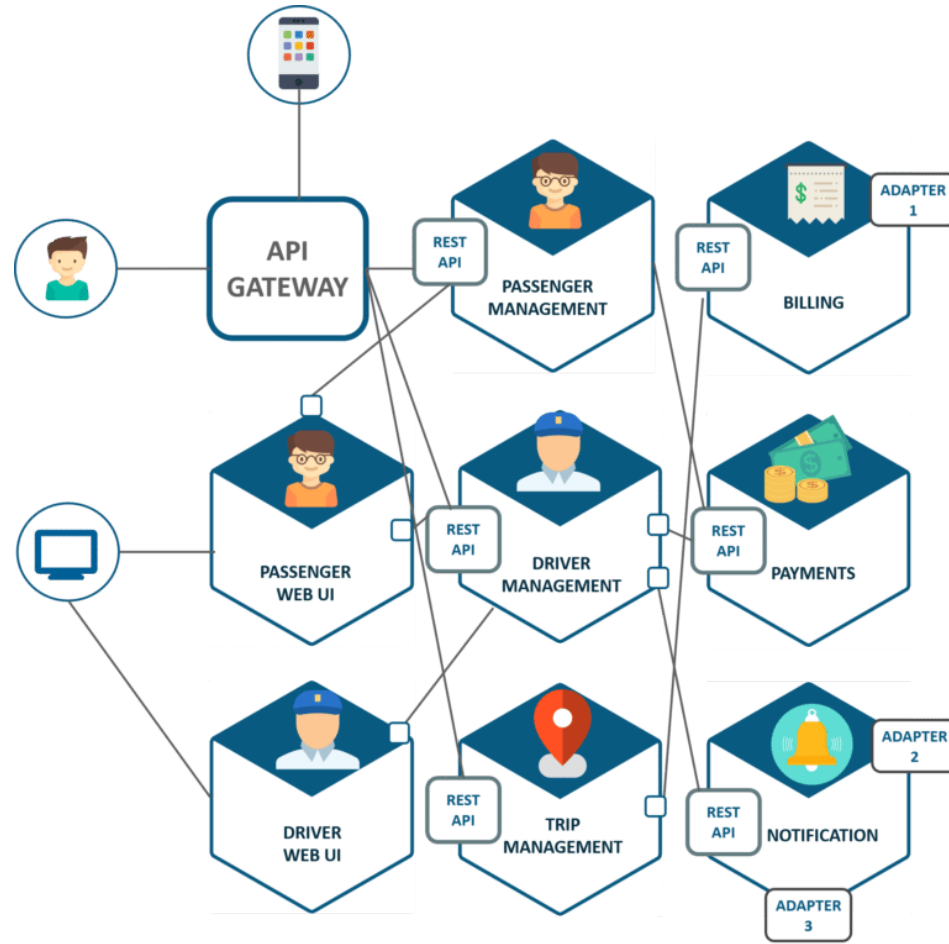
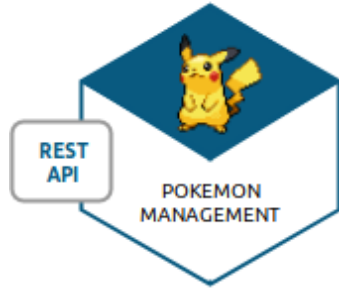


ALOM

GUI - MVC & TEMPLATING 

UBER





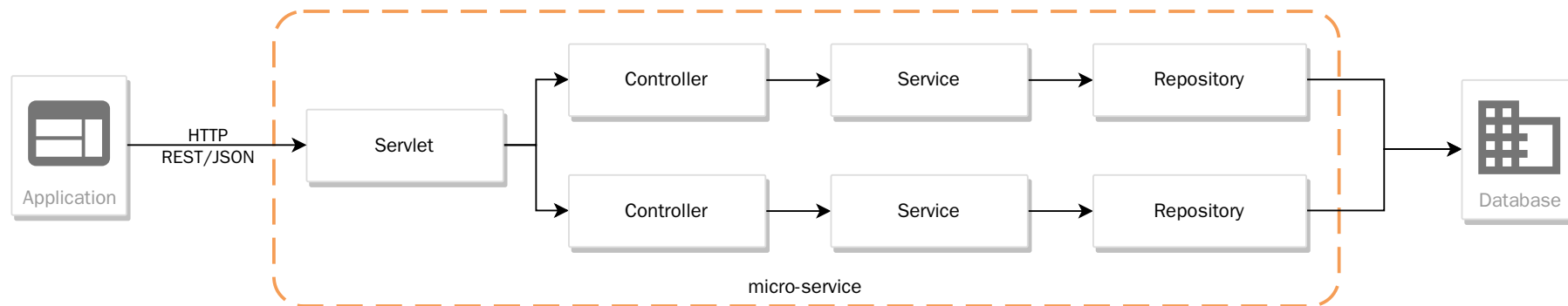
UN MICRO-SERVICE C'EST :

- Un ensemble de fonctionnalités du même domaine métier
- Un ou plusieurs canaux de communication
 - HTTP - REST/JSON
- Une source de données dédiée
- Un composant d'affichage



UN MICRO-SERVICE JAVA

On s'appuie sur les technologies connues: les servlets !



GUI

Au menu

- 😡 JSP
- 😬 JSTL
- 😊 Spring MVC
- 😄 Moteurs de templates



JSP

Java Server Pages

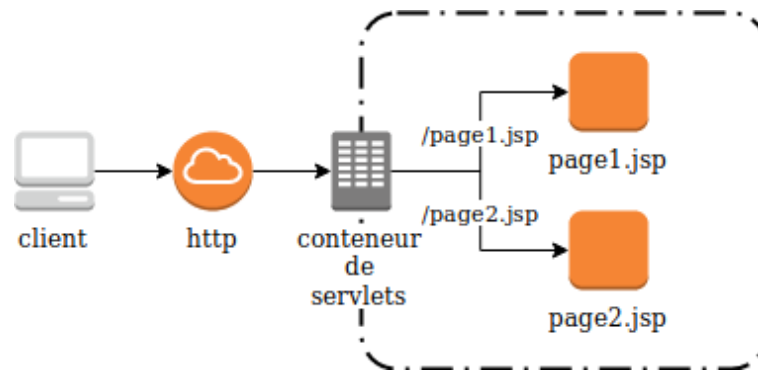


JSP

PRINCIPES

Page HTML embarquant du code Java

Compilée en servlet



JSP

DÉVELOPPEMENT



```
<?xml version="1.0" encoding="UTF-8" ?>
<%@ page language="java" contentType="text/html; charset=UTF-8" %>
<html>
  <body>
    Hello <%= request.getParameter("prenom") %>
      <%= request.getParameter("nom") %>
  </body>
</html>
```

carbon
carbon.now.sh

Relisez votre cours de CAR!



JSP

DÉPLOIEMENT

Packaging dans un fichier ".war" 

Déploiement dans un conteneur de servlets



OUCH.JSP



```
1 <%
2   for( Trainer trainer : trainers ) {
3   %>
4   <div class="trainer">
5
6     <span class="name"><%= trainer.getName() %></span>
7
8     <%
9       for( Pokemon pokemon : trainer.getTeam() ) {
10      %>
11      <div class="pokemon">
12        <p><%=pokemon.getName()%></p> <span class="label"><%=pokemon.getLevel()%></span>
13        
14      </div>
15    <%
16      }
17    %>
18
19  </div>
20 <%
21   }
22 %>
```

😨 JSTL

JSP Standard Tag Library



JSTL

Language d'expression **EL** (Expression Language)



```
<h1>Hello <%= session.getAttribute("trainer").getName() %> </h1>
```

```
<h1>Hello ${sessionScope.trainer.name}</h1>
```

carbon
carbon.now.sh

JSTL EL



```
${object.property}  
${param["pokemonId"]}  
${sessionScope["trainer"]}  
${pokemon.stats[0].value}
```

carbon
carbon.now.sh

JSTL TAGLIBS

Balises facilitant le développement JSP. Evite le code java et minimise les balises %

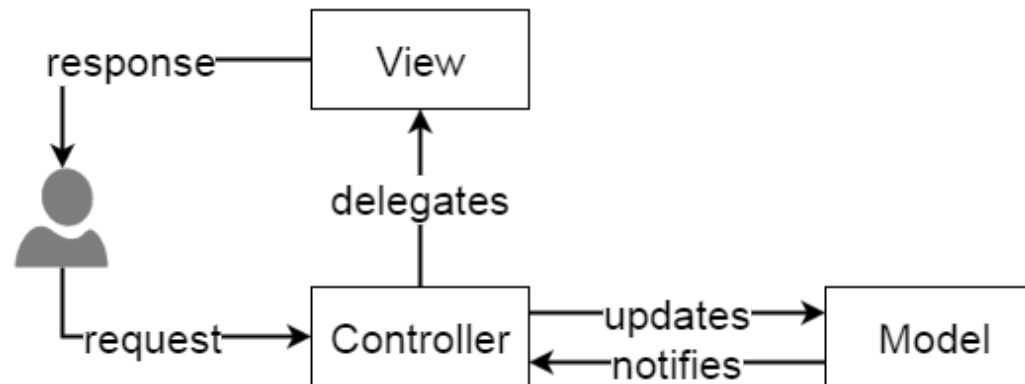


```
1 <%@ taglib uri="http://java.sun.com/jstl/core" prefix="c" %>
2
3 <jsp:include page="header.jsp" />
4 <table>
5     <c:forEach var="pokemon" select="${trainer.team}">
6         <tr>
7             <td><c:out value="${pokemon.name}" /></td>
8             <td><c:out value="${pokemon.id}" /></td>
9         </tr>
10    </c:forEach>
11
12 </table>
```

DESIGN PATTERN MVC

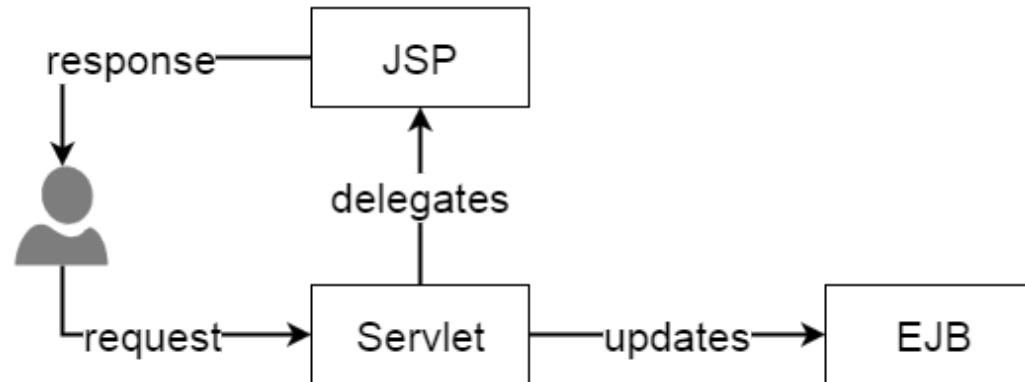
Sépare les responsabilités (👮 SOLID)

- **Model** : Contient les données
- **View** : Affiche les données
- **Controller** : Déclenche des actions sur les données et met à jour la vue



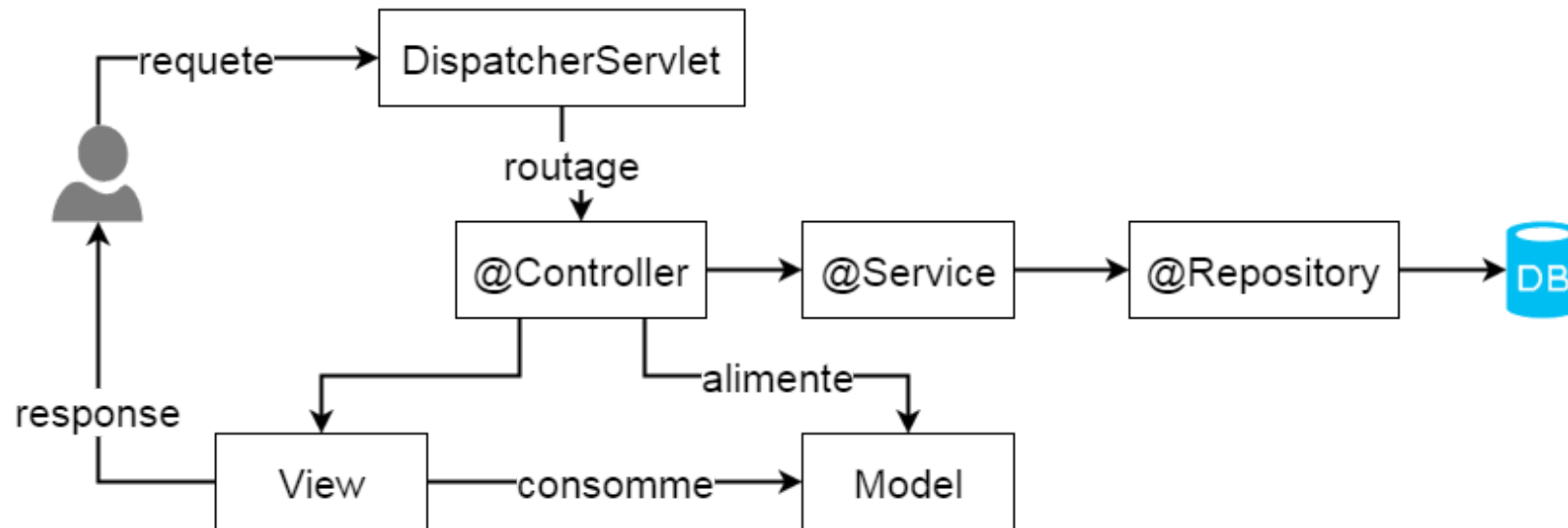
DESIGN PATTERN MVC

Déjà des pistes avec le couplage Servlet/JSP



MVC AVEC spring

by Pivotal™



MVC AVEC SPRING - MODEL

```
@Controller
class TrainerController {

    @Autowired
    private TrainerService trainerService;

    @GetMapping("/trainers")
    String showTrainersPage(Model model) {
        // on reçoit un model en injection et on l'alimente
        model.addAttribute("trainersList", trainerService.getA
        return "trainers/list"; // on retourne le nom de la vu
    }
}
```

MVC AVEC SPRING - MODELANDVIEW

```
@Controller
class TrainerController {

    @Autowired
    private TrainerService trainerService;

    @GetMapping("/trainers")
    ModelAndView showTrainersPage() {
        // on crée un ModelAndView avec le nom de la vue
        var modelAndView = new ModelAndView("trainers/list");
        // on alimente le modèle
        modelAndView.addObject("trainersList", trainerService.
        return modelAndView;
    }
}
```

MVC AVEC SPRING - VIEW

Dans

```
src/main/resources/templates/trainers/list
```

```
{{#trainersList}}  
<div class="card shadow-sm mb-4">  
  <div class="card-header">  
    <h4>{{name}}</h4>  
  </div>  
</div>  
{{/trainersList}}
```

MVC AVEC SPRING

- `Model` : Les données à envoyer à la vue
- `ModelAndView` : Les données + le nom de la vue
- `ViewResolver` : Utilisé par Spring pour charger le fichier de vue

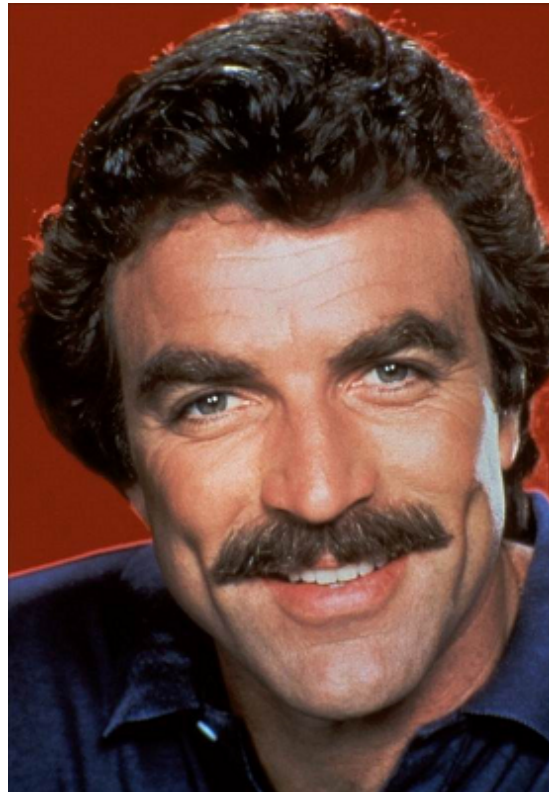
Les moteurs de templates dispo avec *Spring-Boot*:

- Thymeleaf
- Mustache
- FreeMarker
- Groovy

MOTEURS DE TEMPLATES

MUSTACHE (MAN-PAGE)

Logic-less templates



MUSTACHE

```
Hello {{name}},  
You have just won {{value}} dollars!  
{{#taxed}}  
Well, {{taxed_value}} dollars, after taxes.  
{{/taxed}}
```

```
{  
  "name": "Luke",  
  "value": 10000,  
  "taxed_value": 8000,  
  "taxed": true  
}
```

Result

```
Hello Luke,  
You have just won 10000 dollars!  
Well, 8000 dollars, after taxes.
```

TP



GUI - MVC & Templating