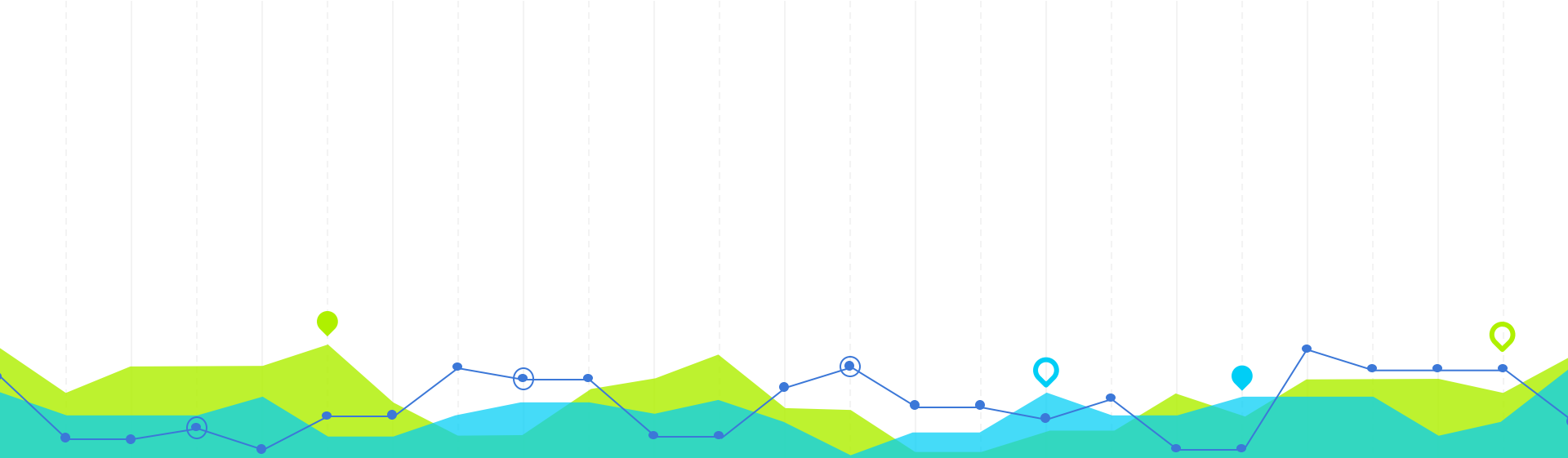




# API REST – Web Service



# Introduction

Les Web services

1

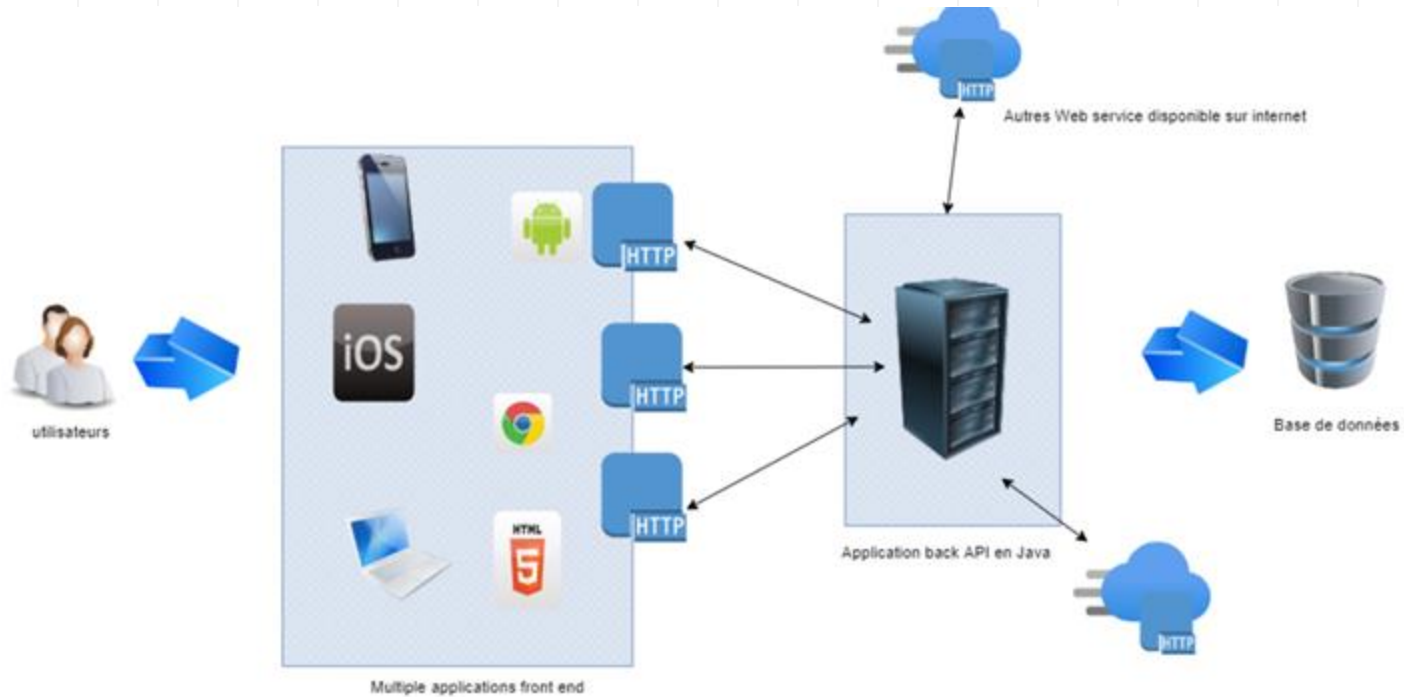
# Introduction

Aujourd'hui la plus part des applications sont architecturé en 3 parties distinct et indépendante

- La partie Front end (Celle qui communique directement avec l'utilisateur)
- La partie Back end (Un web Service qui communique via le protocole HTTP avec le front)
- La base de données.



# Introduction



# Définitions

Un **service Web** est un programme accessible sur internet dont le but est de renvoyer des données, celui qui consomme le web service n'a pas besoin de connaître le langage de programmation du service

**REST** est l'acronyme de "*Representational State Transfer*"

C'est une architecture de **services Web** qui repose sur le protocole **HTTP** : On accède à une **fonctionnalité** (par son URL unique) pour procéder à diverses opérations.

Une **API** est une **Interface Applicative de Programmation** qui permet d'établir des connexions entre plusieurs logiciels pour échanger des données.



# Définitions

Une API peut exposer plusieurs fonctionnalités, chacune possède une URL c'est ce qu'on appelle un **point de terminaison**

Le protocole HTTP nous donne 4 types de requêtes :

- . **GET** pour récupérer des données
- . **POST** pour créer ou modifier une données
- . **PUT** pour sauvegarder
- . **DELETE** pour supprimer une données.



## Exemples

Sur ce site public vous trouverez plusieurs API exploitable comme par exemple celle des jour fériés

<https://api.gouv.fr/documentation/jours-feries>

Cette API dispose des deux points de terminaisons suivants :

GET `//{zone}.json` Liste les jours fériés pour une zone, 20 ans dans le passé et 5 ans dans le futur

GET `//{zone}/{annee}.json` Liste les jours fériés pour une zone, pour une année

# Exemples

Le second point de terminaison renvoie la liste des jours férié en fonction d'une zone géographique et d'une années

GET

`/[zone]/[annee].json`

Liste les jours fériés pour une zone, pour une année

Parameters

Cancel

Name	Description
<b>zone</b> <small>required</small>	Le nom de la zone
string (path)	<div>metropole</div>
<b>annee</b> <small>required</small>	L'année pour les jours fériés
integer (path)	<div>2021</div>

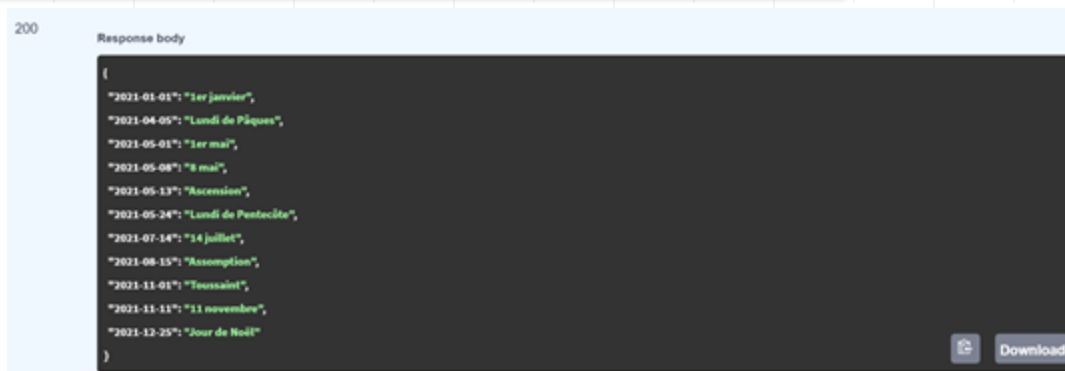
Execute

Clear



## Exemples

On récupère le résultat de l'opération sous forme de JSON



200

Response body

```
{
  "2021-01-01": "1er janvier",
  "2021-04-05": "Lundi de Pâques",
  "2021-05-01": "1er mai",
  "2021-05-08": "8 mai",
  "2021-05-13": "Ascension",
  "2021-05-24": "Lundi de Pentecôte",
  "2021-07-14": "14 juillet",
  "2021-08-15": "Assomption",
  "2021-11-01": "Toussaint",
  "2021-11-11": "11 novembre",
  "2021-12-25": "Jour de Noël"
}
```

Download

```
{ "2021-01-01": "1er janvier", "2021-04-05": "Lundi de Pâques", "2021-05-01": "1er mai", "2021-05-08": "8 mai", "2021-05-13": "Ascension", "2021-05-24": "Lundi de Pentecôte", "2021-07-14": "14 juillet", "2021-08-15": "Assomption", "2021-11-01": "Toussaint", "2021-11-11": "11 novembre", "2021-12-25": "Jour de Noël" }
```

# Exemples

A noter qu'après l'exécution de la requête on peut voir un code de retour 200 ce code indique que l'opération c'est bien passée

Les codes les plus courants sont :

- 200 : succès de la requête ;
- 301 et 302 : redirection, respectivement permanente et temporaire ;
- 401 : utilisateur non authentifié ;
- 403 : accès refusé ;
- 404 : page non trouvée ;
- 500 et 503 : erreur serveur ;
- 504 : le serveur n'a pas répondu.

Sources :

[https://fr.wikipedia.org/wiki/Liste\\_des\\_codes\\_HTTP](https://fr.wikipedia.org/wiki/Liste_des_codes_HTTP)



## Un peu d'outillage

Pour tester vos API ou tout autre Webservice je vous conseille l'outil POSTMAN gratuit et disponible ici

<https://www.postman.com/product/rest-client/>

C'est un outils qui permet entre autre d'envoyer des requêtes HTTP et de visualiser les resultats



# Un peu d'outillage

## Exemple de requêtes postman



On a ici :

- **Adresse de l'API :**  
calendrier.api.gouv.fr
- **Ressources :** jours-feries
- **Paramètres :** zone et années
- *Option :* terminaison par le format

<https://calendrier.api.gouv.fr/jours-feries/metropole/2021.json>





# Creation d'une API

Les Web services

2



## Installation de l'IDE

Pour la suite de ce cours nous allons utiliser l'IDE IntelliJ de la suite JetBrains en version **ultimate**

Normalement vous disposez d'une licence étudiante avec votre adresse mail igs.

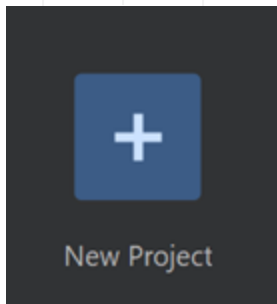
Si ce n'est pas le cas vous pouvez utiliser les 30 jours d'essai gratuit.

<https://www.jetbrains.com/fr-fr/idea/download/#section=windows>

# Création d'un projet

Dans la suite du cours nous allons recréer l'exemple de l'application de gestion des notes du dernier TP sous forme d'API

Une fois l'IDE lancé cliquez sur New Project



# Création d'un projet



Nous allons ensuite nous appuyer sur Spring et plus particulièrement Spring Boot.

**Spring** est un ensemble de framework dont le but est de faciliter le travail du développeur.

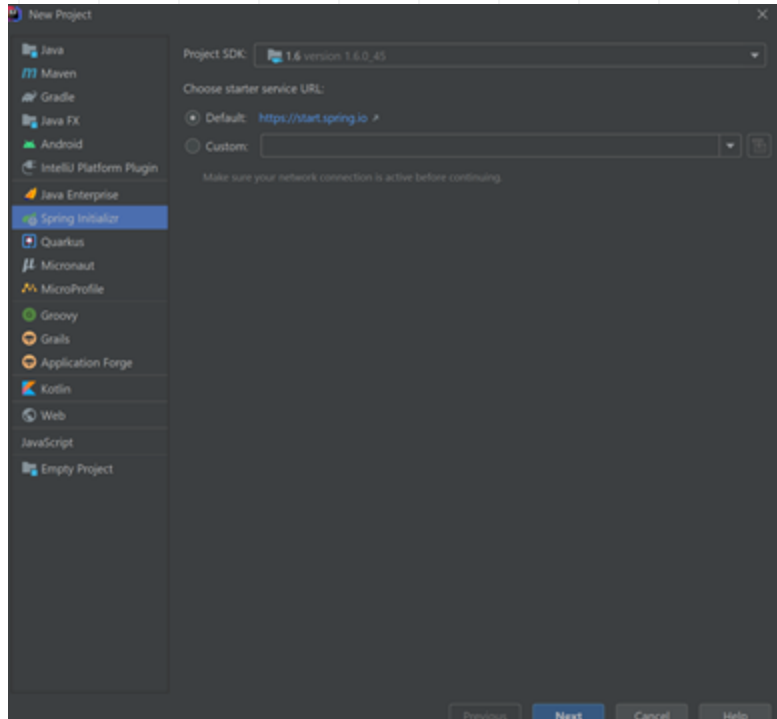
Spring boot met à notre disposition un **cadre de travail et une configuration maven standardisé** pour déployer des applications Web

The Maven logo, featuring the word "Maven" in a bold, black, sans-serif font. The letter "a" is stylized with a colorful, multi-colored vertical bar passing through it. A small trademark symbol (TM) is located to the upper right of the word.

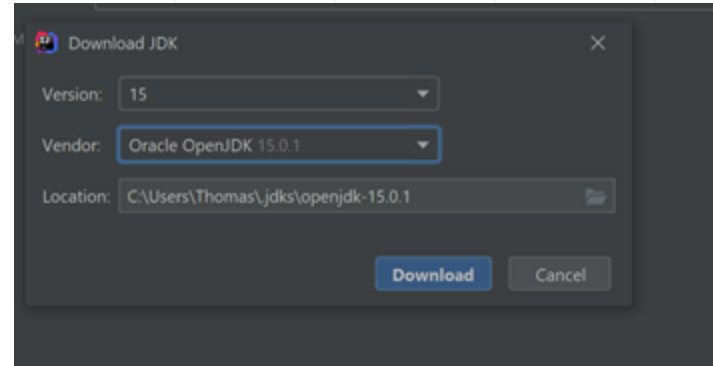
---



# Création d'un projet



Sélectionnez Spring Initializr puis **télécharger le JDK15**



# Création d'un projet



New Project

### Spring Initializr Project Settings

Group:

Artifact:

Type: ☒ Maven ☐ Gradle

Language: ☒ Java ☐ Kotlin ☐ Groovy

Packaging: ☒ Jar ☐ War

Java version:

Version:

Name:

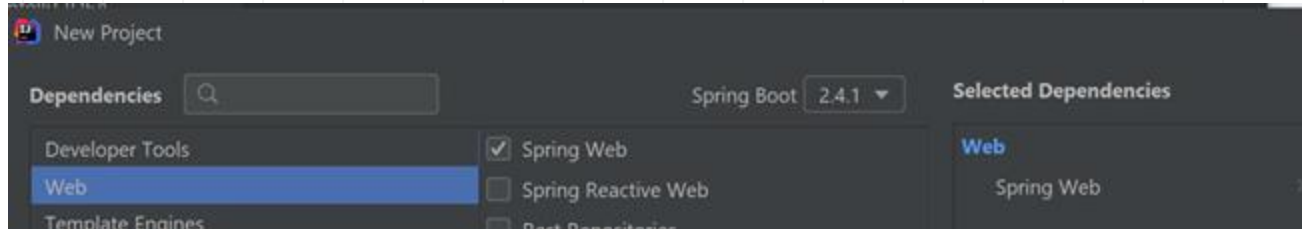
Description:

Package:

# Création d'un projet



Sélectionnez **Web** ☐ **Spring Web**.



# Création d'un projet

```
@SpringBootApplication
public class TpEcoleApplication {

    public static void main(String[] args) { SpringApplication.run(TpEcoleApplication.class, args); }

}
```

L'annotation **@SpringBootApplication** indique qu'on va laisser le framework gérer automatiquement les dépendances de notre projets.

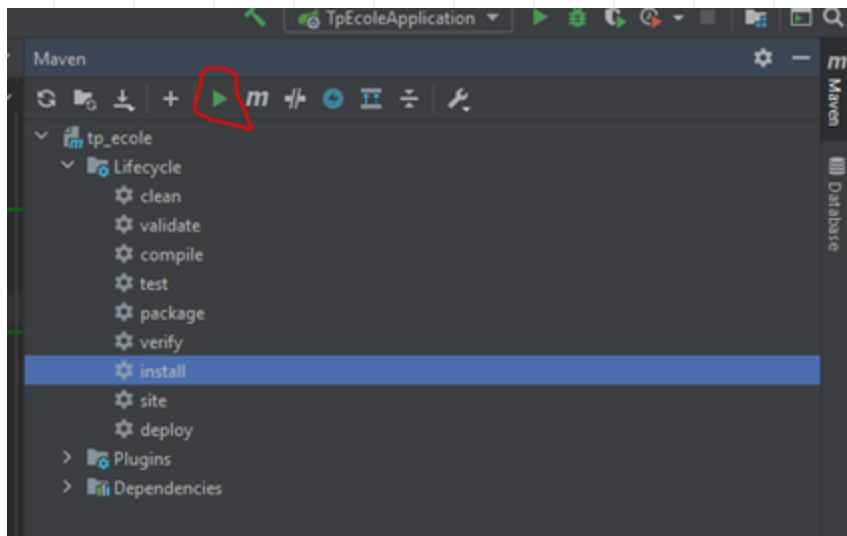
Concrètement dès que l'on va lancer l'application Spring va parcourir le code et effectuée des opérations à chacune des annotations qu'il rencontrera

# Lancement de l'application

IntelliJ nous fournit une interface pour utiliser Maven en fonction du fichier **pom.xml** de notre projet

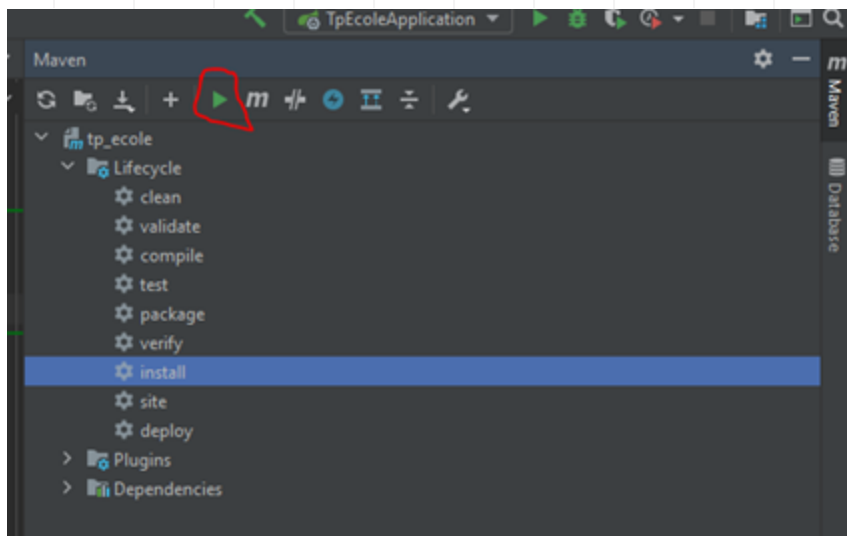
- validate** - valider que le projet est correct et que toutes les informations nécessaires sont disponibles
- compile** - compiler le code source du projet
- test** - tester le code source compilé à l'aide d'un cadre de test unitaire approprié. Ces tests ne devraient pas exiger que le code soit empaqueté ou déployé
- **package** - prendre le code compilé et le conditionner dans son format distribuable, tel qu'un JAR.
- **verify** - effectuer des vérifications sur les résultats des tests d'intégration pour s'assurer que les critères de qualité sont respectés
- **install** - installer le package dans le référentiel local, pour l'utiliser comme dépendance dans d'autres projets localement
- **deploy** - fait dans l'environnement de construction, copie le package final dans le référentiel distant pour le partager avec d'autres développeurs et projets.

# Lancement de l'application



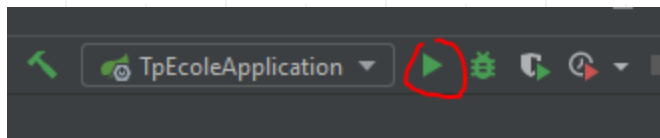
```
[INFO] --- maven-jar-plugin:3.2.0:jar (default-jar) @ tp_ecole ---
[INFO] Building jar: C:\Users\Thomas\IdeaProjects\tp_ecole\target\tp_ecole-0.0.1-SNAPSHOT.jar
[INFO] --- spring-boot-maven-plugin:2.4.1:repackage (repackage) @ tp_ecole ---
[INFO] Replacing main artifact with repackaged archive
[INFO] --- maven-install-plugin:2.5.2:install (default-install) @ tp_ecole ---
[INFO] Installing C:\Users\Thomas\IdeaProjects\tp_ecole\target\tp_ecole-0.0.1-SNAPSHOT.jar to C:\Users\Thomas\.m2\repository\com\tp_ecole\tp_ecole-0.0.1-SNAPSHOT.jar
[INFO] Installing C:\Users\Thomas\IdeaProjects\tp_ecole\pom.xml to C:\Users\Thomas\.m2\repository\com\tp_ecole\tp_ecole-0.0.1-SNAPSHOT.pom
[INFO] BUILD SUCCESS
[INFO] ---
[INFO] Total time: 24.499 s
[INFO] Finished at: 2021-01-03T18:18:29+01:00
[INFO] ---
|
Process finished with exit code 0
```

# Lancement de l'application



```
[INFO] --- maven-jar-plugin:3.2.0:jar (default-jar) @ tp_ecole ---
[INFO] Building jar: C:\Users\Thomas\IdeaProjects\tp_ecole\target\tp_ecole-0.0.1-SNAPSHOT.jar
[INFO] --- spring-boot-maven-plugin:2.4.1:repackage (repackage) @ tp_ecole ---
[INFO] Replacing main artifact with repackaged archive
[INFO] --- maven-install-plugin:2.5.2:install (default-install) @ tp_ecole ---
[INFO] Installing C:\Users\Thomas\IdeaProjects\tp_ecole\target\tp_ecole-0.0.1-SNAPSHOT.jar to C:\Users\Thomas\.m2\repository\com\tp_ecole\tp_ecole-0.0.1-SNAPSHOT.jar
[INFO] Installing C:\Users\Thomas\IdeaProjects\tp_ecole\pom.xml to C:\Users\Thomas\.m2\repository\com\tp_ecole\tp_ecole-0.0.1-SNAPSHOT.pom
[INFO] BUILD SUCCESS
[INFO] ---
[INFO] Total time: 24.499 s
[INFO] Finished at: 2021-01-03T18:18:29+01:00
[INFO] ---
|
Process finished with exit code 0
```

## Lancement de l'application



Une machine virtuel est ensuite lancé sur votre post sur le port 8080

```

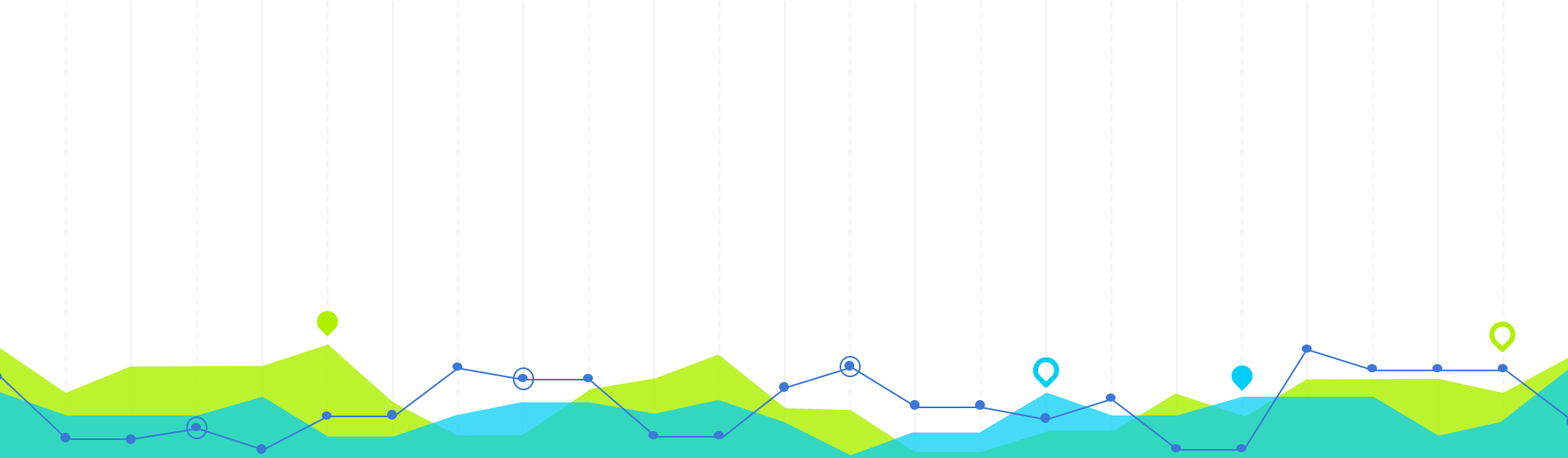
      /\_/\
     (oo)\_____(
        (__)\       )\/\
           ||----w |
           ||     ||

:: Spring Boot ::                (v2.4.1)

2021-01-03 18:40:36.692 INFO 8612 --- [main] com.example.tp_ecole.TpEcoleApplication : Starting TpEcoleApplication using Java 15.0.1 on DESKTOP-SFT76NM with PID 8612 (C:\Users\thumalico\workspace\tp_ecole\target\classes)
2021-01-03 18:40:36.697 INFO 8612 --- [main] com.example.tp_ecole.TpEcoleApplication : No active profile set, falling back to default profiles: default
2021-01-03 18:40:37.000 INFO 8612 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2021-01-03 18:40:37.600 INFO 8612 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2021-01-03 18:40:37.616 INFO 8612 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.41]
2021-01-03 18:40:37.688 INFO 8612 --- [main] w.a.c.e.f.{Tomcat}.LocalhostContext : Initializing Spring embedded WebApplicationContext
2021-01-03 18:40:37.688 INFO 8612 --- [main] w.a.c.f.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 945 ms
2021-01-03 18:40:37.917 INFO 8612 --- [main] w.a.c.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2021-01-03 18:40:38.117 INFO 8612 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2021-01-03 18:40:38.134 INFO 8612 --- [main] com.example.tp_ecole.TpEcoleApplication : Started TpEcoleApplication in 1.838 seconds (JVM running for 4.515)

```





# Les Ressources REST

3

# Les Ressources Rest

```
@RestController
@RequestMapping("/exemple")
public class ExempleRessources {

    @GetMapping("/helloWorld")
    public String hello(){
        return "bonjour ceci est mon premier service";
    }
}
```

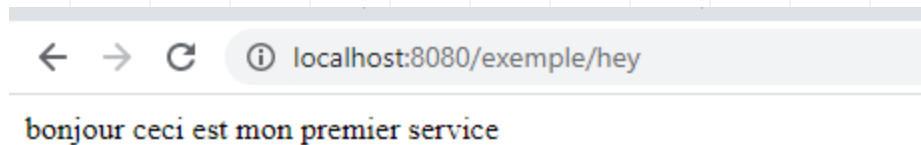
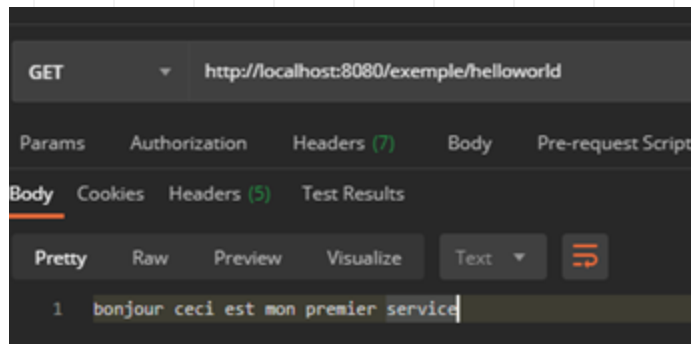
L'annotation **requestMapping** définit l'identifiant de notre ressource ici c'est exemple.

L'annotation **RestController** permet en Spring d'**instancier** l'objet exempleRessource au démarrage de l'application

L'annotation **getMapping** nous permet de définir le point de terminaison hello world



# Les Ressources Rest

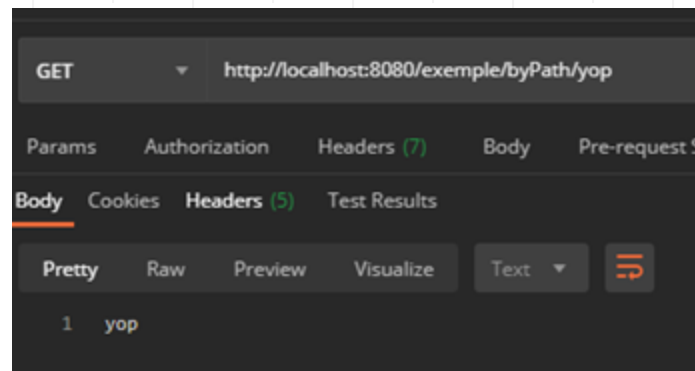


On peut lancer la requête via postMan ou via un navigateur Web

# Les Ressources Rest

Il y a trois manières différentes d'envoyer des informations à un service Web

```
@GetMapping("byPath/{text}")
public String exemple(@PathVariable String text)
{
    return text;
}
```

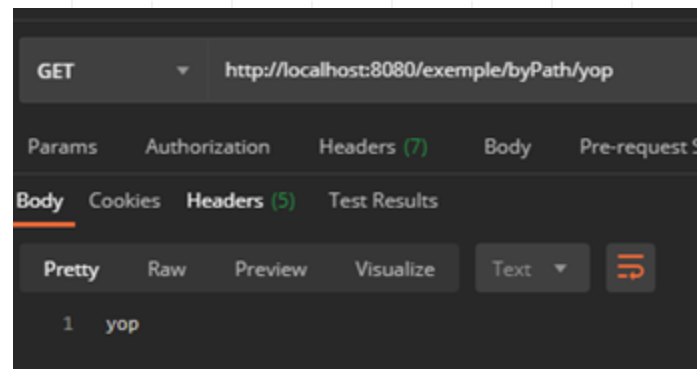


# Les Ressources Rest

Il y a trois manières différentes d'envoyer des informations à un service Web

Directement via l'URI

```
@GetMapping("byPath/{text}")
public String exemple(@PathVariable String text)
{
    return text;
}
```

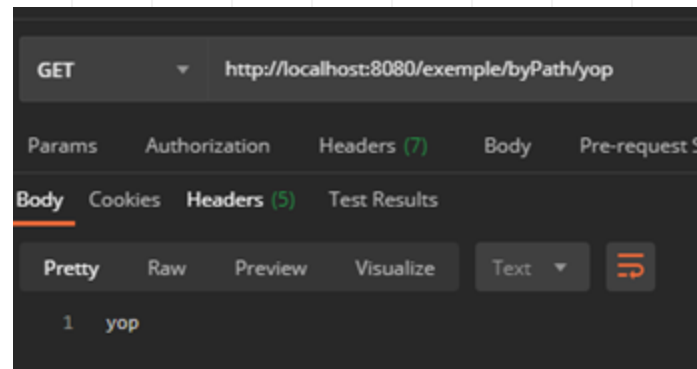


# Les Ressources Rest

Il y a trois manières différentes d'envoyer des informations à un service Web

Directement via l'URI

```
@GetMapping("byPath/{text}")
public String exemple(@PathVariable String text)
{
    return text;
}
```



# Les Ressources Rest

En tant que parametre

```
@GetMapping("hello")
public String exemple(@RequestParam String prenom, @RequestParam Integer age)
{
    return "Bonjour je suis" + prenom + " j'ai " + age + " ans";
}
```

Exemple jour Ferié

GET http://localhost:8080/exemple/hello?prenom=Thomas&age=30

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE
<input checked="" type="checkbox"/> prenom	Thomas
<input checked="" type="checkbox"/> age	30
Key	Value

Body Cookies Headers (5) Test Results

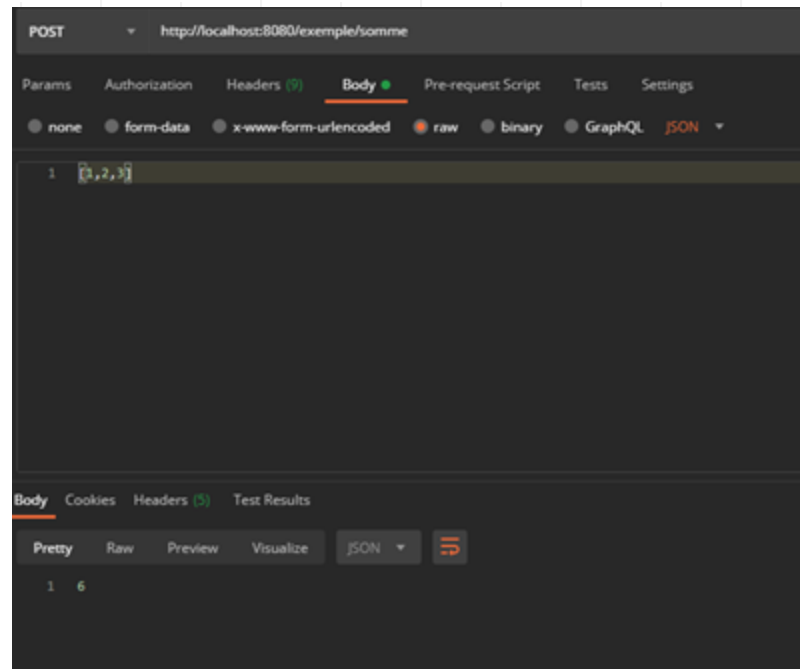
Pretty Raw Preview Visualize Text

```
1 Bonjour je suisThomas j'ai 30 ans
```

# Les Ressources Rest

Envoyer un objet complexe grâce au body

```
@PostMapping("/somme")
public Integer sum(@RequestBody List<Integer> list){
    int result = 0;
    for (int i=0; i<list.size(); i++){
        result += list.get(i);
    }
    return result;
}
```





## Exercices

Créer la ressources Etudiant

Créer le Dto Etudiant

Créer le point de terminaison qui prend un nom un prenom et une date de naissance en parametre, le resultat doit être un json avec le nom le prenom et l'age de l'étudiant

Créer le point de terminaison qui prend un nom un prenom et une liste de notes en paramètres le resultat doit être un json contenant le nom le prenom, la moyenne la note la plus basse ainsi que la notes la plus hautes





# Mise en place de la base de données

JPA/Hibernate

4

# Outils



Pour utiliser une base de données dans nos futures application nous aurons besoin.

- D'un serveur de base de données (**SQL Server Express**)
- Un logiciel de gestion et d'administration de bases de données (**Management Studio**) ou le manager intégré à intellij
- Un ORM Framework pour interagir directement avec notre base de données en java (**JPA/Hibernate**)

# Telechargement SQL Server Express

SQL Server Développeur Edition est une solution Microsoft Gratuite qui permet de mettre en place un serveur de base de données en local sur votre PC.

<https://www.microsoft.com/fr-fr/sql-server/sql-server-downloads>

version Linux

<https://docs.microsoft.com/en-us/sql/linux/sql-server-linux-overview?view=sql-server-linux-ver15>



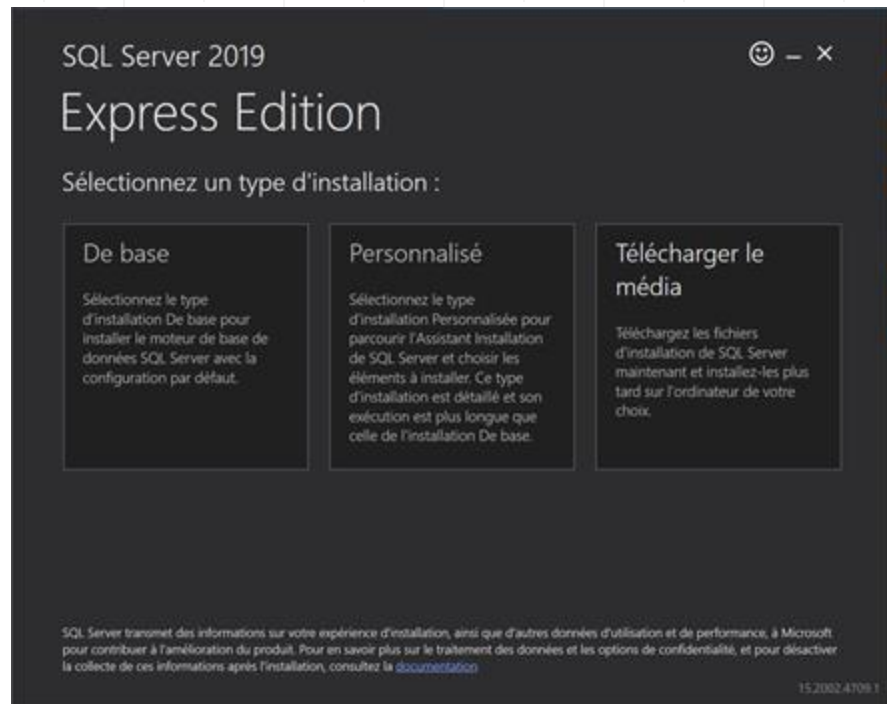
## Express

SQL Server 2019 Express est une édition gratuite de SQL Server, idéale pour le développement et la production d'applications de bureau, d'applications web et de petites applications serveur.

Télécharger maintenant ↓

# Telechargement SQL Server Express

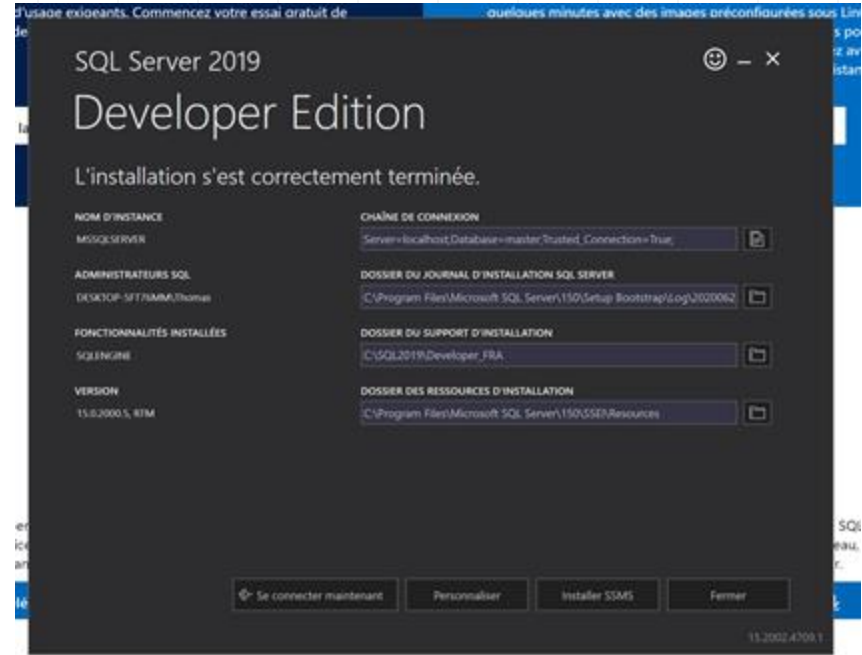
Une fois l'installateur lancé, sélectionnez l'installation de **base**



# Telechargement SQL Server Express

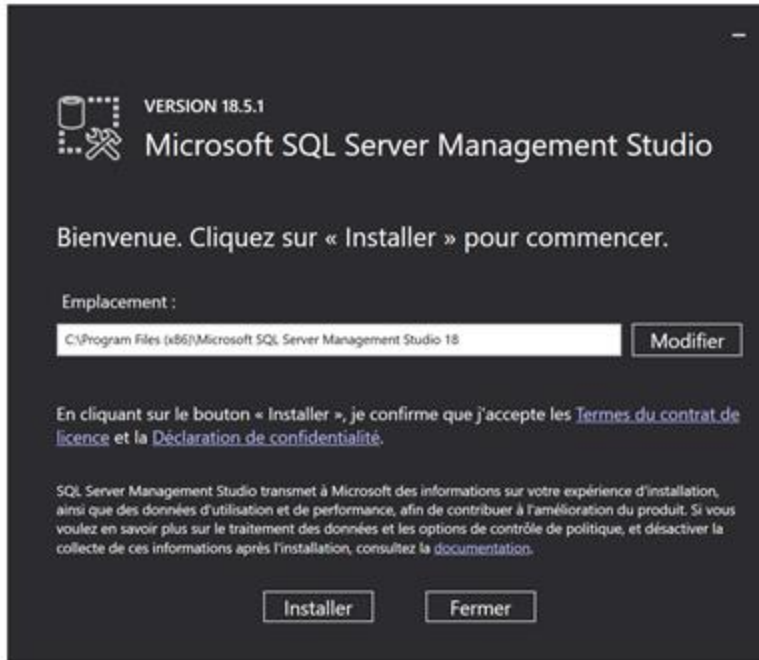
Une fois l'installation terminée, un serveur appelé **locale** est créé sur votre machine. Ainsi qu'une base de donnée par défaut appelé **master**.

Cliquez ensuite sur **Installer SSMS**  
*SQL Server Management Studio*



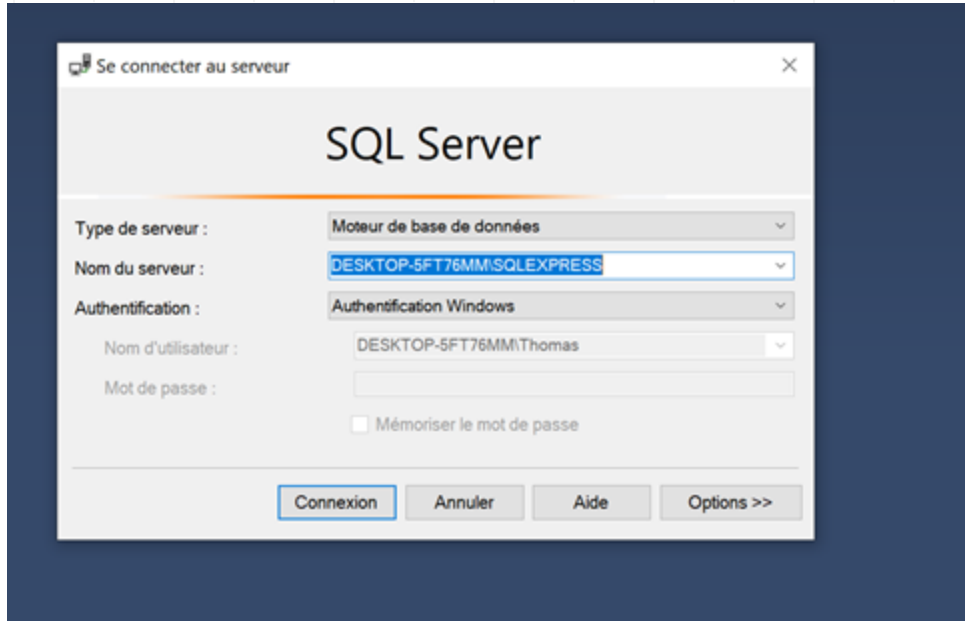
# Telechargement SSMS

Suivre l'installation de base du gestionnaire de base de données



# Telechargement SSMS

Une fois terminé vous pouvez vous connecter à votre serveur de base de données





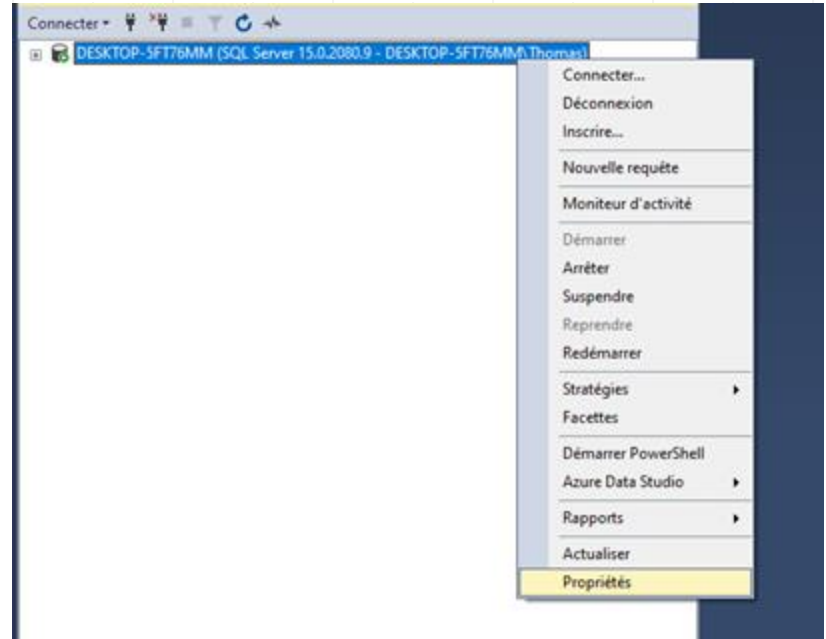
# Configuration du server Local

Du point de vue d'une application java.

Un serveur même installé en local est considéré comme un serveur distant

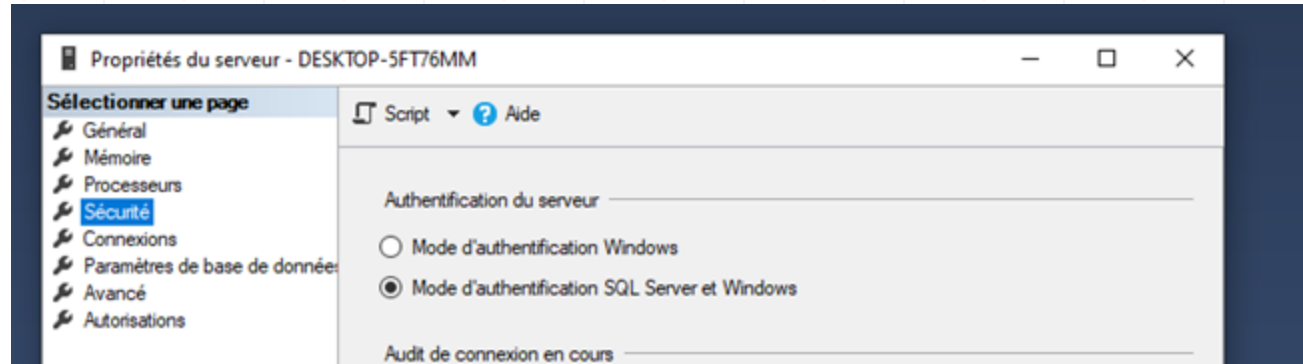
Il faut donc configurer une connexion

Pour cela ouvrez l'onglet propriété du serveur



# Configuration du server Local

Dans l'onglet **sécurité** sélectionnez **Mode d'authentification SQL Server et Windows**

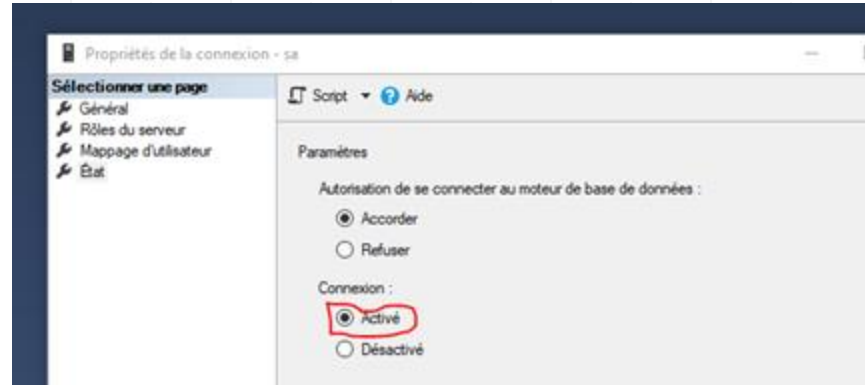


# Configuration du server Local

Deroulez ensuite les repertoire de votre serveur. Puis dans le repertoire **Connexions** double cliquez sur le compte **sa**

Dans l'onglet **Général** définissez un mot de passe.

Dans l'onglet **Etat** activez la connexions



# Configuration

Dans le package ressource se trouve le fichier application.properties c'est ici que l'on va renseigner les informations nécessaires à la connexion

```
spring.datasource.driverClassName=com.microsoft.sqlserver.jdbc.SQLServerDriver
spring.datasource.url=jdbc:sqlserver://localhost:1433;databaseName=ecole
spring.datasource.username=sa
spring.datasource.password=Soleil123
```



# Configuration

Modifier le pom.xml pour ajouter les dépendances suivantes :

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-jdbc</artifactId>
</dependency>
<dependency>
  <groupId>com.microsoft.sqlserver</groupId>
  <artifactId>mssql-jdbc</artifactId>
</dependency>
```



# Creation de la premiere entité

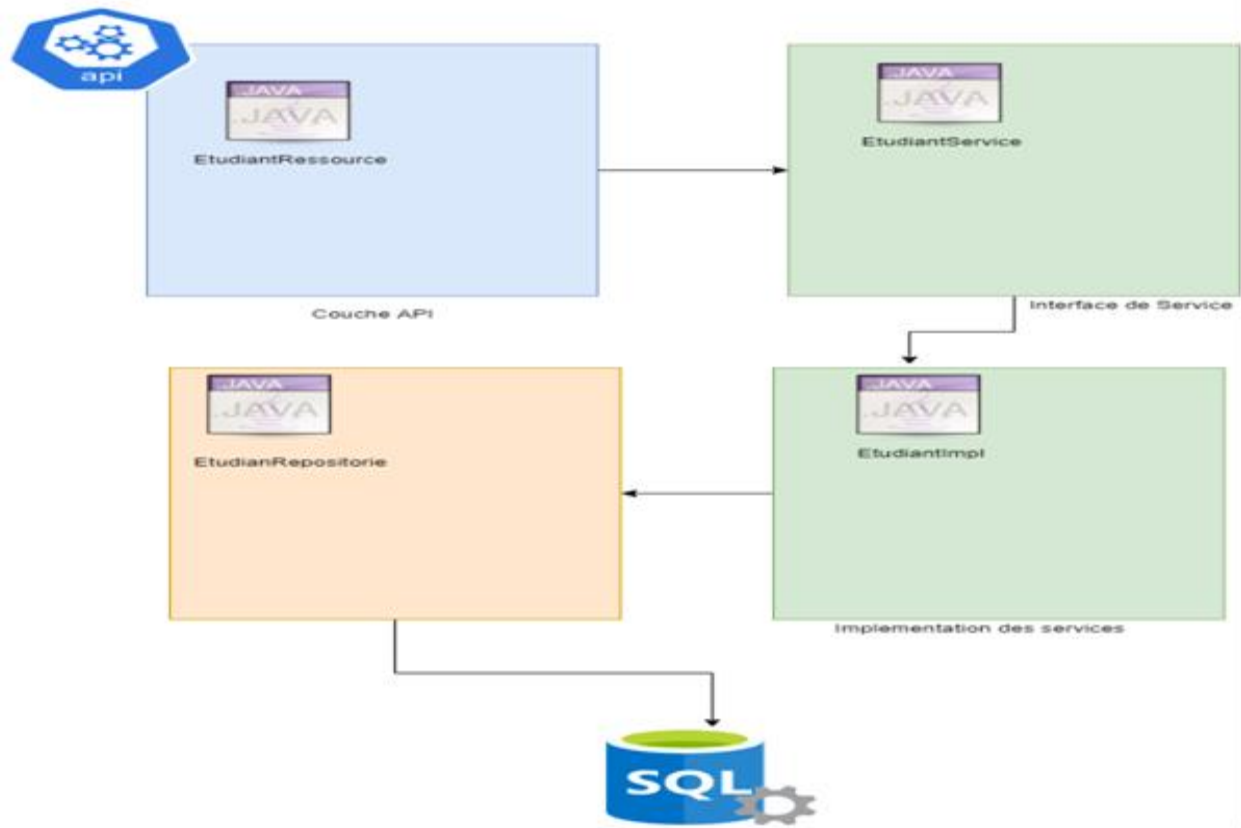
JPA/Hibernate

5

# Hibernate



Hibernate est un framework ORM (Object / Relational Mapping) pour le langage de programmation Java qui concerne la persistance des données. Il s'agit simplement d'une solution de mappage objet-relationnel open-source qui mappe les classes Java aux tables de bases de données dans les bases de données relationnelles et des types de données Java à SQL.





# Entity

```
@Entity
@Table(name = "Etudiant")
public class EtudiantEntity
{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "ID")
    private Integer ID;

    @Column(name = "Prenom")
    private String Prenom;

    @Column(name = "Nom")
    private String Nom;

    @Column(name = "date_naissance")
    private LocalDate dateNaissance;
}
```

# Repository

```
@Repository  
public interface EtudiantRepository extends JpaRepository<EtudiantEntity, Integer> {  
}
```

# Contact

**N'hésitez pas à m'écrire à**  
**[thomas.clamon@gmail.com](mailto:thomas.clamon@gmail.com)**

