Programmation WEB
M1 MIAGE

Alice Jacquot, LRI
jacquot@lri.fr www.lri.fr/~jacquot

Présentation du cours

Mon parcours

- Thèse en informatique théorique
- Actuellement enseignante-chercheuse en bioinformatique
- Formation complémentaires en prog web (3 mois)
- Deux ans en développement web (back et front)

Organisation de l’UE (MIAGE classique)

- 7 semaines
- Séances de 3h de cours-TP, en présentiel*
- 100% contrôle continu :
  - 3 notes de TP (30%, 10% chaque)
  - 1 mini projet en binôme (30%)
  - 3 petits QCM en ligne sur eCampus (20%)
  - 1 contrôle (en classe/en ligne) (20%)

Organisation de l’UE (MIAGE alternance)

- 7 semaines
- Séances de 3h de cours-TP, en présentiel*
- 100% contrôle continu :
  - 3 ou 4 notes de TP (40-50%)
  - 2 petits QCM en ligne sur eCampus (20%)
  - 1 contrôle (en classe/en ligne) (30-40%)

Programmation web

But : conception d'application web

Vous connaissez déjà :

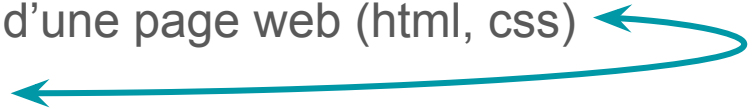
- conception sommaire d'une page web (html, css)
- applications locales
- JDBC/JPA

Programmation web

But : conception d'application web

Vous connaissez déjà :

- conception sommaire d'une page web (html, css)
- applications locales
- JDBC/JPA ?



Connexion entre les deux

+ pages dynamiques (Javascript)

… et pas mal d'outils et concepts intervenants

Technologies utilisées pour ce module

- Serveur (backend) en Java EE (IDE eclipse)
- Framework spring
- Requêtes http (outil insomnia, mode développeur de navigateur F12)
- format JSON, xml
- Front end : html, css, JS
- Organisation “MVC” : modèle-vue-contrôleur

Bien sûr, d’autres outils et technologies existent, les grands principes se transposent (PHP, .NET, JSP, typescript, nodeJS…)

Qu’est-ce qu’une application web ?

Une application web ?

Un **programme** informatique installée (déployée) “sur” un **serveur**, dédié être **appelé via** des communication **internet**, par navigateur, qui se plie à leurs contraintes (protocoles, capacité d’interprétation du navigateur)

Le **client** *utilise* un **navigateur*** qui gère l’affichage et l'exécution de scripts fournis par le serveur.

Ils interagissent via l’envoi de **requêtes** qui respectent des protocoles (*http*, *ftp*,...)

Une application web ?

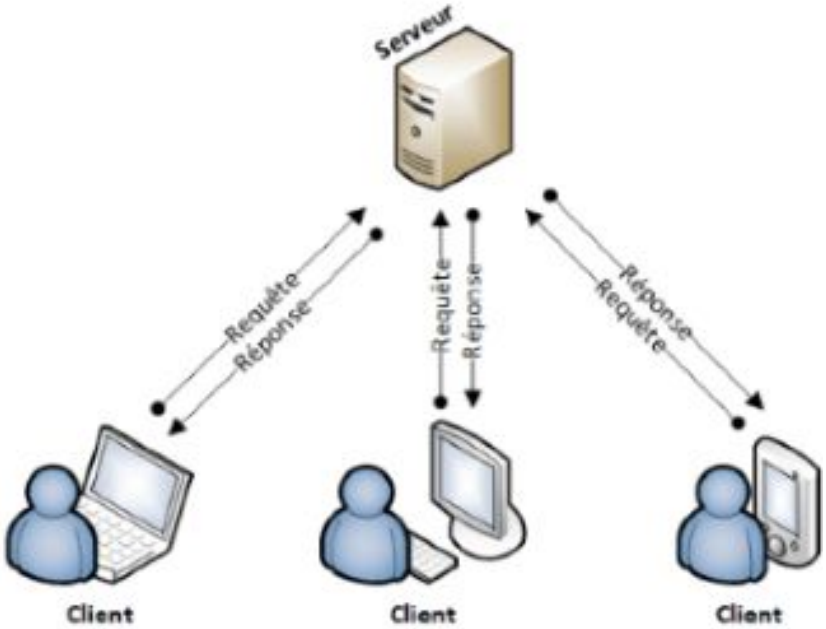
Un **programme** informatique installée (déployée) “sur” un **serveur**, dédié être **appelé via** des communication **internet**, par navigateur, qui se plie à leurs contraintes (protocoles, **capacité d’interprétation du navigateur**)

Le **client** *utilise* un **navigateur*** qui gère l’affichage et l'exécution de scripts fournis par le serveur.

Ils interagissent via l’envoi de **requêtes** qui respectent des protocoles (*http*, *ftp*,...)

*Des applications communiquent via internet avec des “**client lourd**” (application à installer), autrement dit **pas** par navigateur. S’agit-il d’application web ?
Peu importe, il faut savoir que ça existe, ce n’est pas le sujet du cours.

Client-Serveur



- “Une” machine serveur, hébergeant l’application serveur
- Plusieurs clients
- Communication par requêtes/réponse http (ou ftp, socket, etc)

Un serveur, c’est quoi ?

On utilise le même mot pour signifier plusieurs choses (légèrement) différentes :

- L’application mise à disposition aux clients, gérant la communication avec ceux-ci via l’écoute d’un port réseau et les traitements divers
- La/une/les machine hébergeant l’application serveur (n’importe quelle machine peut être un serveur)

Pour ce qui est de la **programmation** web, la distinction n’a que peu d’importance : une fois que la communication aux réseau et différents composants (Bases de données, ressources) est établie on ne se préoccupe “pas” du matériel

Client web

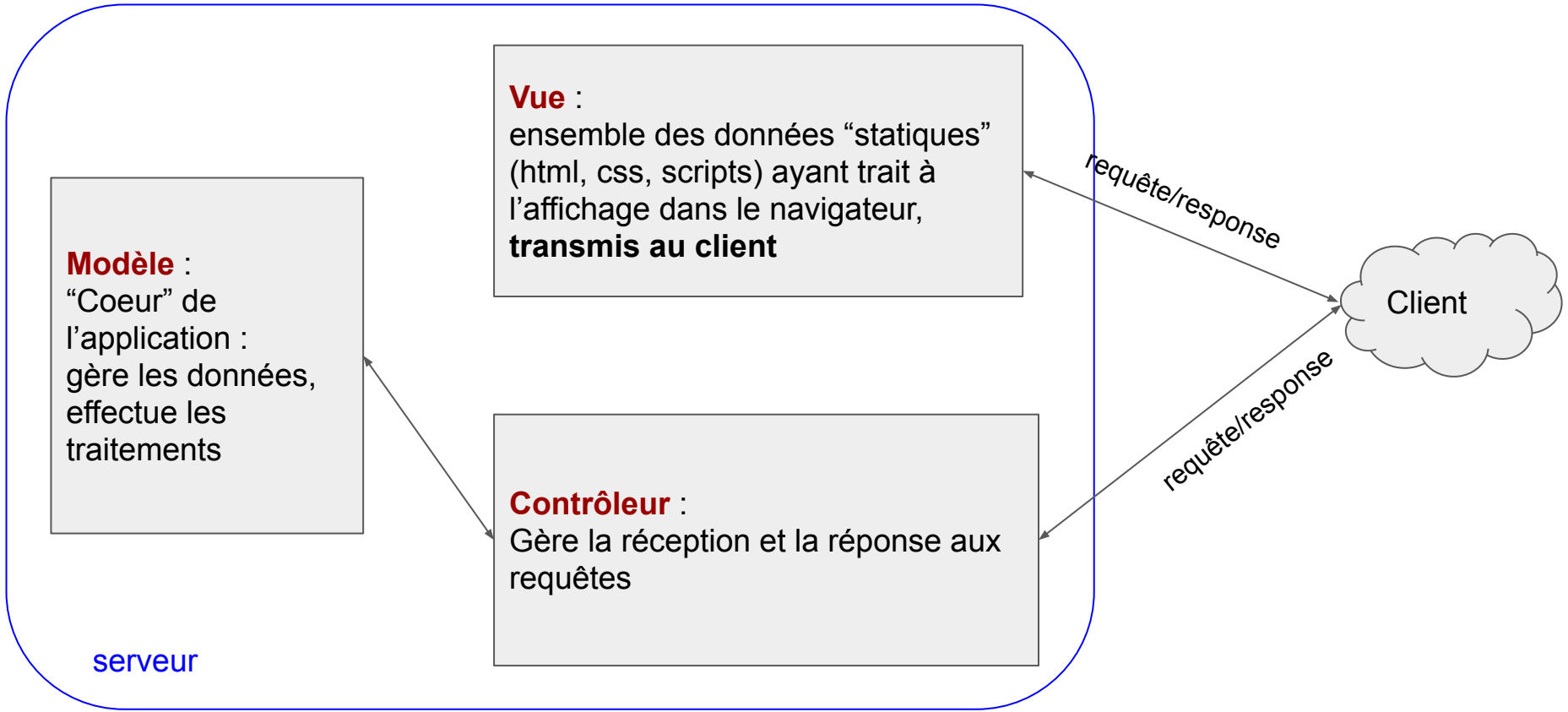
- **Possède un navigateur** : **émet des requêtes** via internet vers une adresse - soit directement par son IP, soit via le DNS, et écoute les réponses

- Le navigateur **interprète** (affiche, fait tourner les scripts) ce qu'il reçoit

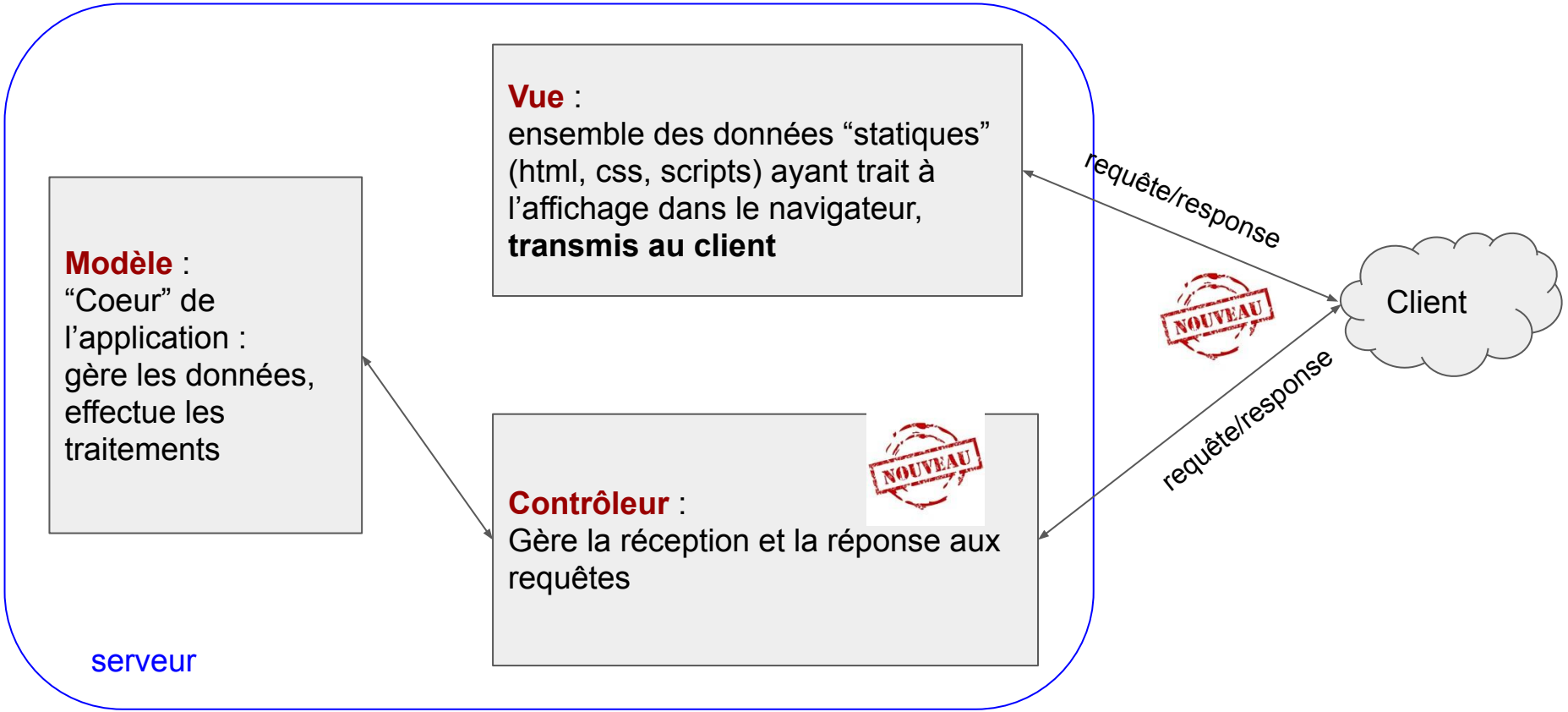
Une application web est :

- **déployée** sur un serveur, qui possède une **adresse** IP, et en général un alias dans le DNS (Domain Name System)
- à l'**écoute d'un port**, par défaut le port 80 : les messages transmis à la carte réseau pour ce port seront transmis à l'application
- en permanence, ou presque, en service

Modèle-vue-contrôleur



Modèle-vue-contrôleur



Programme de l’UE

- Cette semaine : 3 séance, backend, controleur (un TP à rendre)
- Un QCM back
- 17 novembre : une séance, backend + appels HTTP (un TP à rendre)
- Janvier : frontend (html, js, css) (un ou deux TP à rendre, un QCM)
- Contrôle en janvier

Réseau et protocole

Réseau et protocoles : mini cours

Cf extrait de cours de Kim Nguyen

Http : premiers pas

Protocole http

N	Nom	Description	Eq. OSI
4	*Application*	HTTP, Bitorrent, FTP, ...	5, 6, 7
3	*Transport*	TCP, UDP, SCTP, ...	4
2	*Internet*	IP (v4, v6), ICMP, IPsec, ...	3
1	*Link*	Ethernet, 802.11, ...	1, 2

Caractéristiques du protocole HTTP

- Sans connexion permanente:
  - Le client se connecte au serveur, envoie sa requête, se déconnecte
  - Le serveur se connecte au client, envoie sa réponse, se déconnecte

- Indépendant du contenu : permet d'envoyer des documents (hyper) texte, du son, des images,...
- Sans état: chaque paire requête/réponse est indépendante (le serveur ne maintient pas d'information sur le client entre les requêtes)
- Protocole en mode texte

Format des messages HTTP

Les messages ont la forme suivante :

- Ligne initiale CR LF
- zéro ou plusieurs lignes d'option CR LF
- CR LF
- Corp du message (document envoyé, paramètres de la requête, …)
- **Requête** : la première ligne contient un nom de méthode (GET, POST, HEAD, …), le paramètre de la méthode et la version du protocole
- **Réponse** : la version du protocole, le code de la réponse (200, 404, 403, …) et un message informatif

Différence entre GET et POST

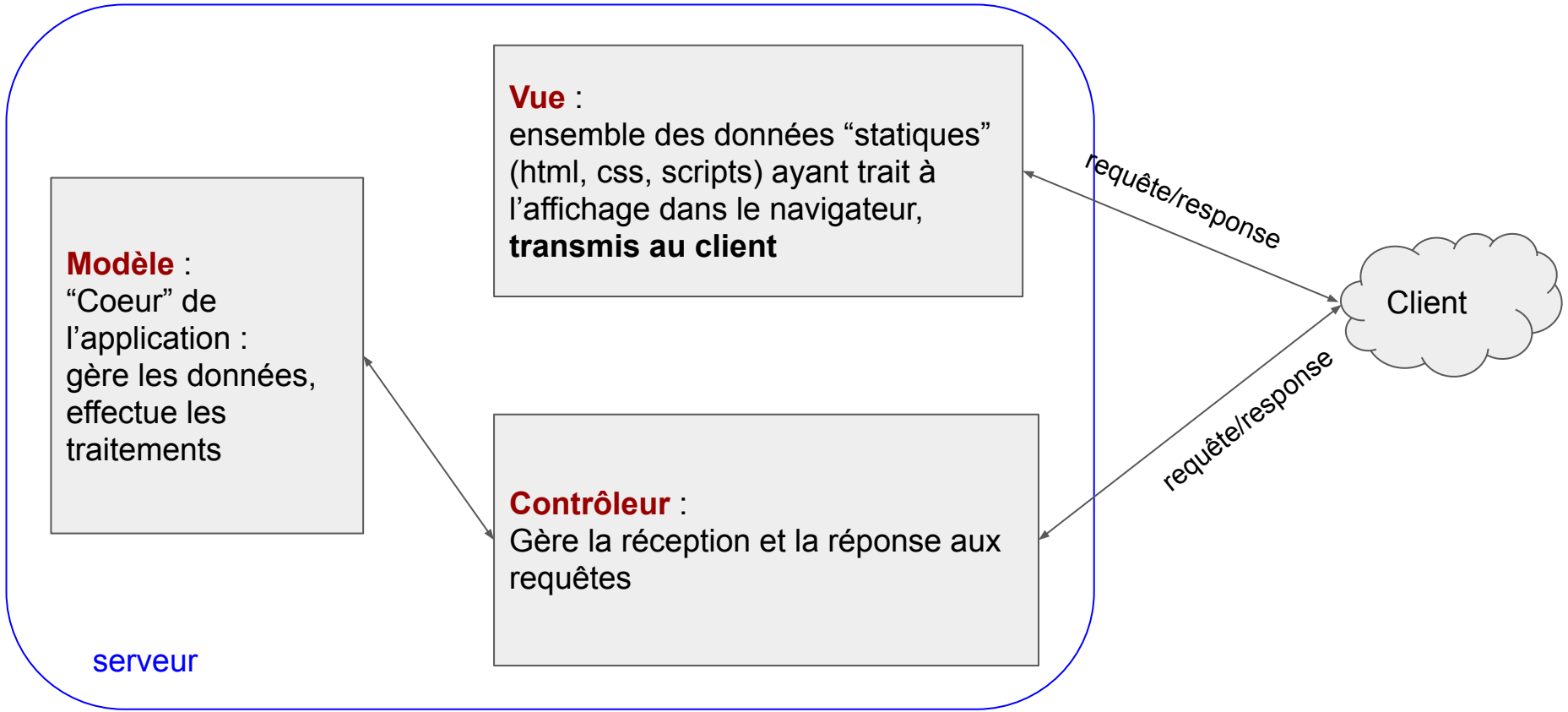
La seule différence entre get et post est la manière dont les paramètres sont passés :

- get : les paramètres sont encodés dans l'url : monurl.com/monAppel.html?val_age=22
- post : les paramètres sont stockés dans le corps de la requête HTTP

Il est donc judicieux d'utiliser post lorsque les paramètres sont longs ou contiennent des caractères ne pouvant pas être présents dans une URL (upload de fichier par exemple).

Application web en java

Modèle-vue-contrôleur



Application web java : war

Une application web est un ensemble d'éléments : classes java, pages et éléments statiques (html, css), scripts, servlets (classes java EE permettant de gérer des requêtes), jar...

Une fois compilés, ces éléments sont regroupés dans un fichier **.war** (**W**eb application **Ar**chive).

Ce .war doit être déployé sur un **serveur d'application**, ou conteneur web, qui s'occupe d'exécuter et de faire tourner "en permanence" l'application (par exemple Apache Tomcat)

Ce que doit faire l’application

- Etre à l’écoute sur un port
- Renvoyer une réponse (typiquement : une vue via http) au client à la demande
- Avoir un ensemble de pages (dynamiques) disponibles
- Calculer ce qu’elle doit calculer

Ce que doit faire l'application

Quel rôle ?

- Etre à l'écoute sur un port (via serveur d'application)
- Renvoyer une réponse (typiquement : une vue via http) au client à la demande

Contrôleur

- Avoir un ensemble de pages (dynamiques) disponibles → Vue
- Calculer ce qu'elle doit calculer → Modèle

Ce que doit faire l'application

Quel rôle ?

- Etre à l'écoute sur un port
- Renvoyer une réponse (typiquement : une vue via http) au client à la demande

Contrôleur

- Avoir un ensemble de pages (dynamiques) disponibles → Vue
- Calculer ce qu'elle doit calculer → Modèle

Tout cela est faisable en natif !
Il n'y a pas ***besoin*** de framework

Framework Spring boot

Le framework Spring boot

C’est un **framework** open source : une “grosse bibliothèque”, générique, imposant une architecture logicielle.

Ce n’est pas indispensable, et il y en a d’autres.

https://spring.io/projects/spring-boot

Pourquoi utiliser un framework ?

Rapidité de mise en place, facilité de compréhension de composants logiciels

Spring boot

C’est un conteneur d’application, qui contient un serveur d’application (Apache Tomcat), et des bibliothèques.

Il permet beaucoup de raccourcis de programmation, notamment grâce à l’**inversion de contrôle** : l’application est avant tout dirigée par le framework, qui fait appel aux éléments que l’on a développé. Beaucoup d’éléments techniques sortent de la responsabilité du développeur.
C’est “l’inverse” de quand le développeur crée une application principale qui fait appel à un élément d’une librairie.

Au démarrage de l’application, on démarre donc une application Spring, qui utilise nos classes

Les annotations

Spring est orienté aspect : les composantes techniques et de configurations classiques sont mis à part du code écrit par le développeur.

Ceci est notamment fait via l'utilisation d'annotation, précédé par un @, sur les classes, attribut et méthodes.
De manière équivalente, ces éléments de configurations peuvent être donné dans un fichier xml.

Au lancement de l'application, les éléments de celles-ci sont scannés pour en extraire ces configurations, et produire du byte code adéquat.

La Servlet : la classe contrôleur

Le contrôleur est le point d'entrée des requêtes HTTP devant être traitées par l'application, hors accès aux pages statiques.

La ou les classes gérant des points d'entrée de requête sont précédées de l'annotation **@Controller** :

```
@Controller
public class MyController {...
```

RequestMapping

Une **méthode** devant pouvoir être **appelée par une requête HTTP**, dans un contrôleur, doit être précédée de l'annotation **@RequestMapping** qui prend en paramètre le chemin (depuis la racine de l'application) et la méthode (GET/POST/PUT…), ou un de ses raccourcis (@GetMapping, @PostMapping,...)

```
@RequestMapping(value = "/cettePage", method = RequestMethod.GET)
        public void myEndPoint(HttpServletRequest request,
                                                    HttpServletResponse response)
```

Argument d’une méthode d’un contrôleur

Les deux arguments de ces méthodes sont :

- HttpServletRequest request : il permet d’accéder aux éléments de la requêtes
    - On accède en particulier à ses paramètres, par String request.getParameter(String nomDuParamètre)
    - Nous verrons d’autres éléments au prochain cours
- HttpServletResponse response : il permet de construire la réponse qui sera renvoyée au client à la sortie de la méthode

Maven - gestionnaire de dépendance

Maven

Apache Maven est un gestionnaire de dépendances pour java et java EE: une manière d'indiquer les librairies, et leur versions, dont on dépend, et de construire le projet.

L'usage peut être vu comme relativement similaire à un "make".

Les éléments sont indiqués dans le pom.xml (l'équivalent du "Makefile")

Utiliser maven donne une structure des sources à utiliser, et simplifie la portabilité de projets entre divers IDE.

La mise à jour du projet (téléchargement des jar qui auraient changer) nécessite une connexion internet

Rien de spécifique au web !

Le contenu du pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<groupId>progWeb</groupId>
	<artifactId>progWeb</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>progWeb</name>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>2.3.3.RELEASE</version>
		<relativePath /> <!-- lookup parent from repository -->
	</parent>
	<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
		</dependency>
	</dependencies>
	<build>
		<plugins>
			<plugin>
				<groupId>org.springframework.boot</groupId>
				<artifactId>spring-boot-maven-plugin</artifactId>
			</plugin>
		</plugins>
	</build>
</project>
```

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
```

Entête xml : désigne le modèle utilisé : celui pour écrire des fichiers POM, en version 4.0.0

```
<groupId>progWeb</groupId>
<artifactId>progWeb</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>progWeb</name>
```

Noms et version de l’arborescence de projet (pour les projets modulaires)

```
<parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.3.3.RELEASE</version>
        <relativePath /> <!-- lookup parent from repository -->
</parent>
```

Le projet parent : ici on se place dans le frameork spring boot donc note application est de type spring boot

```
<dependencies>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
 </dependencies>
```

Les dépendances : la listes des bibliothèque dont l'on dépend, ainsi que leur versions.

L'usage de cet élément facilite grandement la transmission et mise à jour de code par rapport à l'intégration directe de jar !

```
<build>
        <plugins>
                <plugin>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-maven-plugin</artifactId>
                </plugin>
        </plugins>
</build>
```

Les éléments à lier lors de la compilation

Mode développeur - F12

Mode développeur

Dans les navigateurs F12 vous permet d'ouvrir le "mode développeur" ou les "outils de développement". Il vous permet de visualiser différents éléments de la page et des communication réseau.

Nous verrons notamment comment l'utiliser pour :

- observer les requêtes envoyées et réponses reçues
- analyser les éléments html
- utiliser la console (notamment pour tester du javascript)