

Introduction SPRING

esprit 
Se former autrement



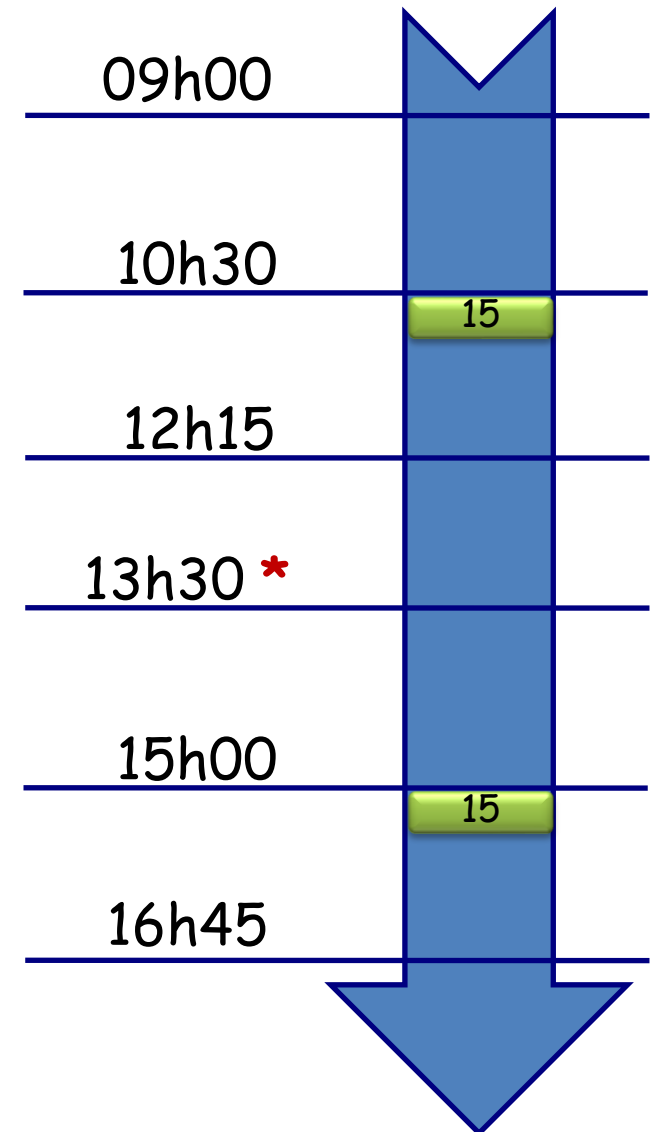
Bureau E204

Plan du Cours

- Introduction : Horaires, Evaluation
- Spring et le Marché de l'Emploi
- Contenu du Module **Spring**
- Historique Spring
- Concurrents Spring
- projets Spring
- Framework Spring
- Serveur web vs. Serveur d'application
- Les architectures physiques
- Présentation et Installation des Outils : JDK, STS, Tomcat, MySQL, Maven, Log4J, Junit

Horaires

- Durée Totale : **42 heures**
 - Séances : **14 séances**
 - Cours : **12 heures**
 - TP : **30 heures**
-
- Durée de chaque Séance : **3 heures**
 - Durée de la Pause : **15 minutes**
-
- * Vendredi : **13h45**



Evaluation

- L'évaluation se fait tout au long du module, et non pas uniquement à la fin.
- La moyenne du module est calculée comme suit :
Moyenne = Note Contrôle Continu * 40% + Note Examen * 60%
- Le **Contrôle Continu** prend en compte l'Assiduité, la Participation, les **TP** à faire en cours ou chez vous et aussi un **Examen Blanc Pratique**.
- L'**Examen** sera écrit par UE ou pratique (selon les conditions).

Evaluation

- Exemple d'évaluation :

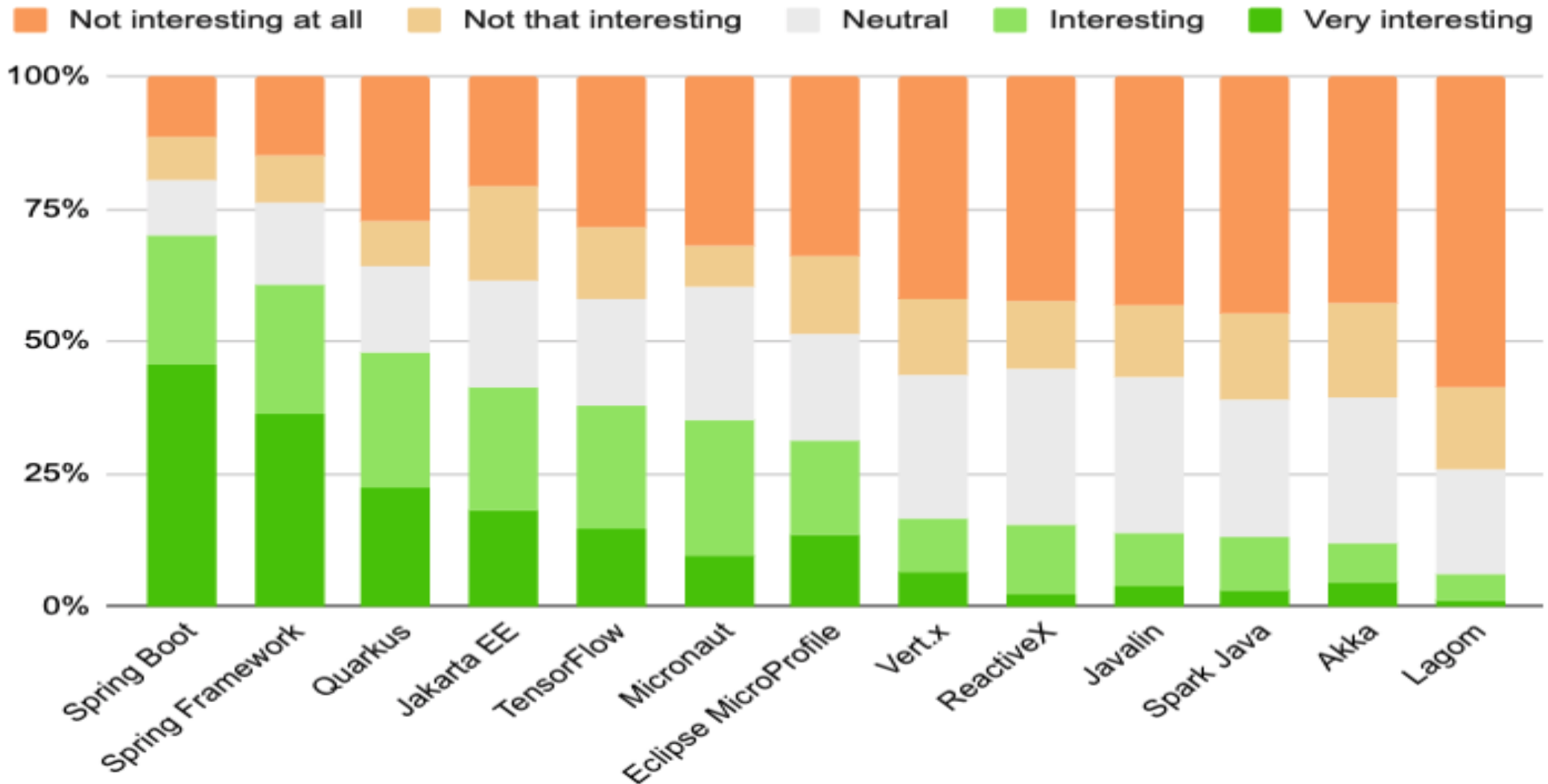
=AF5/14*8+AG5/11*6+MAX(AH5;AI5)/20*6									
AB	AC	AD	AE	AF	AG	AH	AI	AJ	AL
Date	Créneaux	Date	Créneaux	Présences	TP	Q1	Q2	CC	Examen
22/04/2019	3&4	02/05/2019	1&2						
	1 3&4		1 1&2	14	10	0	10	16,45	17,00
	1 3&4		1 1&2	5	10	0	0	8,31	12,00
	1 3&4		1 1&2	8	10	13	19	15,73	8,50
	1 3&4		1 1&2	11	7	15	15	14,60	7,50
	1 3&4		1	11	10	8	18,5	17,29	16,00
	1 3&4		1 1&2	9	2	12	0	9,73	0,00
	1 3&4		1 1&2	10	10	18	15	16,67	13,00
	1 3&4		1 1&2	14	10	9	15	17,95	11,00
	1 3&4		1 1&2	0	0	0	0	0,00	6,00
	1 3&4		1 1&2	14	10	17	20	19,45	5,50
	1 3&4		1 1&2	14	11	17	20	20,00	13,50

Spring et le Marché du Travail

- Les offres d'emploi « **Cherche Développeur Java** », en général cherchent un développeur dans les technologies suivantes :
- Front-End : Angular, **Spring MVC**, VueJS, ReactJS ...
- Back-End : **Spring**, EJB, Hibernate, **Spring Data JPA**, Quarkus
- Outils : **Spring Boot**, **Maven**, GIT, Jenkins, Sonar, ...
- Base de Données : PostgreSQL, Oracle, **MySQL**, ...
- Méthode Agile : Scrum, extreme Programming
- <https://www.tanitjobs.com/job/881048/d%C3%A9veloppeur-java-spring-boot-exp%C3%A9riment%C3%A9/?backPage=1&searchID=1630937137.7228>

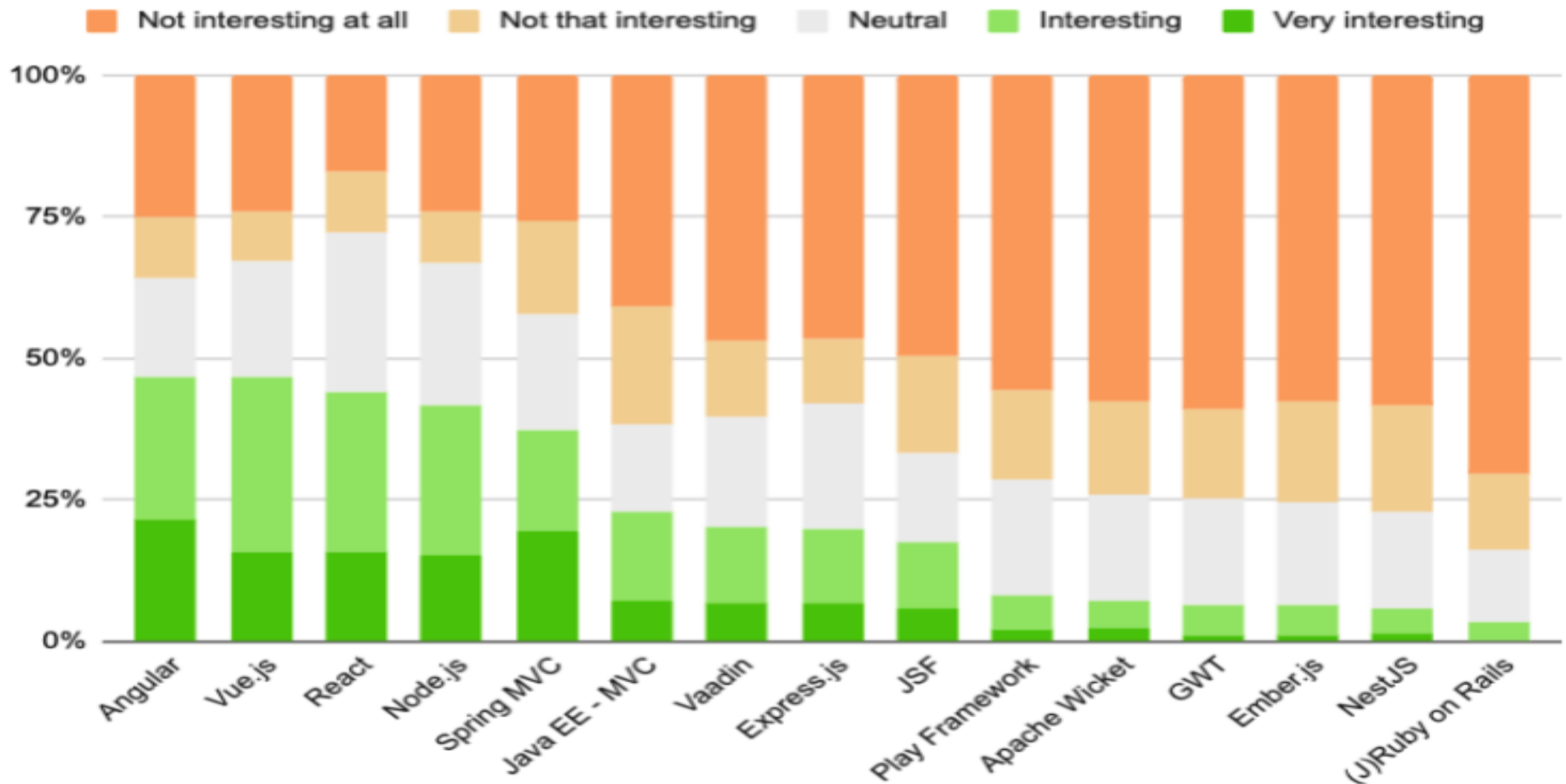
Spring et le Marché de l'Emploi

Application Frameworks



Spring et le Marché de l'Emploi

Web Frameworks



Contenu du Module Spring

- Introduction à **Spring** et Mise en place de l'Environnement
- **Maven**
- **IoC & DI** (Injection des dépendances)
- **Spring Boot**
- **Spring Data JPA** (Première Entité - Associations - CRUD / JPA Repository – JPQL)
- **Lombok**
- **Spring MVC REST (Postman+ Swagger)**
- **Tests unitaires (JUnit , mockData)**
- **TP gestion Magasin et Stock**
- **Spring Scheduler**
- **Spring Batch**
- **AOP** (Aspect Oriented Programmation)
- **Spring Security**
- **Examen Blanc Pratique** (Spring Boot - REST - Spring Data – AOP-etc..)

Bref Historique JavaEE

- **1995** : Java est apparu, développé par **James Gosling** chez **Sun Microsystems**.
- **1997** : Les servlets arrivent pour la création des pages web dynamiques.
- **1999** : Les JSP arrivent pour **faciliter** la création des pages web dynamiques et résoudre les problèmes de servlets.
- Les JSP séparent les contrôleurs de la partie présentation.
- La question qui se pose, comment les servlets/JSP vont assurer, la logique métier et la persistance ?
- **Toujours en 1999** : J2EE vient comme réponse à la question précédente (transactions, sécurité, messaging ...) en introduisant les **EJB**.
EJB promettait sécurité et disponibilité mais la réalité était tout autre (couplage fort avec le serveur d'application, temps de configuration considérable...).

Bref Historique JavaEE

- 1999 : **J2EE** (Java 2 Enterprise Edition) depuis la version 1.**2** et jusqu'à la version 1.4 en 2003).
- 2006 : **JavaEE** 5 ou **JEE** (Java Enterprise Edition) depuis la version 5 (Ajout des annotations)
- 2010 : Sun a été rachetée par Oracle en 2010 (Java est à la base une propriété de **Sun** Microsystems)
Oracle a cédé la plateforme JavaEE à **Eclipse Foundation**, mais en exigeant le changement de son nom
- 2017 : Java EE 8 (version obsolète)
- 2019 : Jakarta EE 8 (Nouveau nom JakartaEE à la place de JavaEE)

Historique Spring

- **2002 : Rod Johnson** publie son livre «Expert One-on-One J2EE Design and Development», dans lequel il propose du code, qui va devenir plus tard le Framework Spring
- **2004** : Rod Johnson publie son livre «**J2EE Development without EJB**».
- **2004** : Spring 1.0, licence Apache 2.0
- **2005** : Spring devient populaire, en particulier en réaction par rapport aux EJB 2.x

=> **La configuration de l'application par la main rendait la manipulation de Spring difficile.**

- **2006** : Spring 2.0 (Introduction de l'injection de dépendances)
- **2007** : Spring 2.5, avec support des **annotations**
- **2009** : Spring 3.0
- **2013** : Spring 4.0
- **2017** : Spring 5.x,
- **Avril 2021** : Spring 5.3.6 (version courante)

Concurrents Spring

Jakarta EE

vs.

Spring

vs.

Quarkus



Concurrents Spring

Jakarta EE

- Jakarta EE 8, comme son nom l'indique, est fonctionnellement identique à Java EE 8, publié par Oracle et possède les mêmes spécifications.
- Jakarta EE a été optimisé pour le **cloud computing**.
- Selon la fondation Eclipse, l'un des principaux objectifs de Jakarta EE est d'accélérer le développement d'applications d'entreprise pour le Cloud Computing (les applications cloud natives).



Concurrents Spring

Jakarta EE

- La plateforme **JakartaEE** est en pleine mutation et modification. Attendons qu'elle soit stable.
- La plateforme **JavaEE** est un ensemble de spécifications **JSF, EJB, JPA**. Elle n'est plus maintenue par Oracle, donc bientôt obsolète.
- Le cœur de cette plateforme est les EJB.
- Inconvénients des EJB :
 - Difficile à coder, il faut implémenter des interfaces spécifiques ...
 - Les tests unitaires sont difficiles à réaliser.
 - C'est une solution qui nécessite un serveur d'application ⇒ lourd et gourmand en ressources.

Concurrents Spring

Jakarta EE vs Spring

- Le choix d'une technologie dépend de plusieurs critères dont les fonctionnalités offertes et l'expérience du développeur (un développeur avec 10 ans d'expérience JavaEE te dira toujours que JakartaEE est le meilleur peut importe les concurrents (sentiment d'appartenance)).
- Les défenseurs de JakartaEE mentionneront que le développement de la plateforme n'est pas lié à une seule compagnie comme c'est le cas pour Spring. Il s'agit plutôt d'une communauté qui assure la longévité de leurs projets pour les années à venir.
- Les pro-Spring vous diront qu'au contraire Spring est plus fiable sur la durée et que le fait qu'il est très demandé sur le marché (beaucoup plus que JakartaEE) est une raison évidente pour choisir ce Framework.

Concurrents Spring

Quarkus

Quarkus est un framework Java natif Kubernetes conçu pour les machines virtuelles Java (JVM) optimisant Java spécifiquement pour les conteneurs et lui permettant de devenir une plate-forme efficace pour les environnements sans serveur, cloud et Kubernetes.

Tout comme Spring, Quarkus a été conçu pour être facile à utiliser dès le départ, avec des fonctionnalités avec peu ou pas de configuration.



Concurrents Spring

Quarkus vs. Spring

Spring et Quarkus sont en concurrence sur les critères **du temps du lancement** et **l'occupation de la mémoire**.

L'évolutivité et l'architecture **sans serveur** sont aussi des critères pris en considération dans les dernières versions fournies par ces deux frameworks.

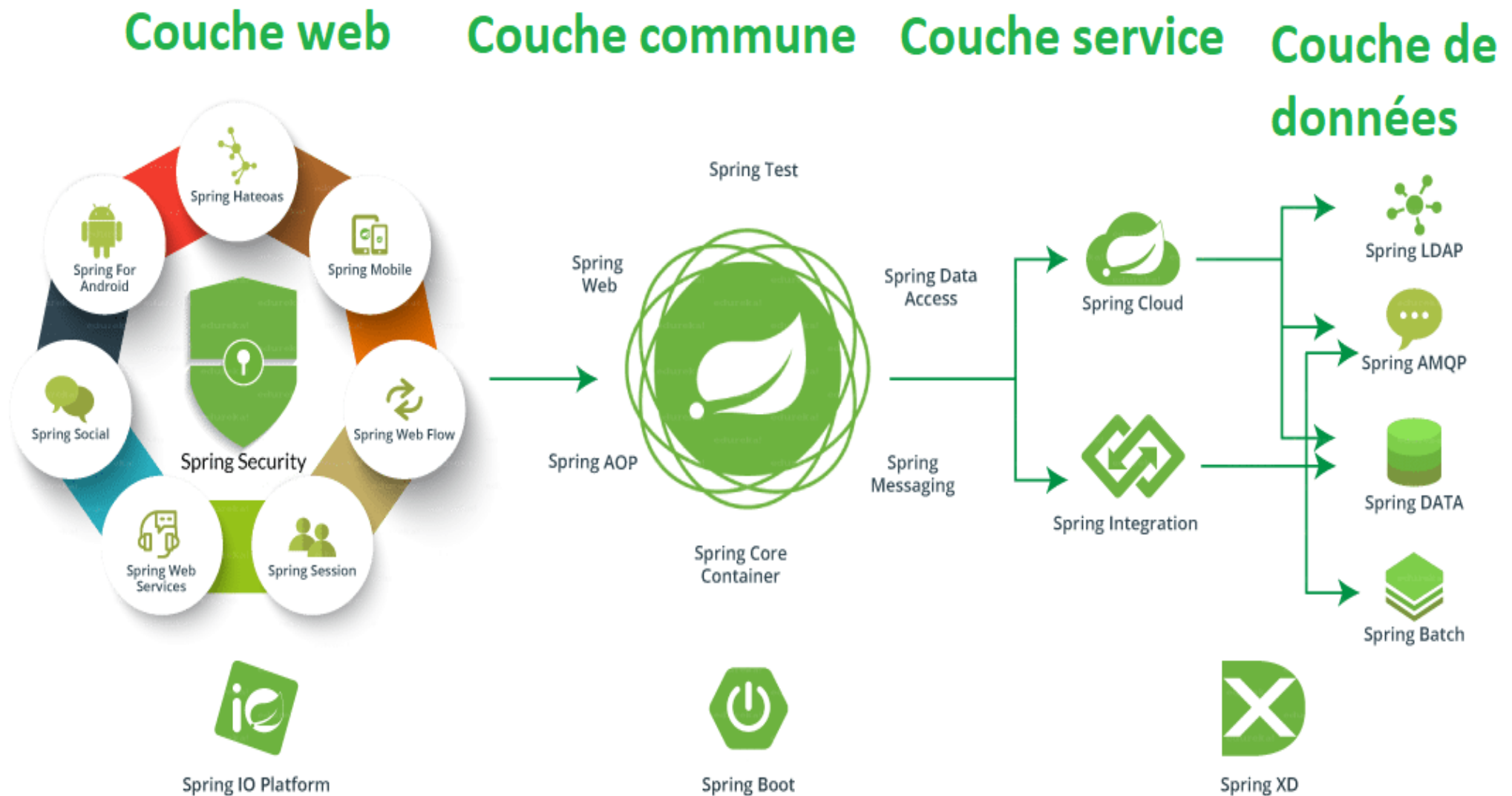
Concurrents Spring

Basé sur la comparaison et la demande du marché, nous allons nous focaliser dans notre cours sur **Spring** et utiliser une implémentation de JPA fournie par Spring : **Spring Data JPA**.

Les Projets Spring

- <https://spring.io/projects> :
- **Spring est un ensemble de projets.**
- Spring a une vaste communauté et une bonne documentation.
- Le code est sous license Apache 2, il est Open Source et disponible sur GitHub.

Les Projets Spring



Les Projets Spring

- **Spring Framework (Spring Core)**: C'est le cœur des projets Spring. Ce framework contient les fonctionnalités de base de Spring. Ce framework contient **Spring MVC**.
- **Spring Batch** : permet le développement des applications de type batch qui peuvent gérer de gros volumes de données
- **Spring Integration** : il s'agit d'un ESB (Enterprise Service Bus) pour interconnecter les applications d'une entreprise
- **Spring Android** : a pour but de faciliter le développement d'applications Android
- **Spring Boot** : C'est un outil Spring qui vous permet d'embarquer un serveur Tomcat dans votre livrable, et simplifier la livraison et le test de votre application.
- **Spring Data JPA** : fournit une implémentation de la couche d'accès aux données.
- **Spring Security** : permet de gérer l'authentification et les habilitations d'une application web.

Framework Spring (Spring Core)

- C'est quoi un **Framework** :
 - C'est un **cadre de développement**
 - Contient des «bonnes pratiques»
 - Permet d'éviter de recoder des classes utilitaires
 - Permet de se focaliser sur le métier
- Spring fournit la «plomberie» : le socle technique
- Les développeurs peuvent se concentrer sur le code métier (le vrai travail)
- « Attention un **framework** ne doit pas être considéré comme une **plate-forme**, dans la mesure où il n'intègre pas d'environnement d'exécution système ou applicatif. »

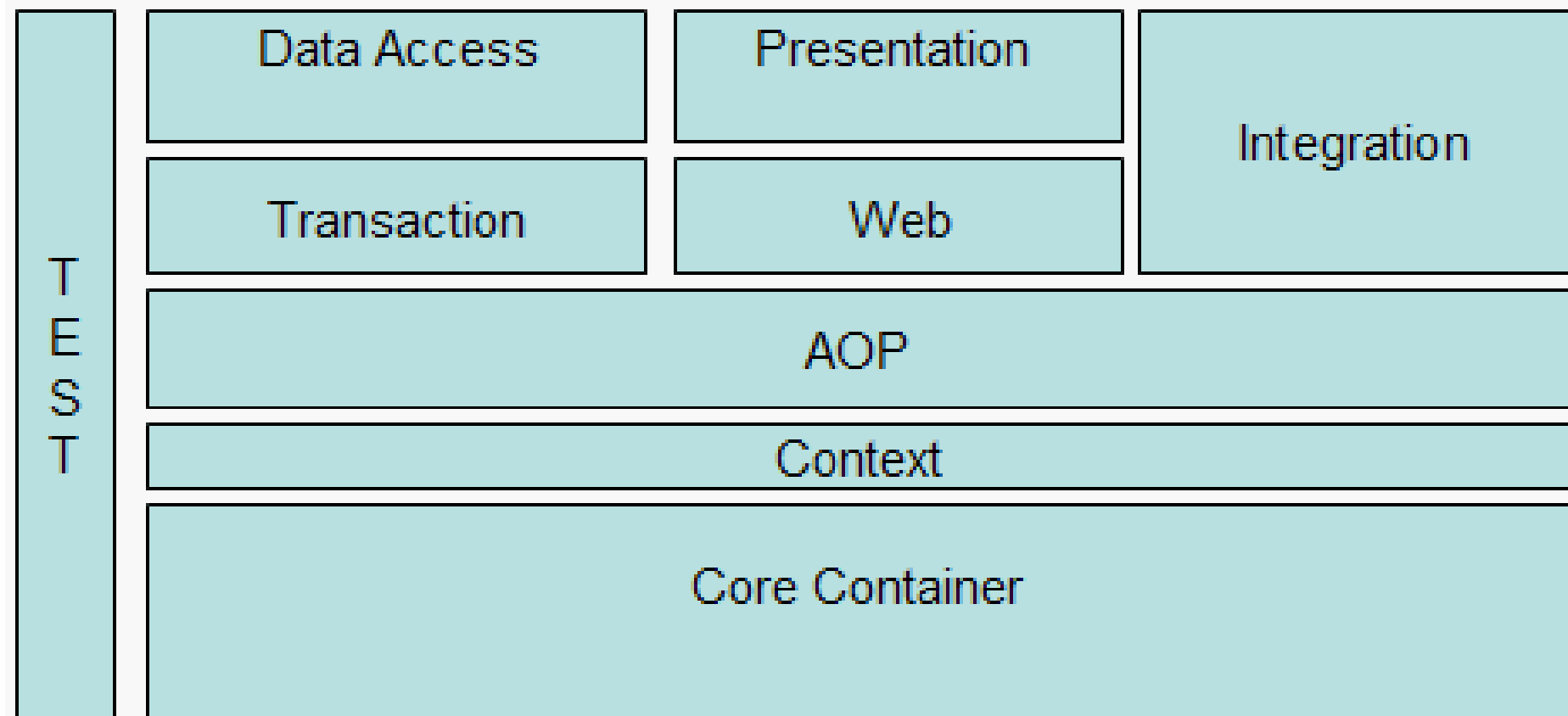
Framework Spring

- Le Framework Spring est un socle pour le développement d'applications.
- Il fournit de nombreuses fonctionnalités.
- C'est l'un des Frameworks les plus répandus dans le monde Java.
- Framework Open Source
- **Framework Spring (Spring Core)** : <http://projects.spring.io/spring-framework>

Framework Spring

- Le but de Spring est de faciliter et de rendre productif le développement d'applications.
- Il fournit beaucoup de fonctionnalités :
 - **Injection de Dépendances** (IoC : Inversion de Control).
 - **AOP** : Aspect Oriented Programming : Injection de Code en RunTime.
 - **Data Access** : Permet de simplifier l'accès aux données (DAO, ORM, Transaction, ...).
 - **Web** : Permet de développer des interfaces web évoluées (MVC).
 - **Testabilité** de l'application facilitée
 - **Intégration** : Spring offre un ESB, qui permet l'intégration et la communication entre les applications (EAI).

Framework Spring

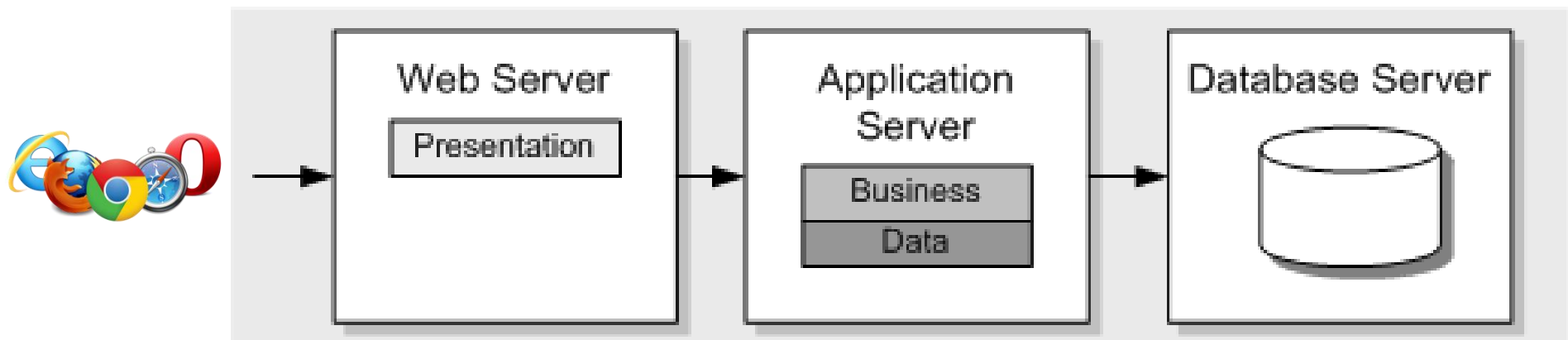


Architecture logique

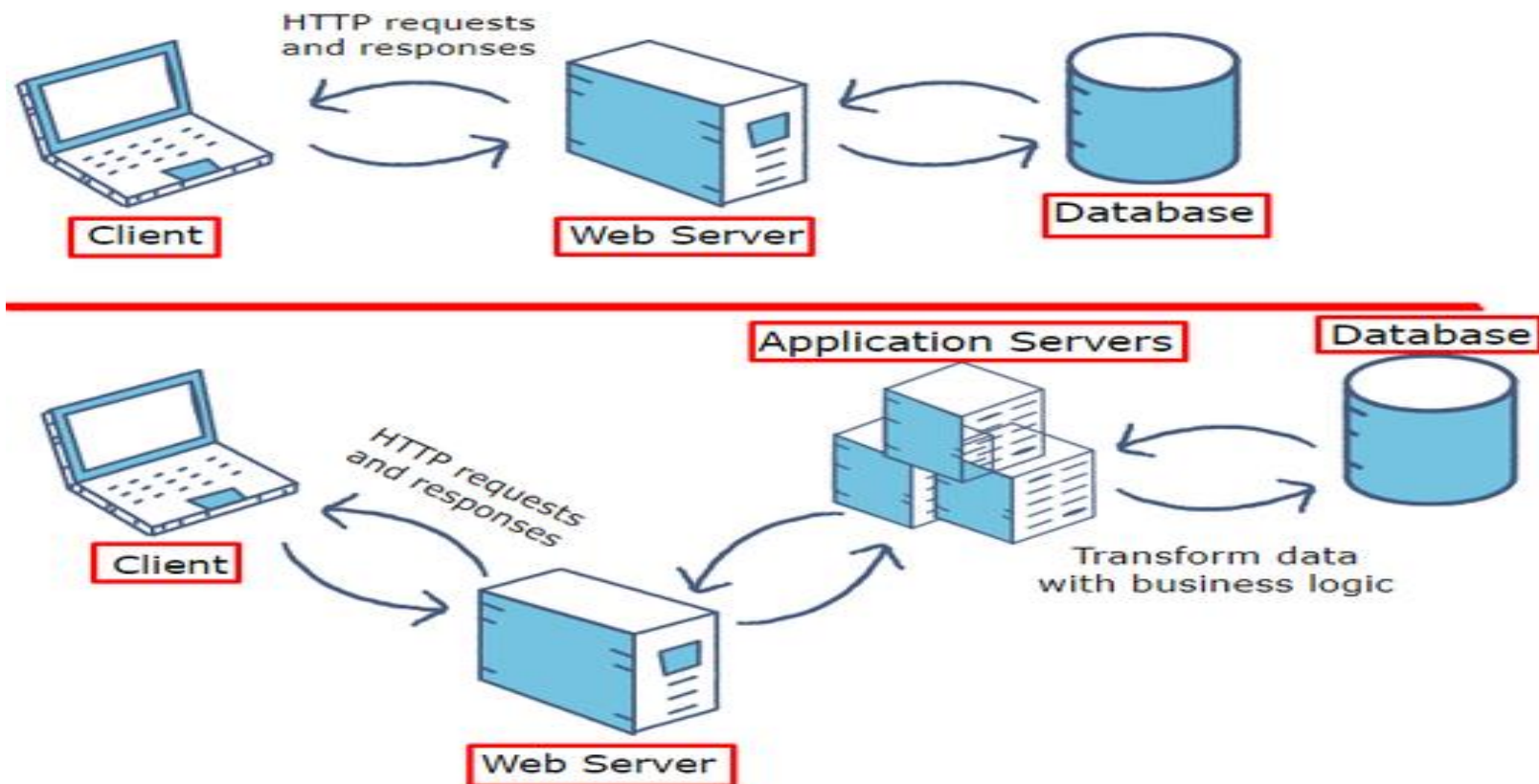
- Une application typique utilisant Spring est généralement structurée en trois couches :
 - Couche **Présentation** : (Web + Contrôleur)
 - Couche **Service** : interface métier avec mise en œuvre de certaines fonctionnalités.
 - Couche **Accès aux Données** : recherche et persistance des objets.
- **Spring est un Framework utilisé pour créer et injecter les objets requis pour communiquer entre les différentes couches.**

Serveur Web vs Serveur d'Application

Serveur Web	Serveur d'application JavaEE Serveur web + container
Héberge que la couche présentation et l'expose qu'à travers le protocole HTTP(S)	Héberge la logique métier et peut aussi héberger la couche présentation (supporte différents protocoles : HTTP, JNDI, ...).
Ne peut pas inclure un EJB Container.	Doit inclure un EJB Container.
lightweight	Relativement gourmand en ressources (CPU, RAM et Disk).
Exp : Apache HTTP Server, Tomcat, Jetty	Exp : Wildfly, WebSphere ...

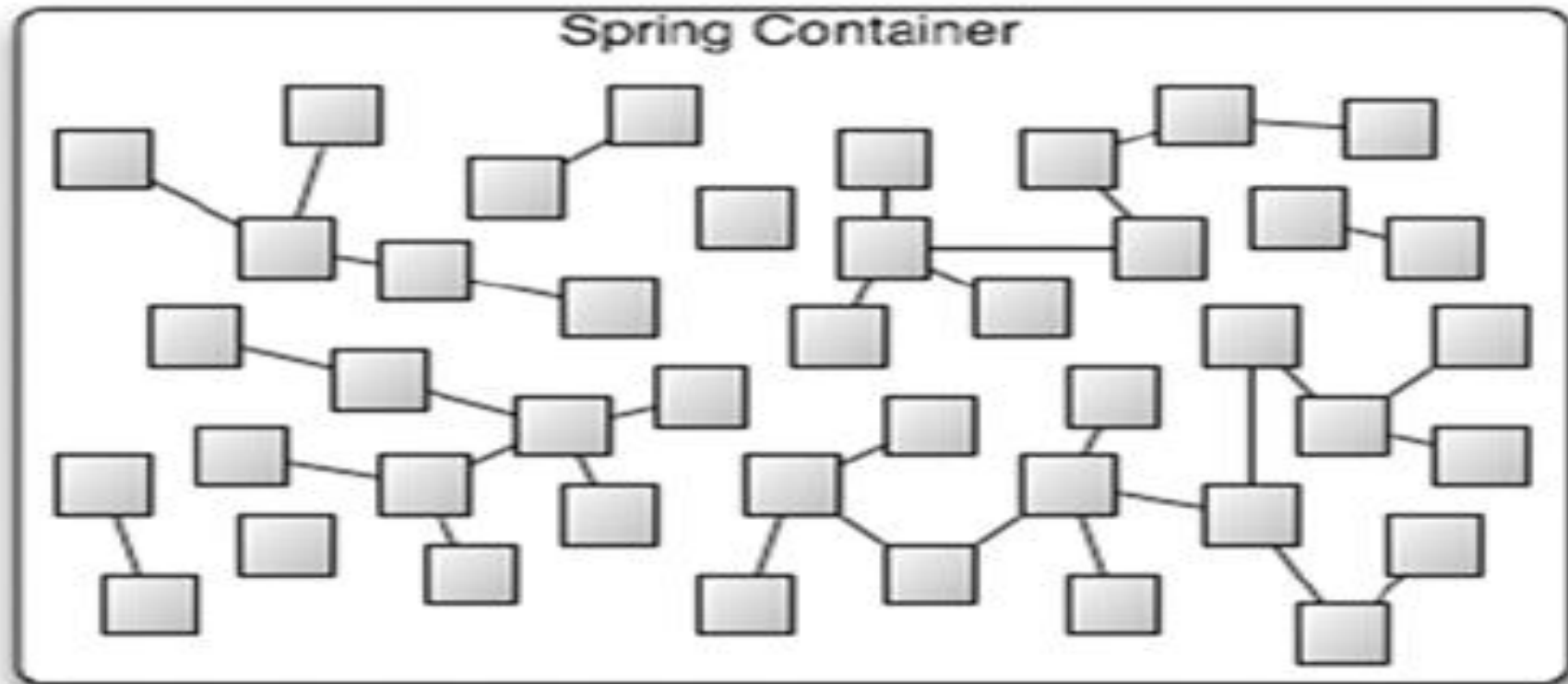


Serveur Web vs Serveur d'Application



Serveur Web vs Serveur d'Application

Spring IOC Container



Dans une application Spring, les objets sont créés, sont liés ensemble et communiquent dans le Spring IOC Container.

Architecture Physique

- **Tier** est un mot anglais qui signifie étage ou niveau.
- Une application peut être **1-Tier**, **2-Tiers**, **3-Tiers** ou **N-Tiers**.

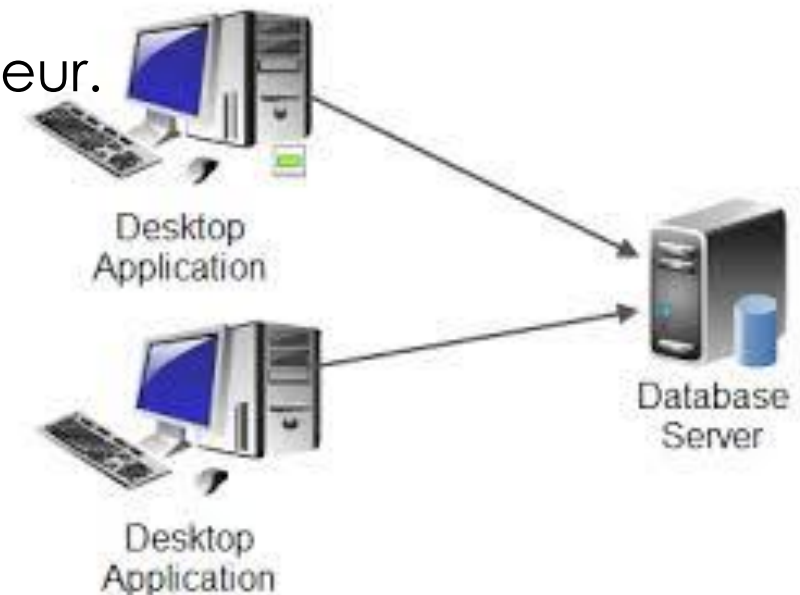
Architecture Physique - 1-Tiers

- Une application **1-Tier** est, par exemple, la Modification d'un document Word sur un ordinateur Local.
- Tout est sur la même machine et les couches sont fortement liées.
- Inconvénients : Risque de perte des données (non sauvegardées à distance), Impossible d'accéder à une même ressource par deux utilisateurs en même temps.



Architecture Physique - 2-Tiers

- Une application **2-Tiers** est typiquement une application **client lourd**.
- Le niveau **Présentation (IHM)** et le niveau **Traitement** sont sur la machine de l'utilisateur.
- Le niveau **Base de Données** est sur un autre serveur.
- C'est une architecture **Client / Serveur**.
- Client = demandeur de ressource
- Serveur = fournisseur de ressource



Inconvénients

- Toute mise à jour des fonctionnalités nécessitent un déploiement sur toutes les machines des utilisateurs.
- Le serveur ne fait pas appel à une autre application pour fournir le service.

Architecture Physique - 3-Tiers

- Une application **3-Tiers** introduit un niveau intermédiaire (middleware) entre le client et le serveur.
- Le niveau intermédiaire est chargé de fournir la ressource en faisant appel à un autre serveur.

Avantages

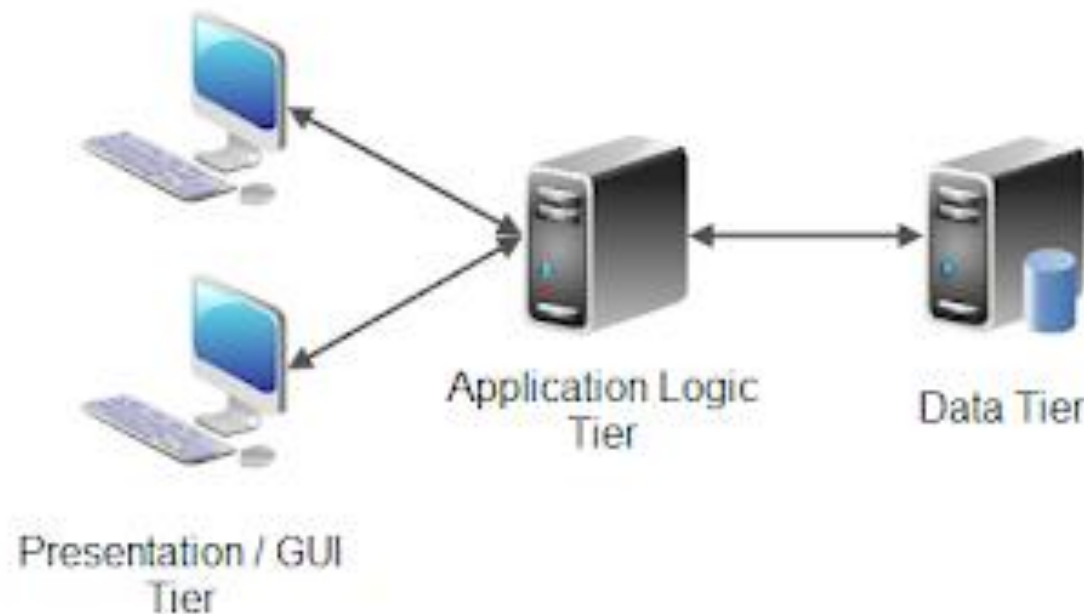
- Centraliser la logique application sur un serveur HTTP

Inconvénients

- Le serveur HTTP (élément principale de l'architecture)est fortement sollicité d'où une charge de demandes provenant à la fois du client et du serveur.
- Bien que cette architecture résout le problème du client lourd de l'architecture deux tiers, le soulagement du client est remplacé par un serveur fortement sollicité.

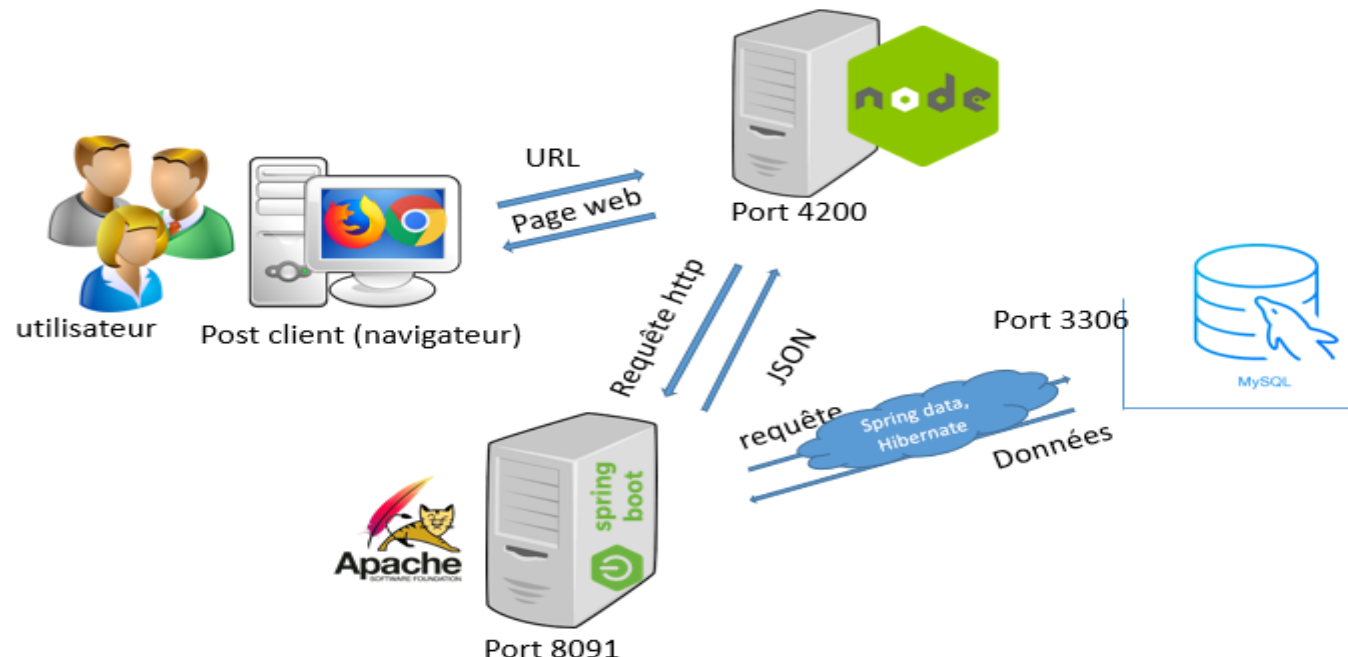
Architecture Physique - 3-Tiers

- Une application **3-Tiers** est typiquement une application Web :
 - Niveau **Présentation** : IHM (Navigateur sur la machine de l'utilisateur)
 - Niveau **Traitement** : Un serveur web (Tomcat, ...) qui contient le WAR de notre application.
 - Niveau **Base de données** : Un serveur de BD qui stocke les données de notre application.



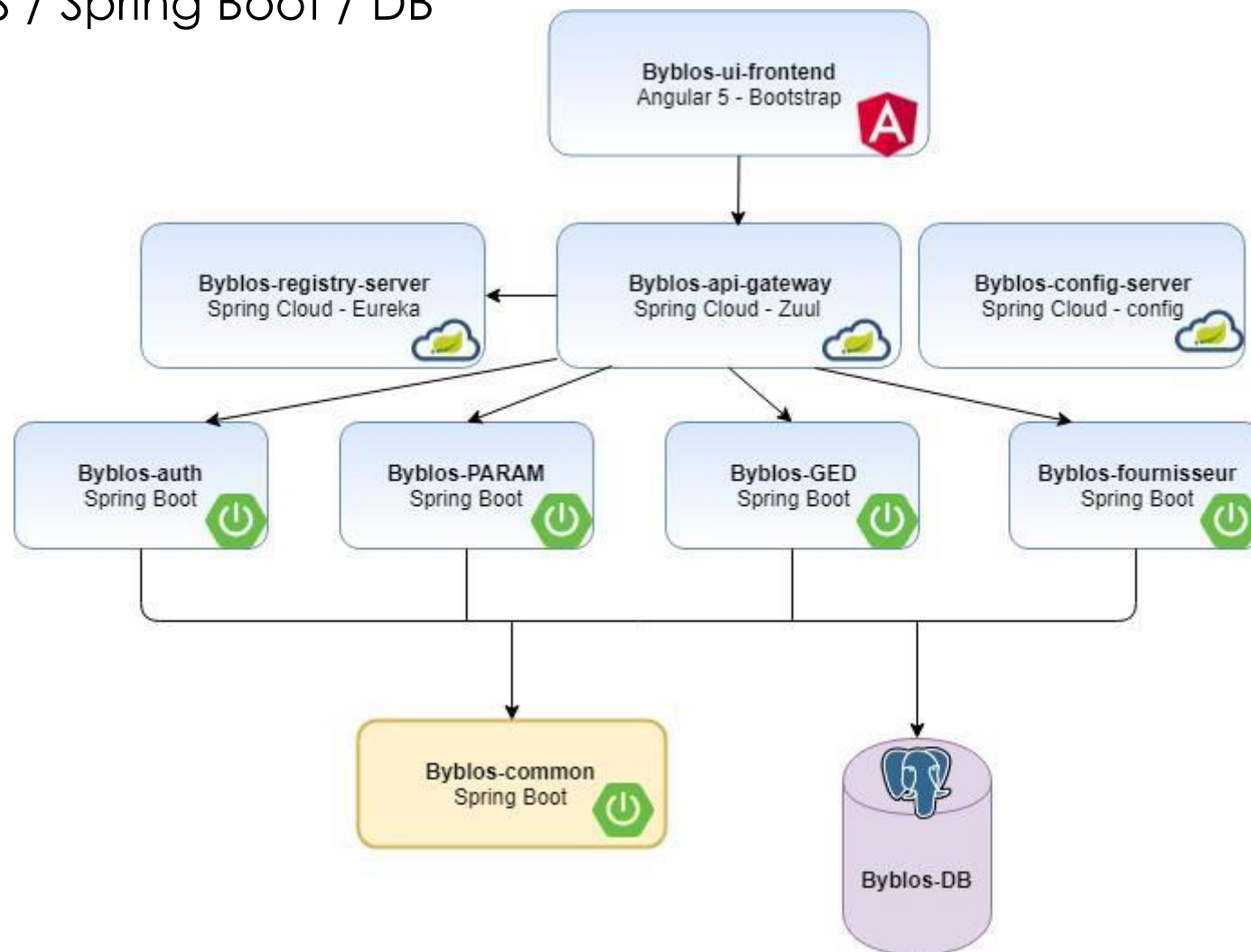
Architecture Physique - N-Tiers

- L'architecture N tiers assure un équilibre de charge entre le client et le serveur par l'introduction de nouvelles couches.
- Voici une architecture 4-Tiers d'une application web développée par un étudiant Esprit pendant son projet de fin d'étude (GUI – Angular sur le Serveur NodeJS – Spring Boot (Serveur Web Tomcat embarqué) – Serveur de base de données MySQL) :



Architecture Physique - N-Tiers

- Voici une architecture 4-Tiers, **en Micro-Servcies**, d'une application web développée par un étudiant Esprit pendant son projet de fin d'étude : GUI / NodeJS / Spring Boot / DB

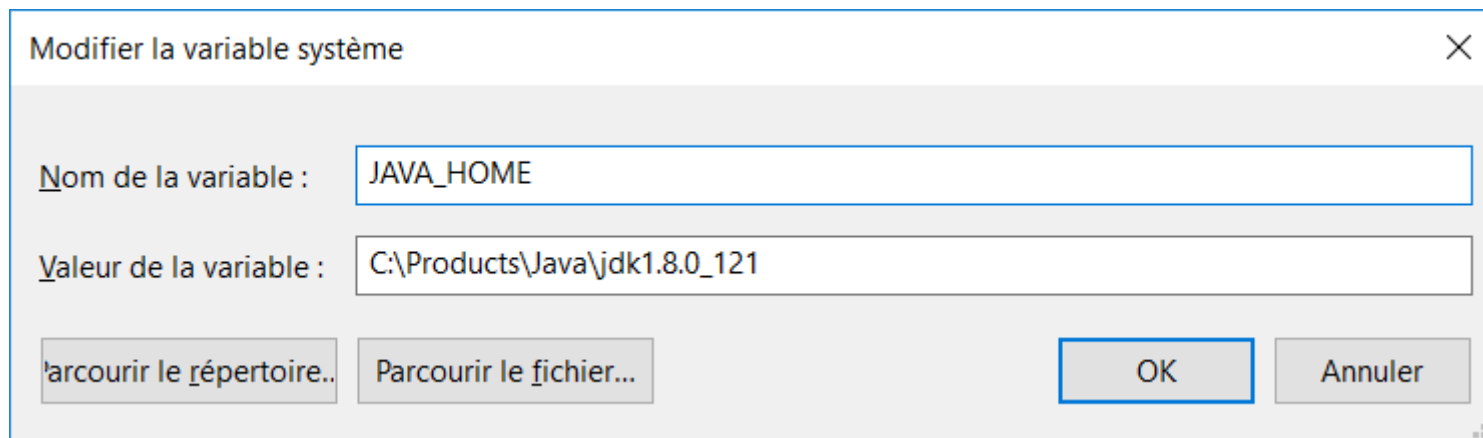


Installation des Outils

- Les outils suivant vont nous permettre de développer des applications Web avec Spring.
- Avant de faire l'installation, vérifiez que vous n'avez pas ces outils déjà installés. Si c'est la cas, pas besoin de les réinstaller (vous pouvez utiliser d'autres versions) :
- **JDK 8** - version 1.8.0.060 - (1.8 -peu importe la version mineure- déjà installé sur vos machine normalement)
- **Tomcat 8.5.28**
- **STS 3.8.4 (Eclipse 4.6.3)**
- **WAMP 3.1 ou XAMP** (déjà installé sur vos machine normalement), pour avoir **MySQL 5.6.17**
- **Maven 3.5.0** (existe déjà en tant que plugin Eclipse, rien à installer).
- Par souci d'homogénéité, vous pouvez créer un dossier **C:\Products**, dans lequel, vous allez installer tous les outils que nous allons utiliser par la suite.

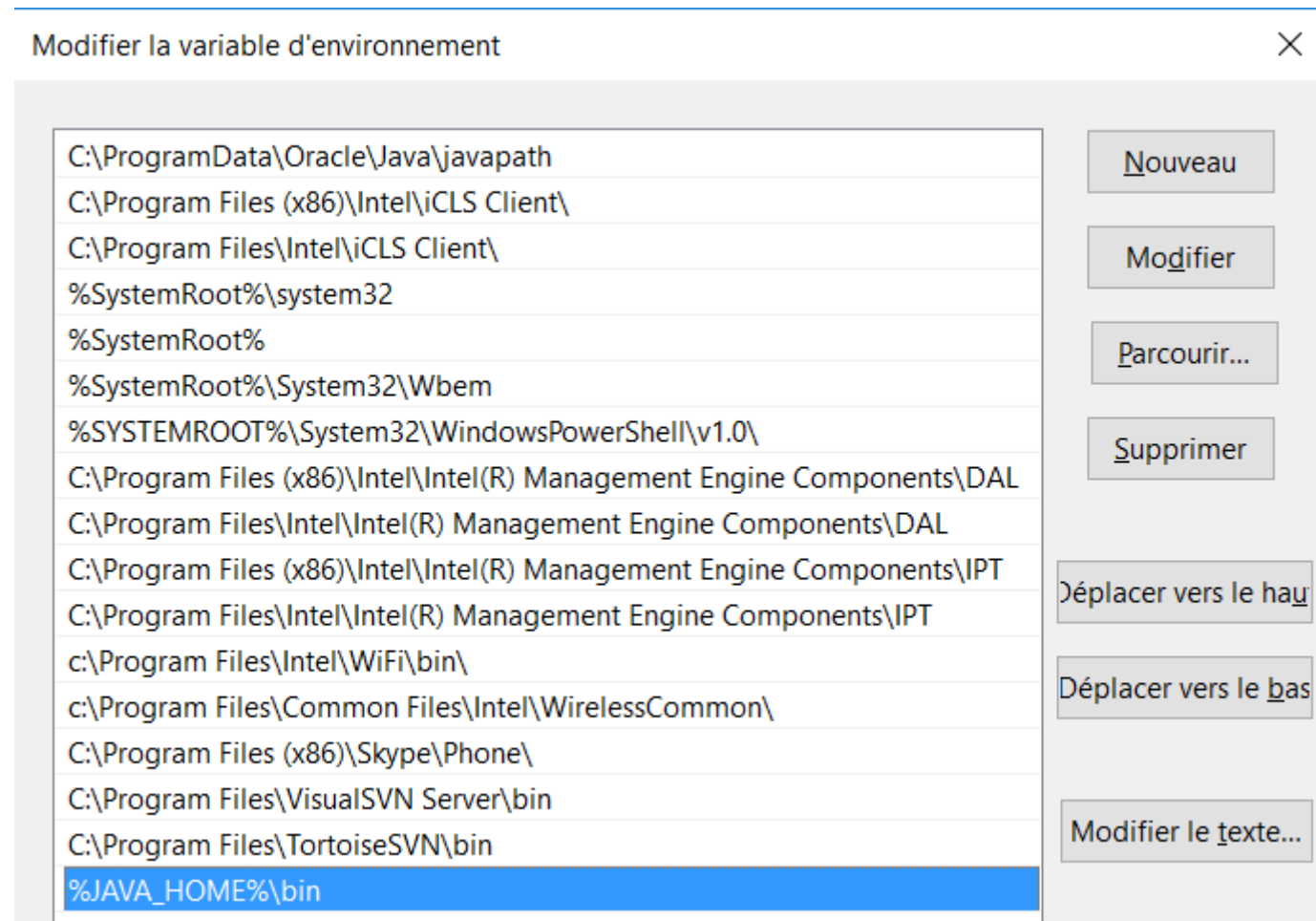
Installation JDK 8

- Installer la JDK dans **C:\Products\java\jdk-1.8.0.60**
- Installer la JRE dans **C:\Products\java\jre-1.8.0.60**
- La JRE est installée en même temps que la JDK.
- Créer la variable d'environnement **système : JAVA_HOME** qui contient le chemin de la JDK:



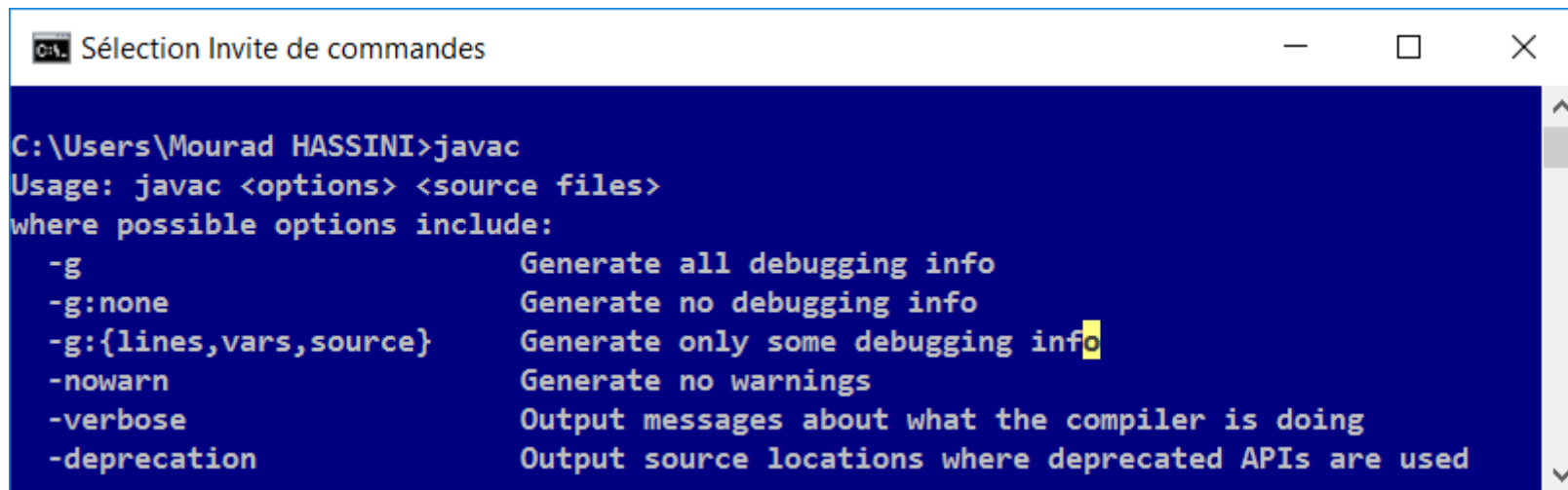
Installation JDK 8

- Ajouter **%JAVA_HOME%\bin** au Path :

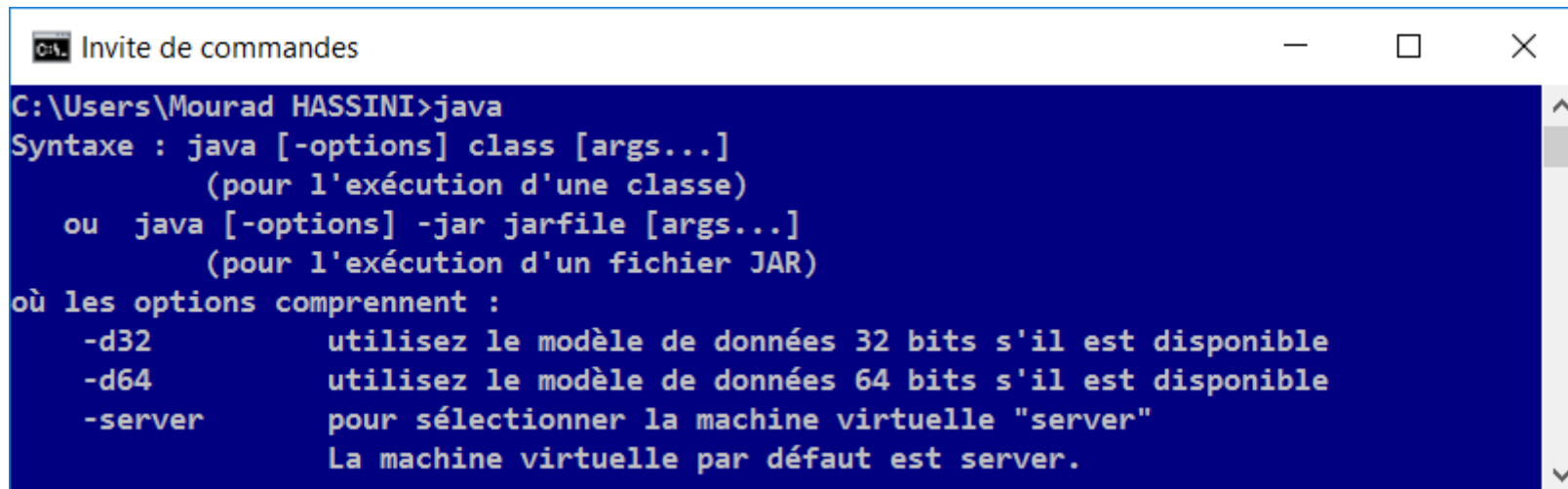


Installation JDK 8

- Vérifier que Java est bien installé et que la variable d'environnement est bien positionnée (**javac** et **java**) :



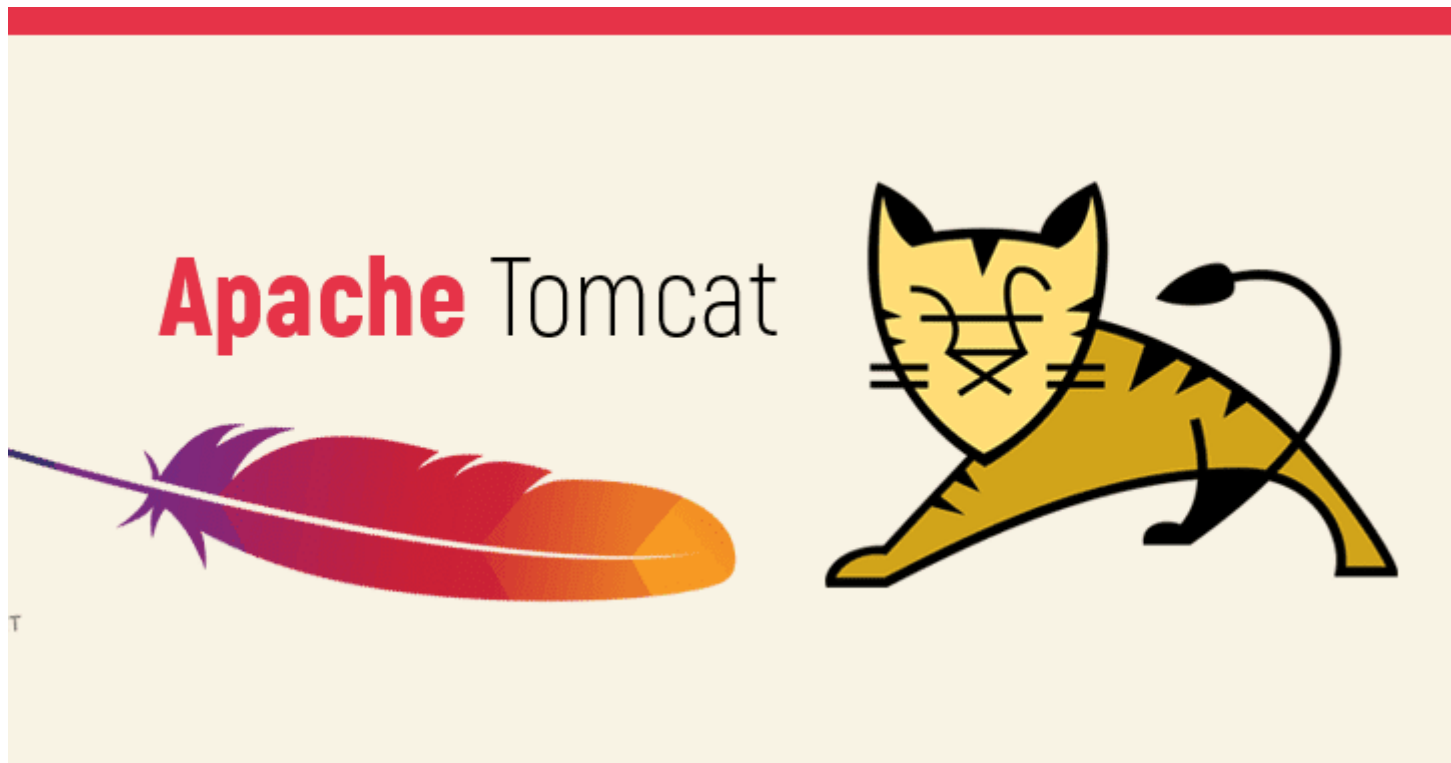
```
C:\Users\Mourad HASSINI>javac
Usage: javac <options> <source files>
where possible options include:
    -g                Generate all debugging info
    -g:none           Generate no debugging info
    -g:{lines,vars,source}  Generate only some debugging info
    -nowarn           Generate no warnings
    -verbose          Output messages about what the compiler is doing
    -deprecation      Output source locations where deprecated APIs are used
```



```
C:\Users\Mourad HASSINI>java
Syntaxe : java [-options] class [args...]
           (pour l'exécution d'une classe)
ou java [-options] -jar jarfile [args...]
           (pour l'exécution d'un fichier JAR)
où les options comprennent :
    -d32             utilisez le modèle de données 32 bits s'il est disponible
    -d64             utilisez le modèle de données 64 bits s'il est disponible
    -server          pour sélectionner la machine virtuelle "server"
                    La machine virtuelle par défaut est server.
```

Installation Tomcat

- Il suffit de dézipper le fichier **apache-tomcat-8.5.28-windows-x86.zip**



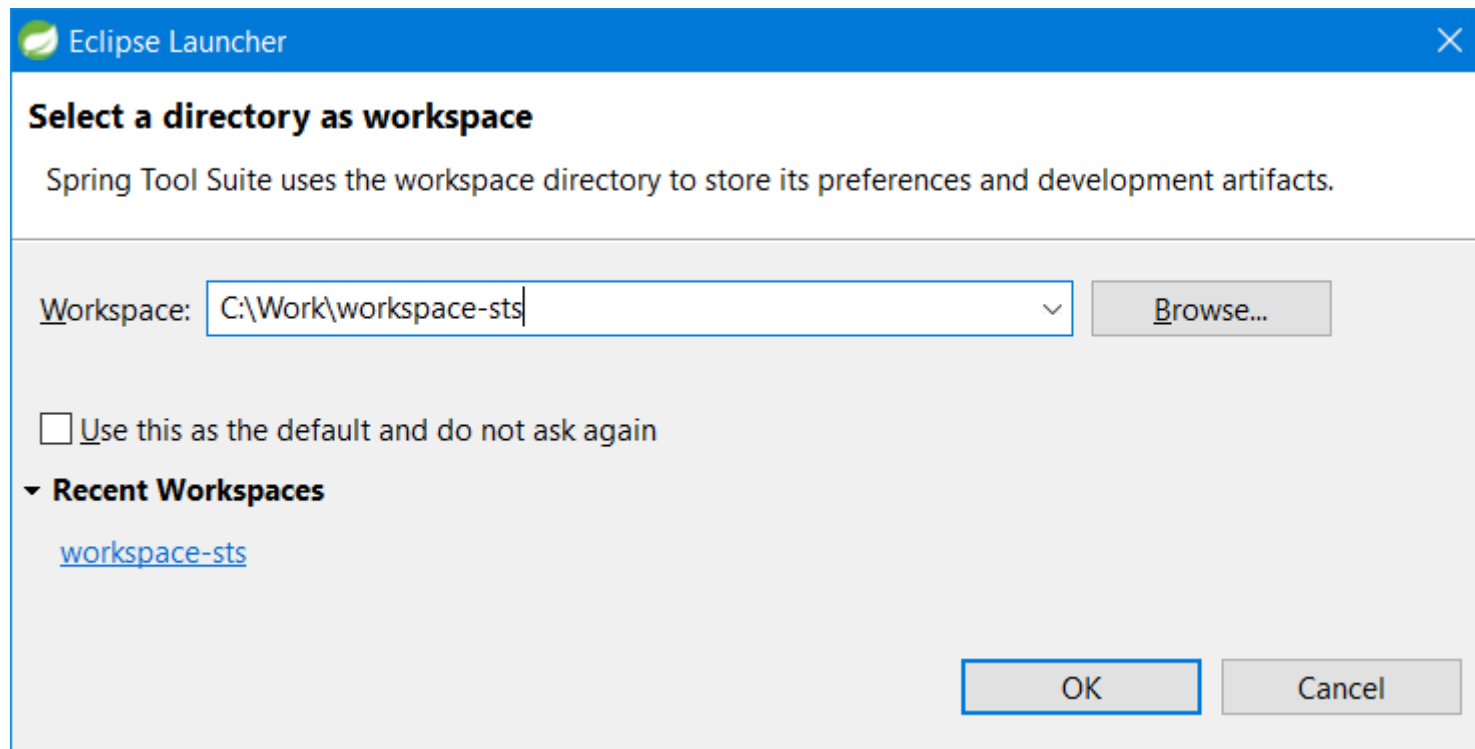
Installation STS (Spring Tool Suite)

- Il suffit de dézipper le fichier **spring-tool-suite-3.8.4.RELEASE-e4.6.3-win32-x86_64.zip**
- Si vous n'avez pas le fichier vous pouvez le télécharger :
Téléchargement STS 3.8.4 : <http://spring.io/tools/sts/all>



Configuration IDE STS

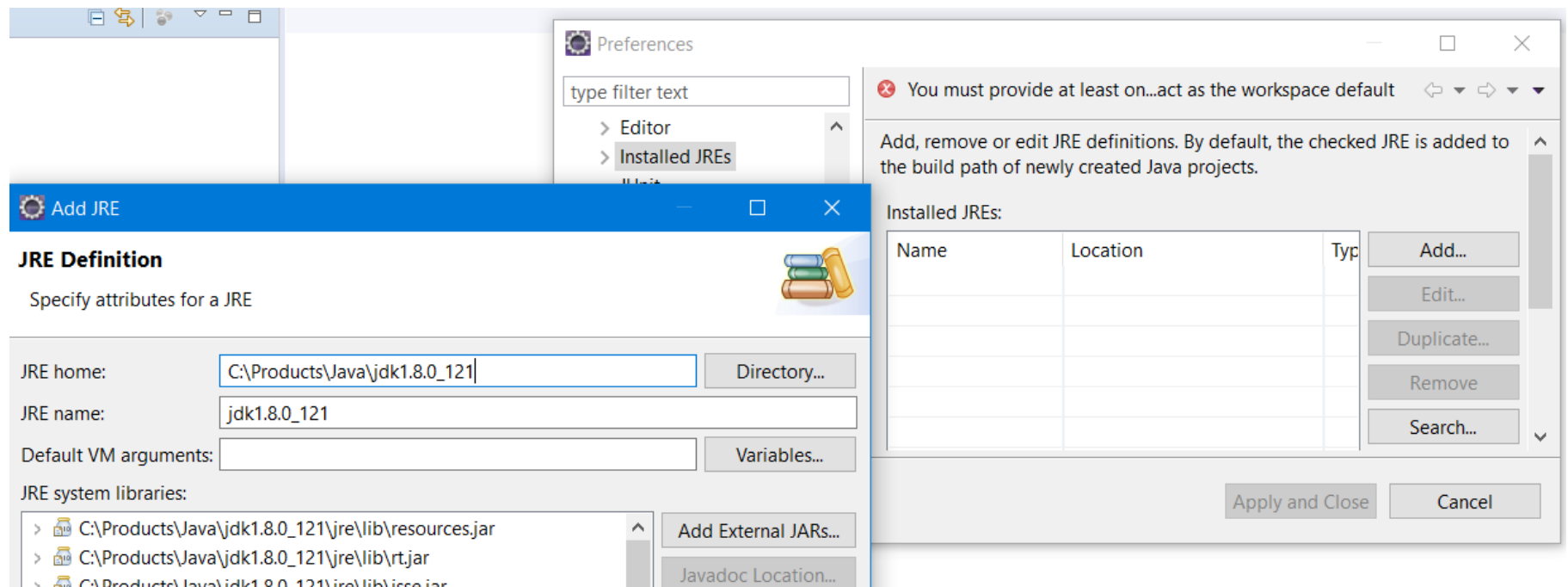
- Ouvrir STS, choisissez le workspace, par exemple **workspace-sts** :



- Et cliquer sur WorkBench.

Configuration IDE STS

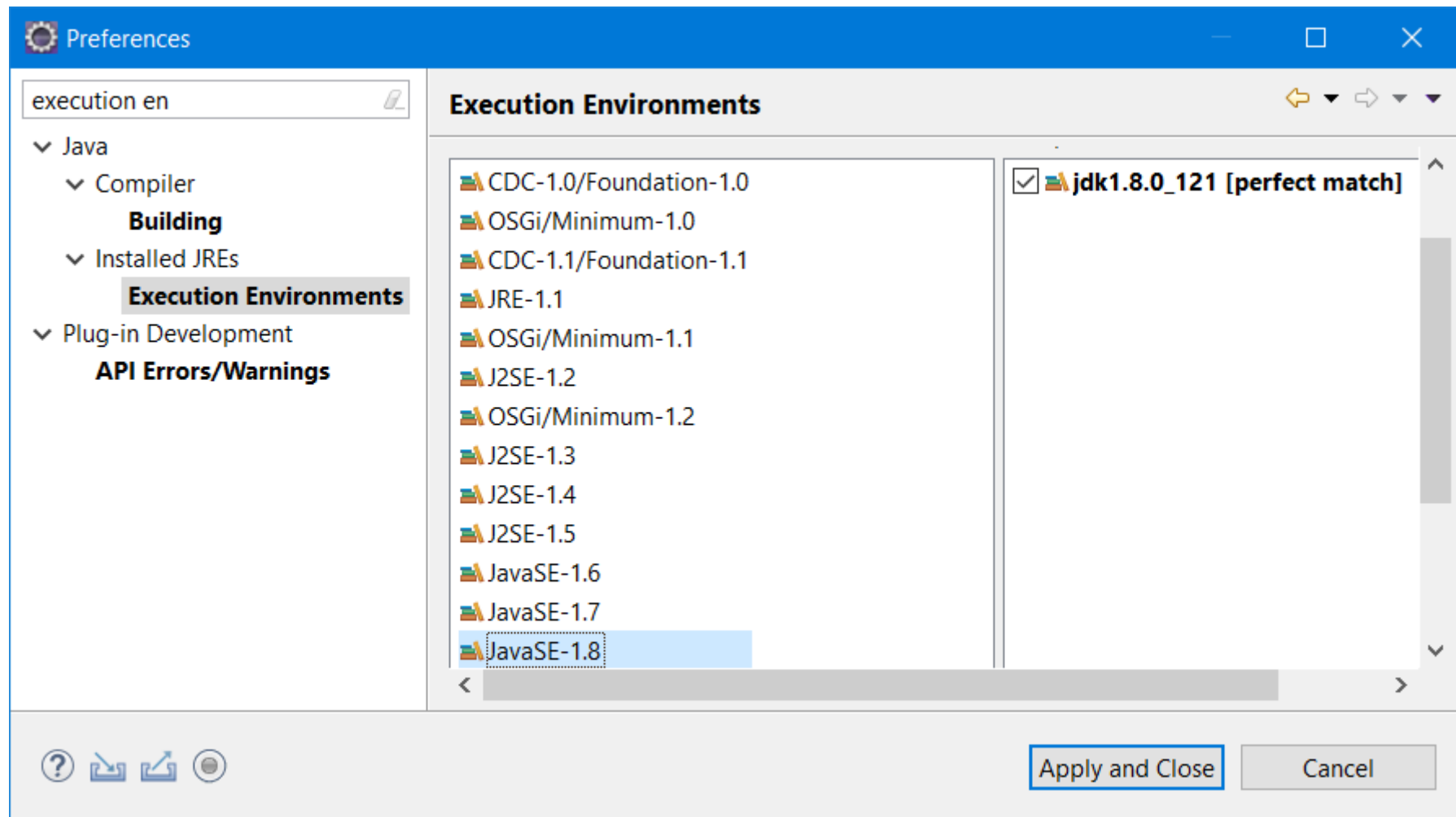
- Faire pointer STS sur JDK 8 :
- Ouvrir window -> preferences -> installed jre -> mettre que la jdk 8(pas la jre). Voir slides suivants
- Supprimer la ligne correspondant à la JRE
- Créer une standard VM en pointant sur la **JDK** :



- Important : Pointer sur la **JDK** et non la JRE (Window – Preference – Installed JRE).

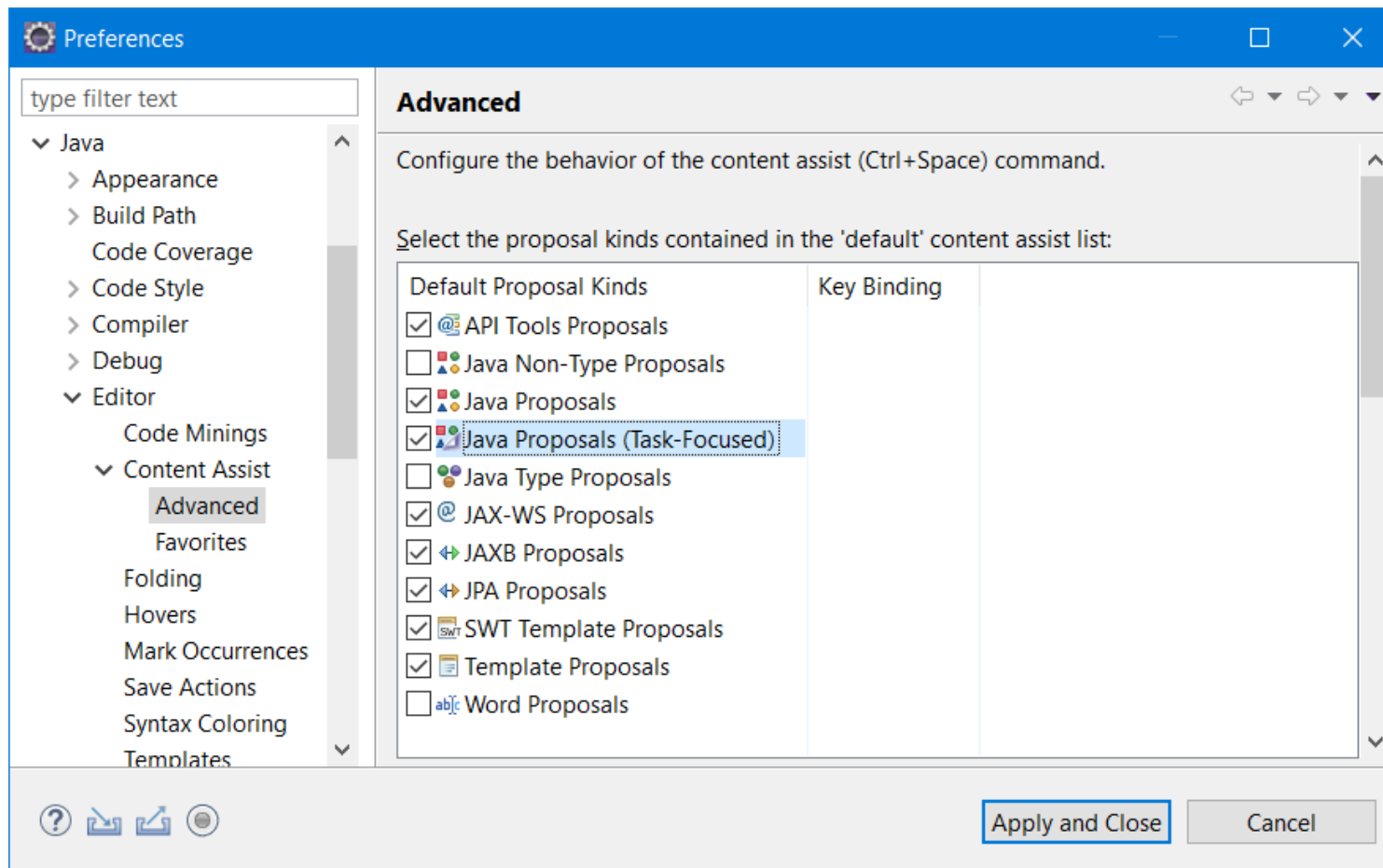
Configuration IDE STS

- Important : Pointer votre Execution Environment vers la JDK.



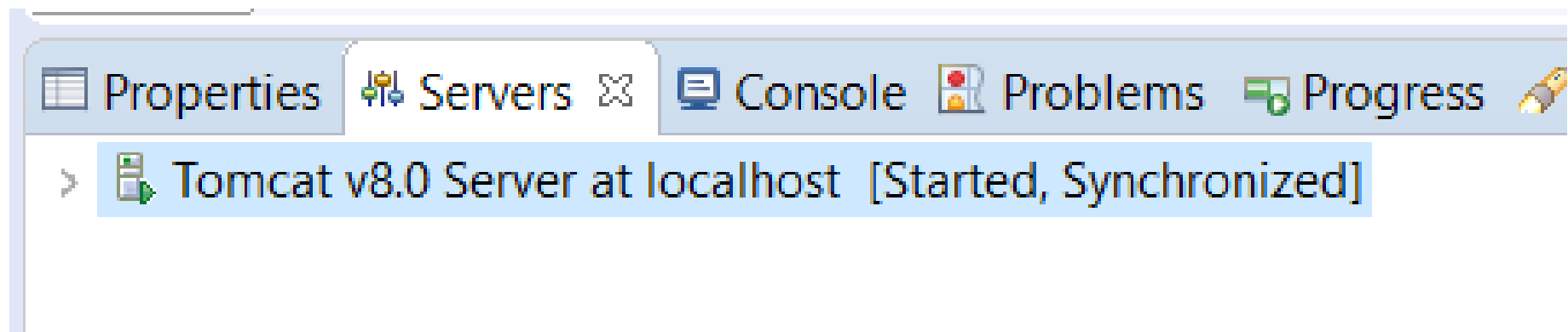
Configuration IDE STS

- Activer l'auto-complétion (**CTRL + ESPACE**) : Windows - Preferences - Java - Content Assist – Advanced :



Configuration Tomcat dans STS

- Ouvrir STS
- Aller à la Vue « **servers** » (window -> show view -> servers).
- new-> server et pointer sur le chemin du Tomcat 8.5.
- Démarrer le serveur, voir les logs (view console).
- Ce serveur sera utilisé pour le cours Spring MVC uniquement.
- Avec Spring Boot, il n'y aura plus besoin d'installer un Tomcat nous-mêmes. Tomcat sera embarqué directement dans notre application.



Prochain Cours

- Finaliser l'Installation de l'Environnement de travail (JDK, STS, Tomcat, MySQL, Maven)
- Comprendre **Maven**
- Créer vos premiers projets avec les outils ci-dessus.

INTRODUCTION SPRING

Si vous avez des questions, n'hésitez pas à nous contacter :

Département Informatique
UP Architectures des Systèmes d'Information

Bureau E204

Introduction SPRING

