

Projet Informatique

2019

Steve Hostettler

Software Modeling and Verification Group

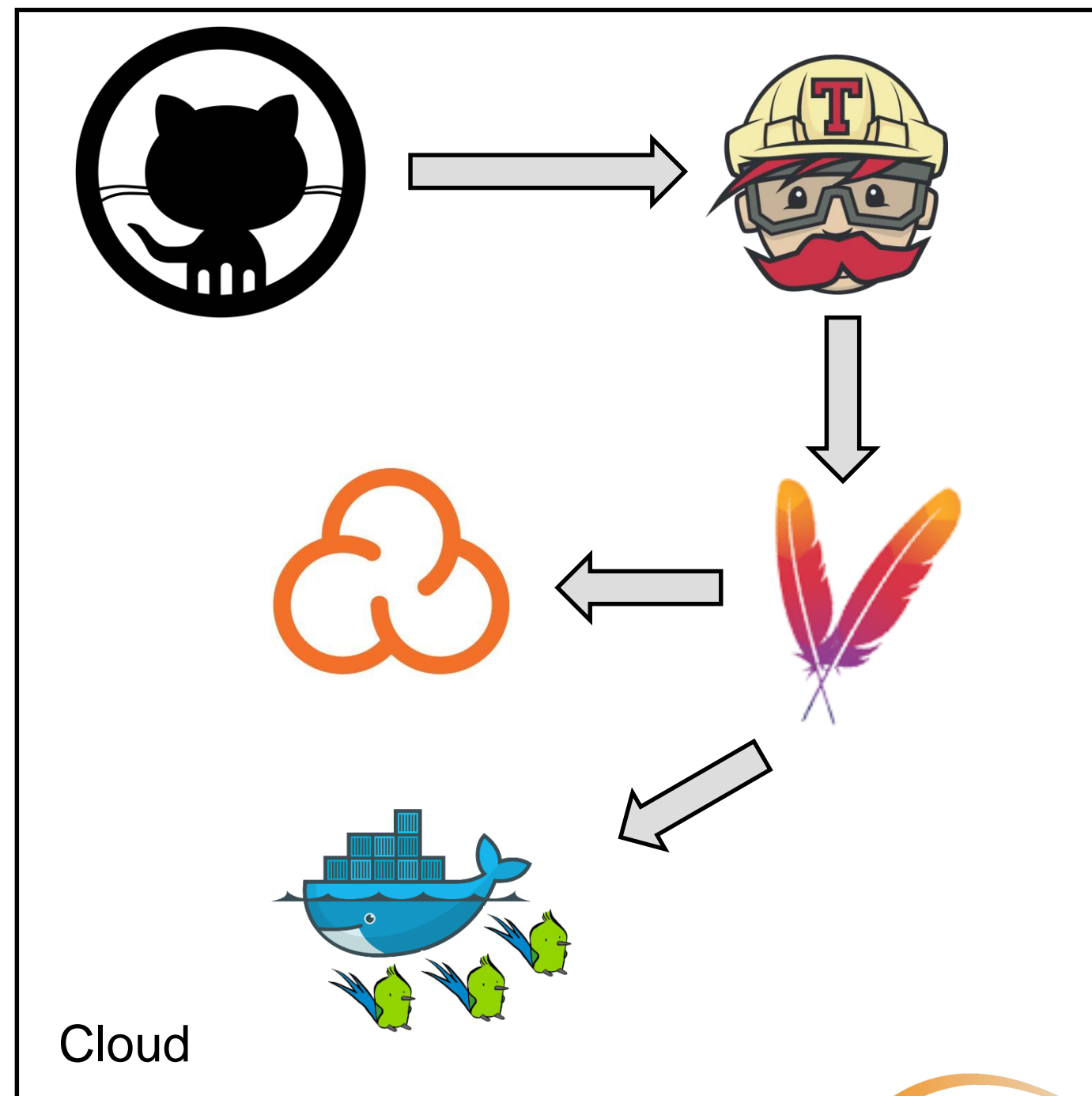
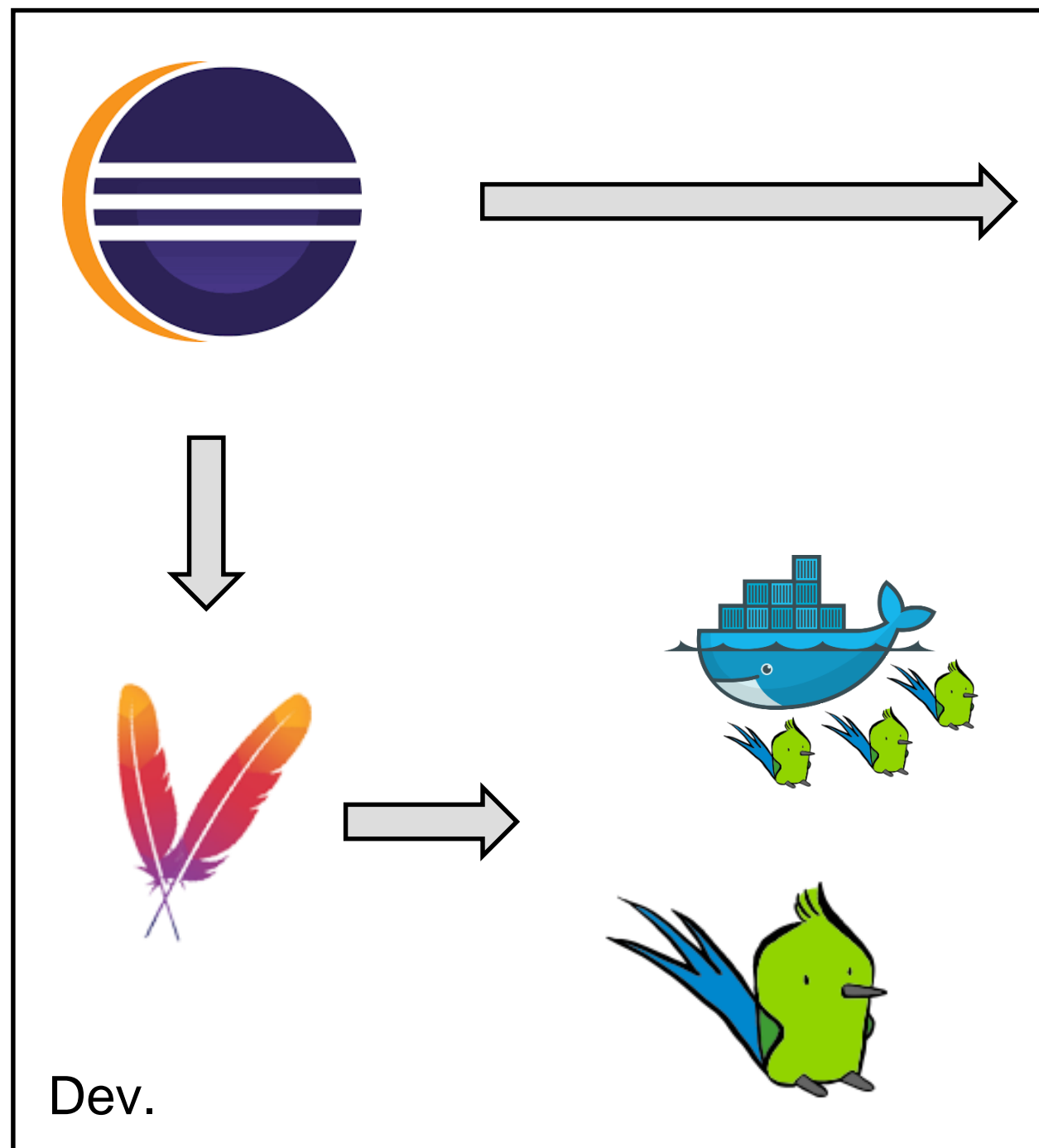
University of Geneva



Infrastructure de développement



L'usine logicielle



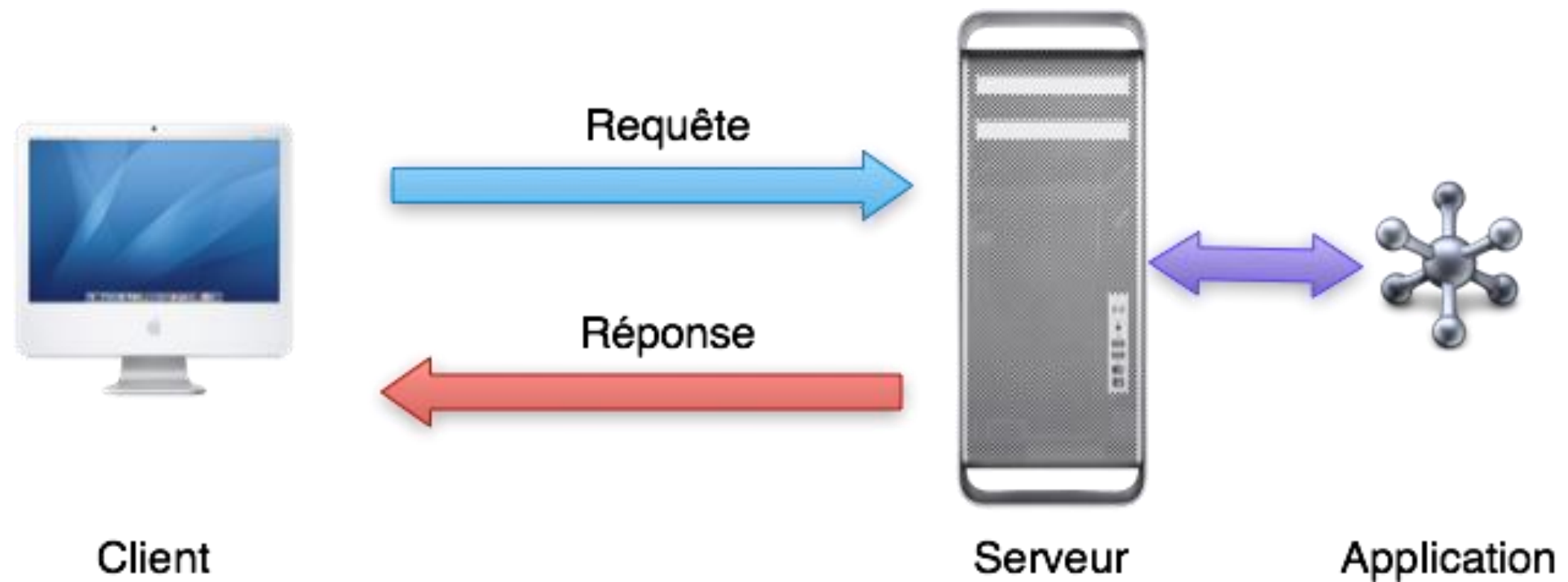
Applications WEB



Partie I : HTTP



Le protocole HTTP



Le protocole HTTP (commandes)

Commande	Description
GET adr	Requête de la ressource à l'adresse adr 255 caractères max. Les paramètres sont séparés par des &
POST data to adr	Envoi des données data à l'adresse adr
...	...



Le protocole HTTP (codes d'erreur)

Code	Description	Exemples
10x	Message d'information	
20x	Message de succès	200
30x	Redirection	301
40x	Erreur due au client	403'404'410
50x	Erreur due au serveur	500



Exercice 1 : Utilisez les commandes GET et PUT sur le serveur de l'université.

Partie II : Servir de l'information



Servir de l'information

Les navigateurs ne comprennent (presque) que HTML, javascript et CSS

Ces technologies sont adaptées pour afficher de l'information statique pas pour les interactions.

⇒ Comme toutes interactions reposent sur ces technologies, il a fallu bidouiller pour s'en sortir avec ça.



Servir de l'information

La plupart des approches reposent soit sur

- une génération de code HTML/javascript/CSS du côté du serveur en suivant le protocole HTTP. (JSPs-Servlets, CGI, PHP, ASP.net)
- Une generation de données business (Json) via des services Rest et des pages statiques (HTML/JavaScript/CSS)

La seconde approche est plus “scalable” et fourni un meilleur “découplément” entre la vue et la logique.



Génération de contenu

Vue + Métier



Anatomie d'une Servlet

- C'est une classe Java qui étend "HttpServlet" ou implémente "Servlet"
- Deux méthodes principales : doGet et doPost



Anatomie d'une Servlet

```
public class HelloWorld extends HttpServlet {  
  
    public void doGet(HttpServletRequest request, HttpServletResponse response)  
        throws IOException, ServletException {  
  
        response.setContentType("text/html");  
        PrintWriter out = response.getWriter();  
  
        out.println("<html>");//production de code HTML  
        out.println("<head>");  
        out.println("<title>Bonjour</title>");  
        out.println("</head>");  
        out.println("<body bgcolor=\"white\">");//Le fond est blanc  
  
        out.println("<h1> HelloWorld </h1>");  
  
        out.println("</body>");  
        out.println("</html>");  
  
    }  
}
```



Défauts des servlets

- Mélange d'interface graphique, de logique de présentation et de logique métier (voir d'accès à la couche de données).
- Mélange de langages sous forme de chaînes de caractères
- Pas de vérification syntaxique sur le HTML et le javascript.



Java Server Pages

- Un langage spécifique à l'écriture de page web dynamique
- Transformées en servlet pendant l'exécution



Java Server Pages : Exemple

```
<html>
<head>
<title>Sample Application JSP Page</title>
</head>

<body bgcolor=white>

    <table border="0">
        <tr>
            <td>
                <h1>Ma première page</h1> Ceci est un exemple
            </td>
        </tr>
    </table>

    <%=new String("Hello!")%>

</body>
</html>
```

Java Server Pages

La jsp précédente produit la servlet suivante:

```
public final class hello_jsp extends org.apache.jasper.runtime.HttpJspBase
    implements org.apache.jasper.runtime.JspSourceDependent {

    private static final JspFactory _jspxFactory = JspFactory.getDefaultFactory();

    private static java.util.List<String> _jspx_dependants;

    private javax.el.ExpressionFactory _el_expressionfactory;
    private org.apache.tomcat.InstanceManager _jsp_instancemanager;

    public java.util.List<String> getDependants() {
        return _jspx_dependants;
    }
    .....
```



Java Server Pages : un premier formulaire

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0">
  <jsp:directive.page contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" session="false"/>
  <jsp:output doctype-root-element="html"
    doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"
    doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
    omit-xml-declaration="true" />
  <html xmlns="http://www.w3.org/1999/xhtml">
    <head>
      <title>Insert title here</title>
    </head>
    <body>
      <form method="get" action="my">
        <INPUT type="text" name="first" id="first" />
        <INPUT type="text" name="second" id="second" />
        <INPUT type="submit" name="compute" title="compute" value="compute" />
      </form>
    </body>
  </html>
</jsp:root>
```

Java Server Pages : un premier formulaire ... et sa servlet

```
/**
 * Servlet implementation class MyFirstServlet
 */
@WebServlet(name="MyFirstServlet", urlPatterns="/my")
public class MyFirstServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.getOutputStream().print(computeResult(request));
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.getOutputStream().print(computeResult(request));
    }
}
```



Conclusion

- + Les servlets sont multi-threadées et portables
 - Les JSPs et les servlets permettent de mélanger les parties graphiques et logiques.
- => Il faut séparer les différentes parties d'une application par rapport à leur objectif respectif



Service JAX-RS



Génération de contenu métier uniquement



Service JAX-RS

Découpler les données de la mise en forme.

Gérer les formats standards facilement (XML, JSON)

Construit pour le protocole HTTP et basé sur les verbes GET, POST, ...

Java Architecture for XML RESTful services.



Service JAX-RS

Lier un chemin à une resource avec `@Path`

Des paramètres peuvent être passés au service avec `@QueryParam`



Service JAX-RS (DTO)

Les DTO (Data Transfer Object) sont annotés pour savoir comment les sérialiser (marshaller)

Il est possible d'ignorer un champs: `@Transient`

Sérialisation sur mesure: `@Converter`



Service JAX-RS (exemple)

```
@Path("/studentService")
public class StudentServiceFacade implements Serializable {

    private static final long serialVersionUID = 1318211294294344900L;

    @EJB
    private StudentService studentService;

    @GET
    @Produces({ "application/xml", "application/json" })
    @Path("all")
    public Response getStudents() {
        List<Student> students = studentService.getAll();
        return Response.ok(new StudentsDto(students)).build();
    }

    @GET
    @Produces({ "application/xml", "application/json" })
    @Path("session")
    public Response myMethod (@QueryParam("max") @DefaultValue("50") int maxResult) {
        ...
    }
}
```

Service JAX-RS (exemple)

```
@XmlRootElement
public class StudentsDto {

    @XmlElement
    private List<Student> students;

    public StudentsDto(List<Student> pStudents) {
        this.students = pStudents;
    }

    public StudentsDto() {
    }

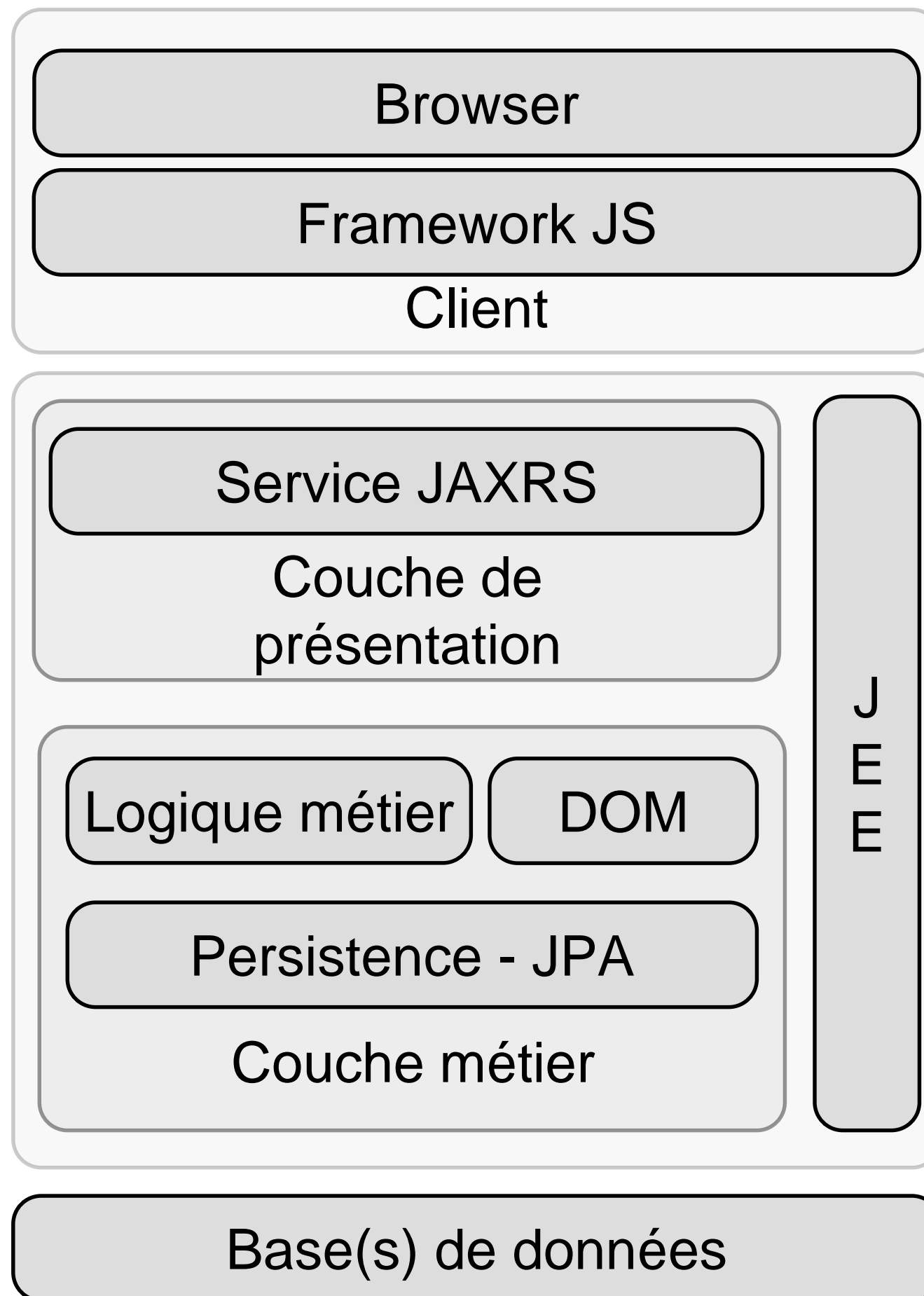
    public List<Student> getStudents() {
        return this.students;
    }
}
```

Service JAX-RS (exemple)

```
<studentsDto xmlns:ns2="http://ch.demo.app">
  <students>
    <id>0</id>
    <last_name>Doe</last_name>
    <firstName>John</firstName>
    <birthDate>1965-12-10T00:00:00+01:00</birthDate>
    <phoneNumber>
      <areaCode>0</areaCode>
      <countryCode>0</countryCode>
      <number>0</number>
    </phoneNumber>
    <address>
      <number>22</number>
      <street>wisteria street</street>
      <city>Downtown</city>
      <postalCode>34343</postalCode>
    </address>
  </students>
</studentsDto>
```

Architecture multi-tiers





Micro-Service vs Multi-Tiers

Les deux architectures ne sont pas mutuellement exclusives:

- Principalement une question de granularité, un micro se concentre un seul objectif
- Un micro-service peut avoir toutes les layers à l'exception des pages statiques.



- Analyse de l'application de démonstration

Bibliographie

<http://docs.oracle.com/javaee/8/tutorial/>

