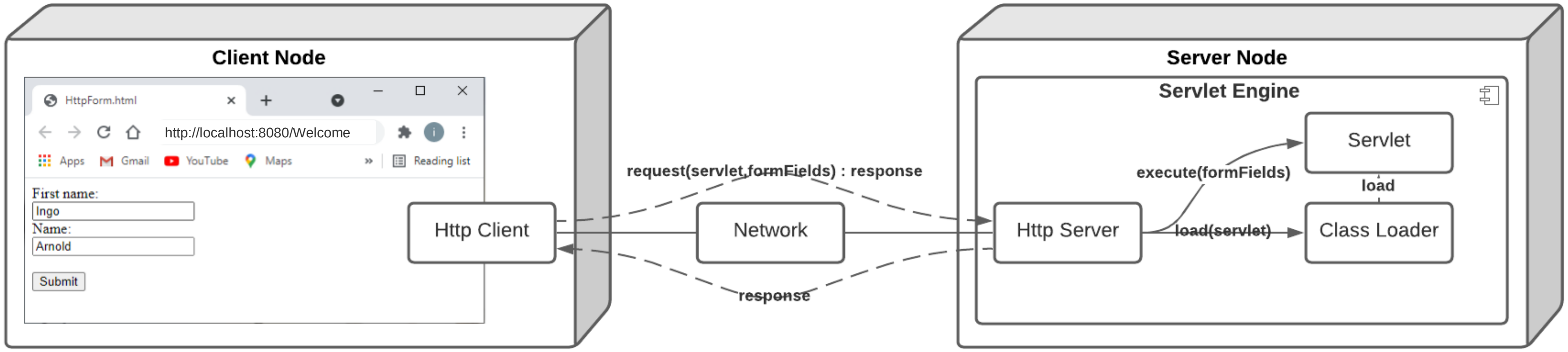
Requirements Elicitation
Scenario

Requirements elicitation was already conducted by the team so that you receive the below scenario as an input for further elaboration.

8. The Http server establishes an Http-response object. Into this, a servlet can write data that is sent back to a client as a response.
9. The Http server instantiates a servlet of the loaded servlet class and calls its service method by passing the established objects (i.e., Http-Request and Http-Response) to it.
10. The servlet uses the Http-request object to access the form fields, it calculates an output according to its business logic and writes it to the Http-response object.
11. The Http server converts the Http-response object into text to be able to send the text via the socked connection.
12. The Http server sends this back to the Http client via the established socket connection. After sending off the response, the Http server is again "ready for service".
13. The Http client receives the Http-post response, reads the sent data and displays it to the user on the client side.

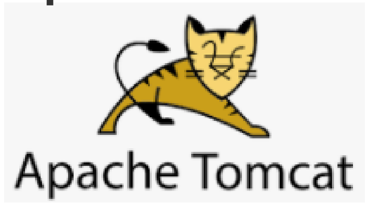
Exercise Opening

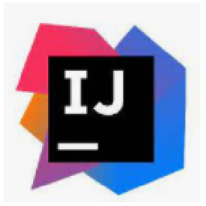
Servlet Demo



Postman

Chrome


Apache Tomcat


IntelliJ

Exercise Opening

Servlet Demo

Postman **Chrome**

Booting
- Postman: chrome://apps/
- Chrome: run browser

Request: http://localhost:8080/Welcome?firstName=Ingo&lastName=Arnold

Apache Tomcat

Booting
- XAMP control panel

WebApp directory: C:\xampp\tomcat\webapps\ROOT\WEB-INF
Server port: 8080

IntelliJ

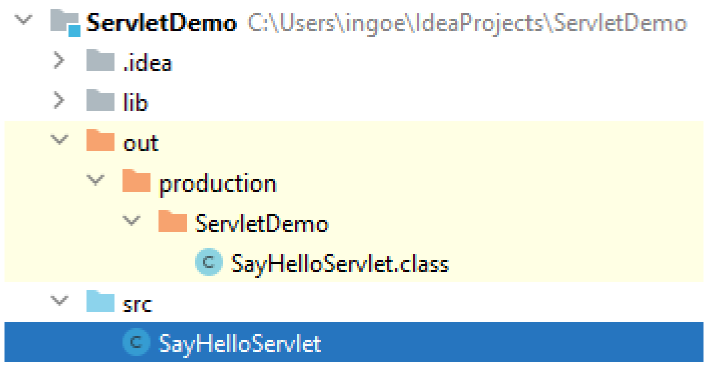
Booting
- Run IntelliJ

Servlet class deployment target: .\WEB-INF\classes
Deployment descriptor: .\WEB-INF\web.xml

Exercise Opening
Servlet Demo

IntelliJ

Servlet source:



```
ServletDemo C:\Users\ingoe\IdeaProjects\ServletDemo
  .idea
  lib
  out
    production
      ServletDemo
        SayHelloServlet.class
  src
    SayHelloServlet
```

```
public class SayHelloServlet extends HttpServlet {
   @Override
   protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
      String firstName = req.getParameter("firstName");
      String lastName = req.getParameter("lastName");
      ServletOutputStream out = resp.getOutputStream();
      out.println("<H1>Welcome</H1>");
      out.println("<span>Hello dear " + firstName + " " + lastName + "<span>");
   }
}
```

Exercise Opening

Servlet Demo

IntelliJ

Web.xml

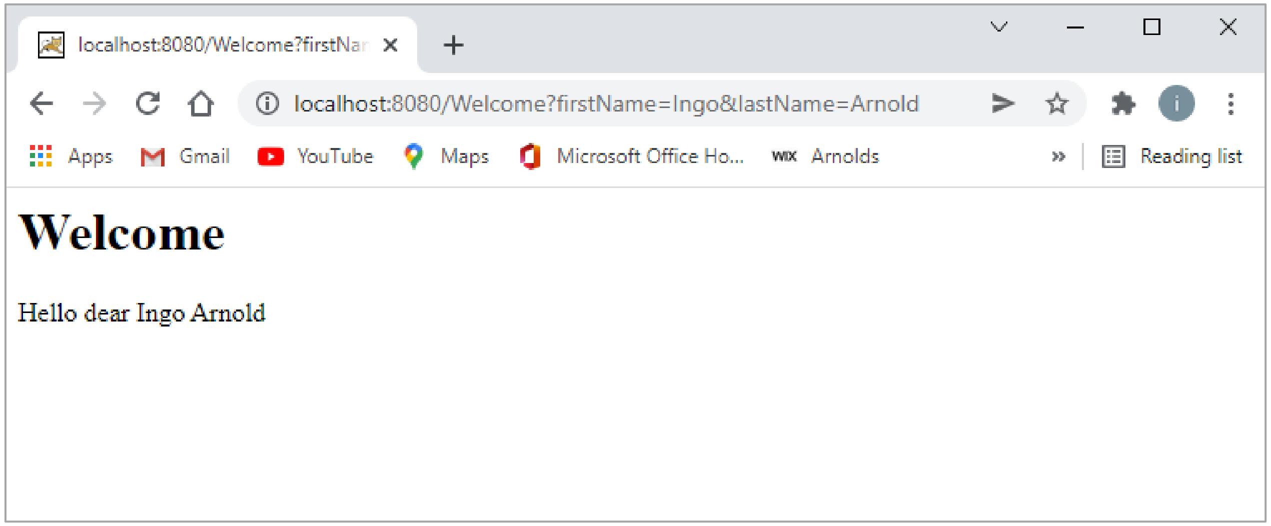
```
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
        http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
    version="3.1"
    metadata-complete="true">

  <display-name>Welcome to Tomcat</display-name>
  <description>
    Welcome to Tomcat
  </description>

  <servlet>
    <servlet-name>Welcome Servlet</servlet-name>
    <servlet-class>SayHelloServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>Welcome Servlet</servlet-name>
    <url-pattern>/Welcome</url-pattern>
  </servlet-mapping>

</web-app>
```

Exercise Opening

Servlet Demo

Chrome

Client view


localhost:8080/Welcome?firstNar
localhost:8080/Welcome?firstName=Ingo&lastName=Arnold
Apps
Gmail
YouTube
Maps
Microsoft Office Ho...
Arnolds
Reading list

Welcome

Hello dear Ingo Arnold

Exercise Agenda

- Requirements Elicitation
- System Analysis
- System Architecture
- System Implementation

System Analysis
Object Interaction

Create an activity diagram based on the provided scenario.

Exercise Agenda

- Requirements Elicitation
- System Analysis
- System Architecture
- System Implementation

System Architecture
Class Diagram

Create an initial class diagram from the analysis model (i.e., activity diagram).

Lecture Agenda

- Requirements Elicitation
- System Analysis
- System Architecture
- **System Implementation**

System Implementation
Initial Implementation in Java

Build on the initial architecture design and implement an initial version of an servlet engine.

Questions