

# Atelier Framework Spring-03

## JPA –Partie02

### Objectifs

- **Implémenter le modèle**
  - Créer une entité JPA
  - Créer une base de données Mysql
  - Configurer la base de données avec « **Spring** »

1. Après avoir préparé la structure du projet, passons maintenant à implémenter notre modèle conceptuel :
  - ✓ Créer, dans « **src/main/java** », un package nommé : « **spring.jpa.model** ».
  - ✓ Définir la classe « **Produit** » qui présente les caractéristiques suivantes :
    1. Elle implémente l'interface **Serializable** (**pour être utilisée à distance et envoyée entre différentes machines**).
    2. Elle est annotée par « **@Entity** » pour indiquer qu'il s'agit d'une classe persistante (à traduire en une table dans la base de données)
    3. Elle présente pour l'attribut « **id** » (de type Long) deux annotations :
      - **@Id** : pour spécifier une clé primaire
      - **@GeneratedValue** : pour l'auto incrémentation
    4. Voici le code de la classe « **Produit** » :

```
package spring.jpa.model;

import java.io.Serializable;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.Id;

@Entity
public class Produit implements Serializable {
    @Id
    @GeneratedValue
    private Long id;
    private String designation;
    private double prix;
    private int quantite;
    public Long getId() {
```

```

        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }
    public String getDesignation() {
        return designation;
    }

    public void setDesignation(String designation) {
        this.designation = designation;
    }

    public double getPrix() {
        return prix;
    }

    public void setPrix(double prix) {
        this.prix = prix;
    }

    public int getQuantite() {
        return quantite;
    }

    public void setQuantite(int quantite) {
        this.quantite = quantite;
    }

    public Produit() {
        super();
    }

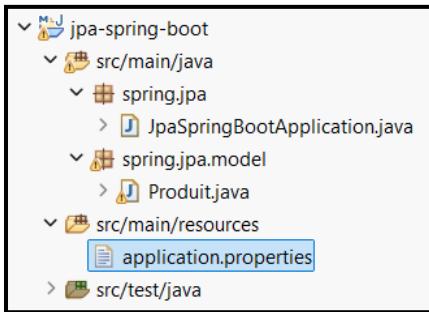
    public Produit(String designation, double prix, int quantite) {
        super();
        this.designation = designation;
        this.prix = prix;
        this.quantite = quantite;
    }

    public Produit(Long id, String designation, double prix, int quantite) {
        super();
        this.id = id;
        this.designation = designation;
        this.prix = prix;
        this.quantite = quantite;
    }
}

```

2. Passons maintenant au mapping (ORM du JPA) avec la base de données :

- ✓ Editer le fichier « **application.properties** » situé sous « **src/main/resources** » pour spécifier les paramètres de connexion à la base de données, ainsi que des paramètres de configuration de Hibernate (framework d'implémentation de JPA):



- ✓ Saisir les instructions suivantes :

```
#database Configuration:
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/spring-jpa
spring.datasource.username=root
spring.datasource.password=

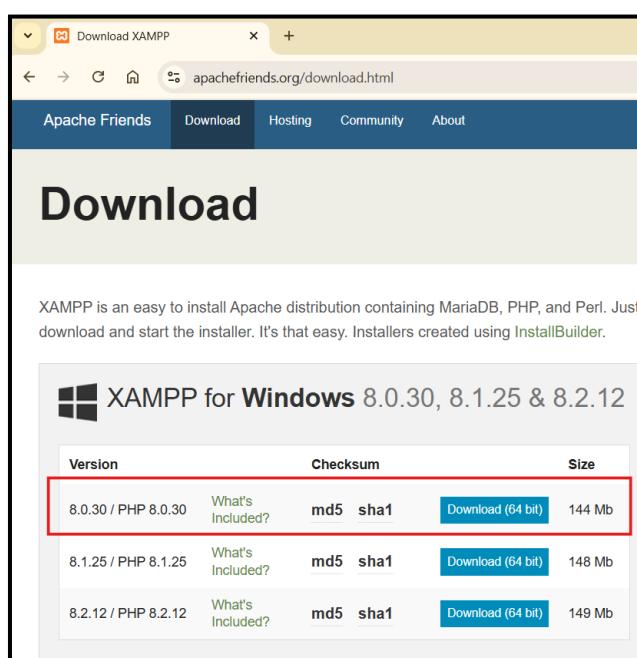
#Hibernate Configuration:
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto= update
```

Nom de la base de données

- ✓ Créer, dans mysql, une base de données nommée : « **spring-jpa** ».
- ✓ Pour se faire, utiliser l'outil **Xampp comme suit** :

- a. Télécharger le logiciel « **xampp** » à partir de l'adresse suivante :

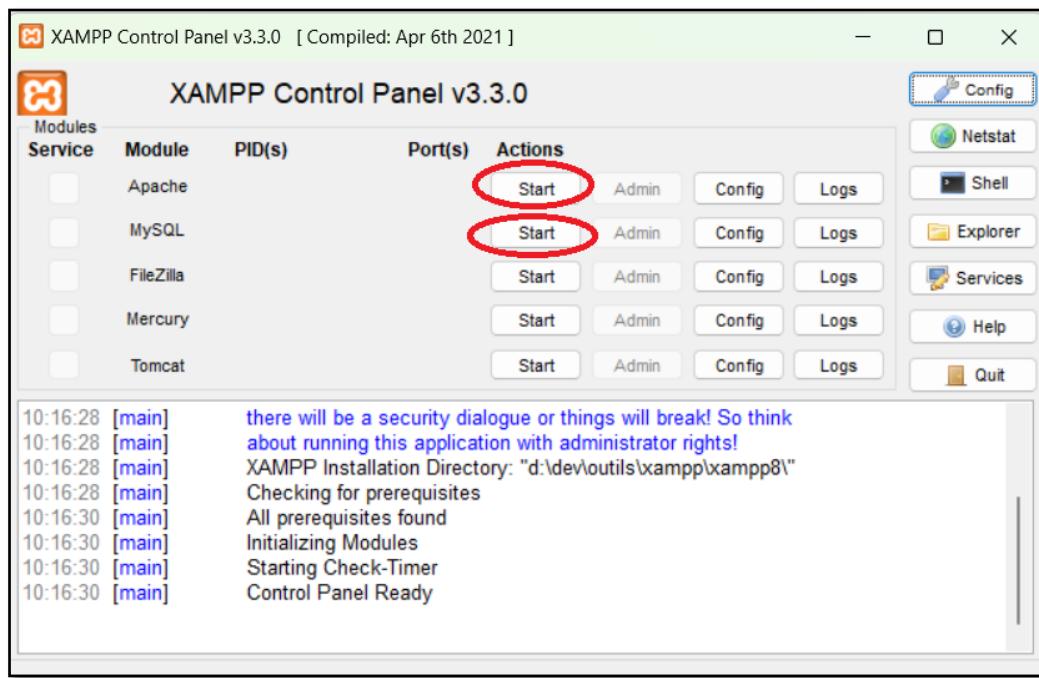
**<https://www.apachefriends.org/download.html>**



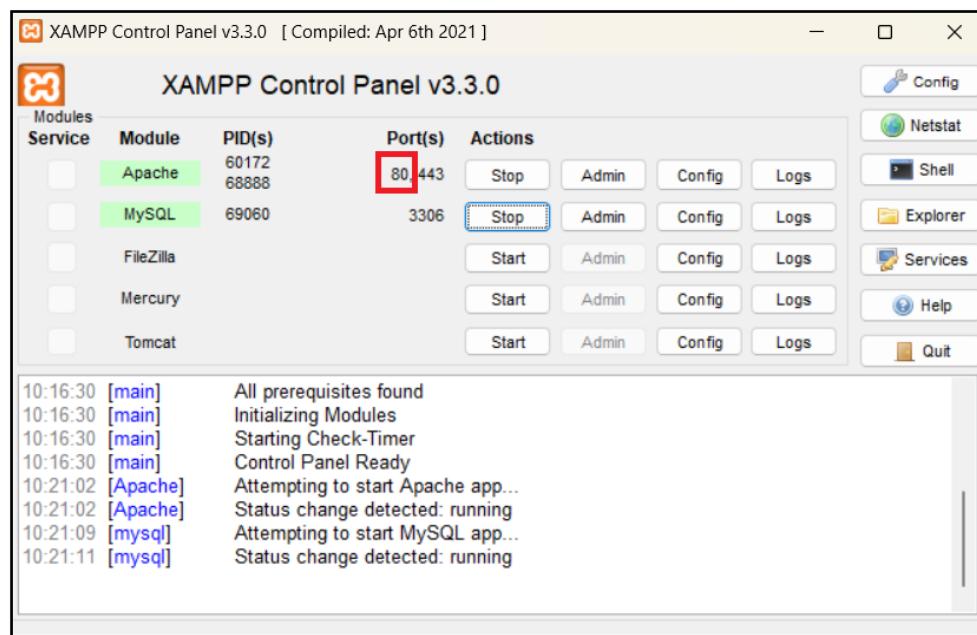
**b.** Double-cliquer sur le fichier téléchargé pour installer «**Xampp**» dans le dossier :

**D:\Dev\Outils\Xampp\Xampp8**

**c.** Ouvrir Xampp, un panneau de contrôle de « **Xampp** » s'affiche pour indiquer les états des différents modules. Par défaut, les services des modules sont arrêtés :

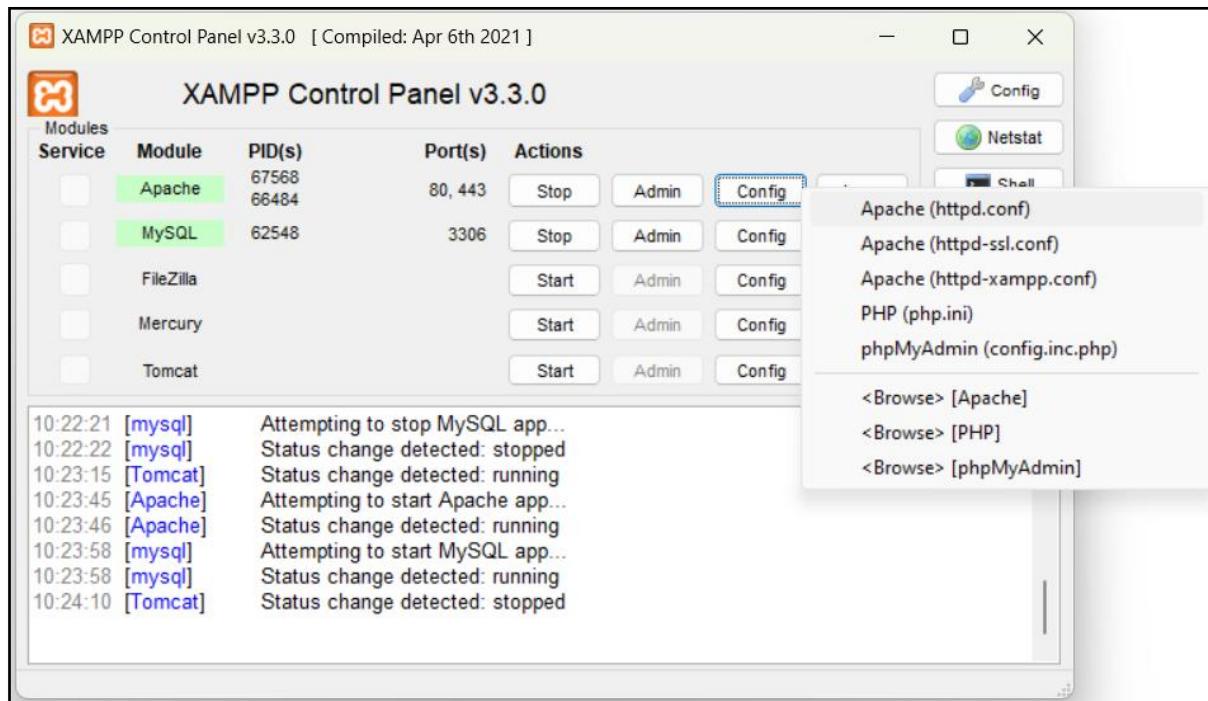


**d.** Démarrer les deux modules «**Apache** » et « **MySQL** ». Remarquer que le module **Apache** est lancé avec le port « **80** » et que le module MySQL est lancé avec le module « **3306** » :



e. Changer le port HTTP à « **81** » (afin de libérer le port « **80** » pour notre serveur web par la suite. Ainsi, suivre les instructions suivantes :

- ❖ Cliquer sur le bouton «**Config**» qui correspond au module «**Apache**»
- ❖ Choisir l'option «**Apache (httpd.conf)**» :

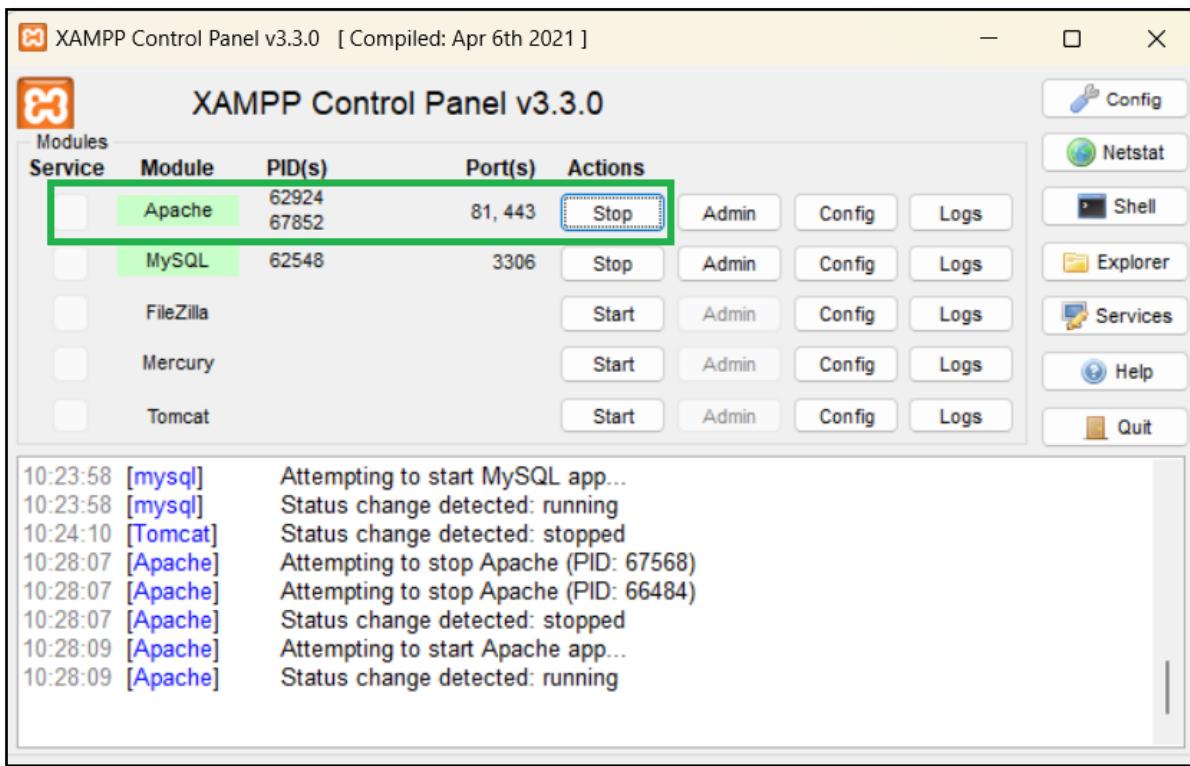


- ❖ Ainsi, ouvrir le fichier « **httpd.conf** » et chercher l'instruction « **Listen 80** » et changer la valeur du port d'écoute du module « **Apache** » à « **81** » et enregistrer.

The screenshot shows a Windows Notepad window titled 'httpd.conf - Bloc-notes'. The file contains the Apache configuration file. A red arrow points to the 'Listen 80' line, which is highlighted in red. The rest of the file content is in black text.

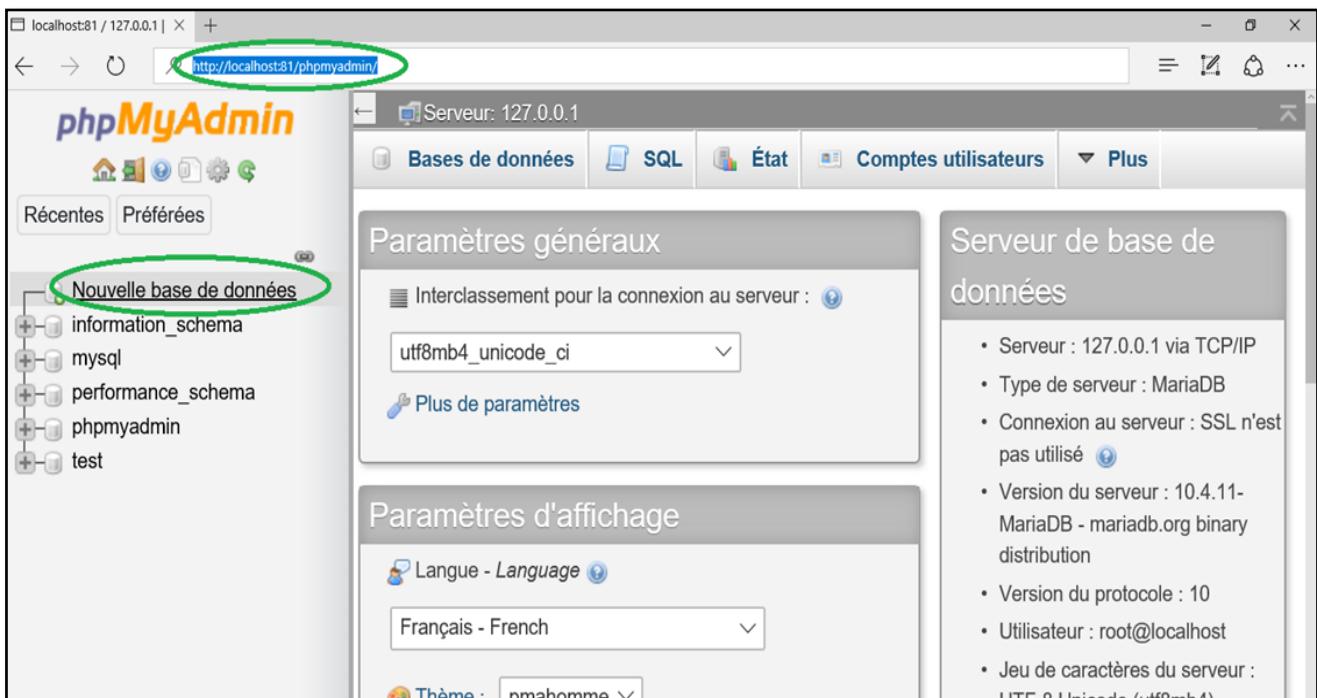
```
# Mutex default:logs
#
# Listen: Allows you to bind Apache to specific IP addresses and/or
# ports, instead of the default. See also the <VirtualHost>
# directive.
#
# Change this to Listen on specific IP addresses as shown below to
# prevent Apache from glomming onto all bound IP addresses.
#
#Listen 12.34.56.78:80
Listen 80
#
# Dynamic Shared Object (DSO) Support
#
# To be able to use the functionality of a module which was built as a DSO you
# have to place corresponding 'LoadModule' lines at this location so the
# directives contained in it are actually available _before_ they are used.
# Statically compiled modules (those listed by 'httpd -l') do not need
# to be loaded here.
#
# Example:
# LoadModule foo_module modules/mod_foo.so
```

- ❖ Arrêter le module « **Apache** » puis le démarrer de nouveau.  
 Remarquer le lancement de « **Apache** » avec le port **81** :

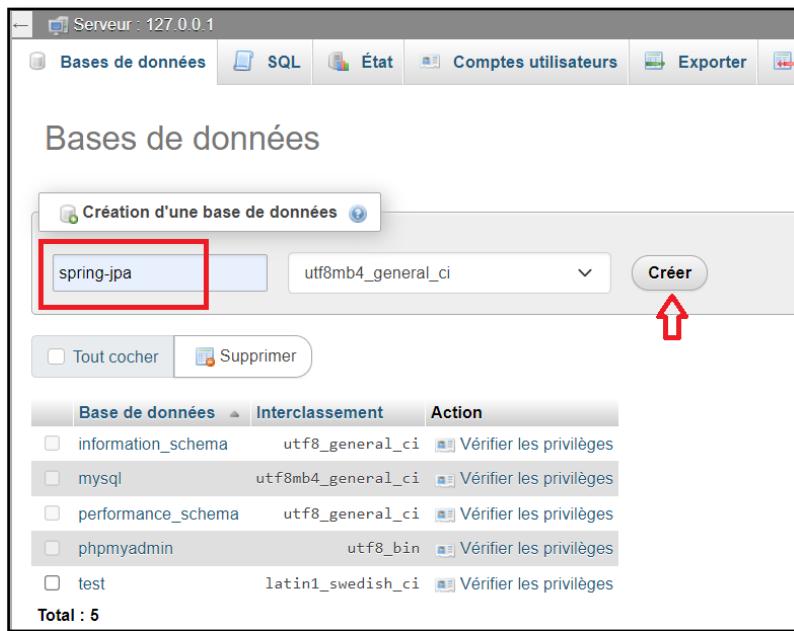


- ❖ Pour lancer l'application « **phpmyadmin** » ouvrir le navigateur web et spécifier l'url suivante :

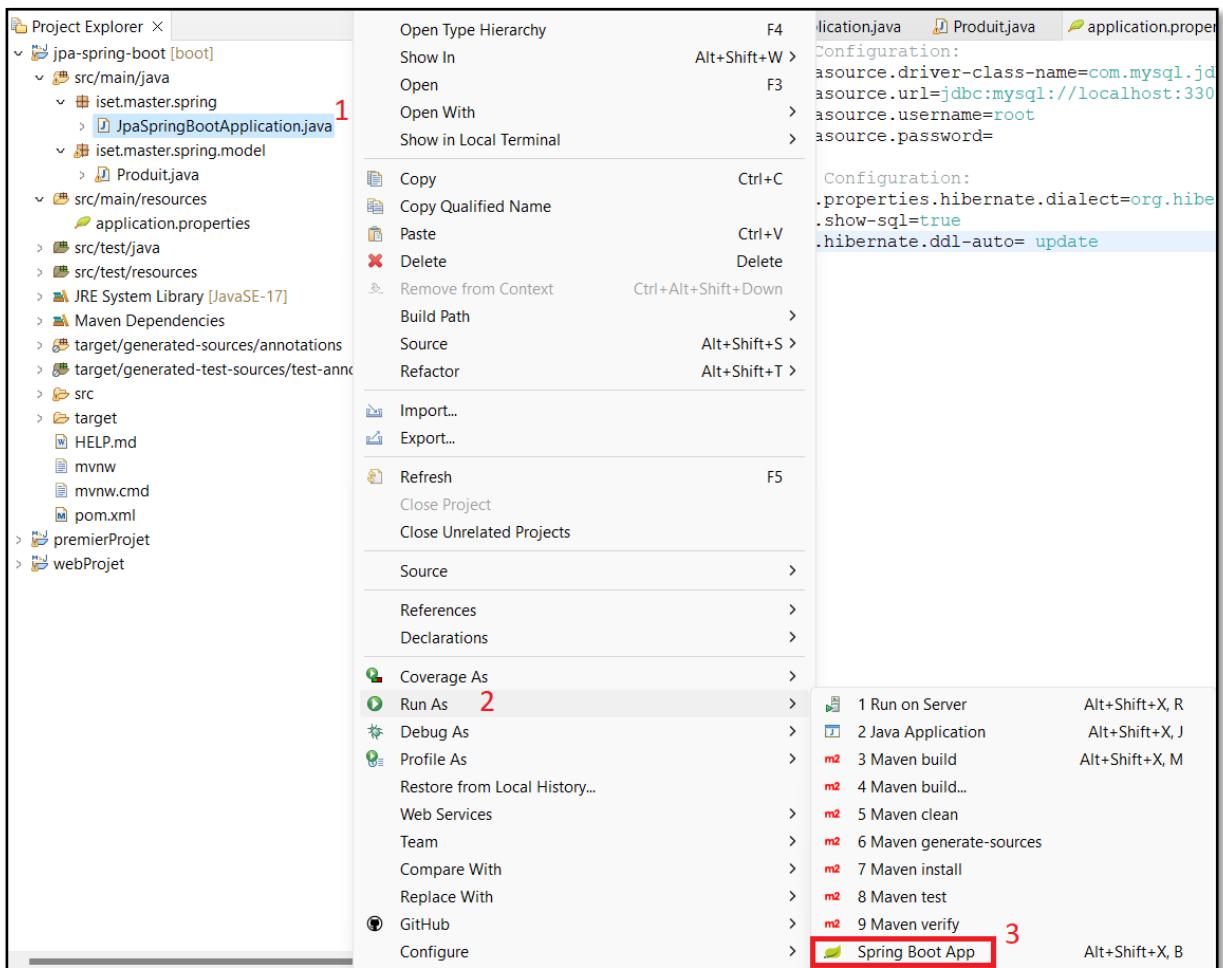
**http://localhost:81/phpmyadmin/**



❖ Créez une base de données nommée «**spring-jpa**» :



- ✓ Maintenant, lancer l'exécution de la classe «**JpaSpringBootApplication**» (classe de démarrage) avec « **Run As/ Spring Boot App** » pour lancer « **Spring Boot**» qui charge le fichier de configuration « **application.properties** » et déclenche le mécanisme ORM :



- ✓ Remarquer la création de la table « **produit** » :

The screenshot shows the phpMyAdmin interface with the following details:

- Serveur :** 127.0.0.1
- Base de données :** spring-jpa
- Structure :** Selected tab.
- Table :** produit (highlighted with a green border).
- Action :** Parcourir, Structure, Rechercher, Insérer, Vider, Supprimer.
- Filter :** Contenant le mot : [empty input field].
- Tables :** produit (selected), produit\_seq.
- Lignes :** 2 tables, Somme.

## NB :

Il est possible de créer automatiquement la base de données à travers l'application gérée par Spring Boot en ajoutant la configuration suivante, dans le fichier « **application.properties** », à la propriété « **spring.datasource.url** » comme suit :

```
application.properties
1 #database Configuration:
2 spring.datasource.driver-class-name=com.mysql.jdbc.Driver
3 spring.datasource.url=jdbc:mysql://localhost:3306/spring-jpa?createDatabaseIfNotExist=true&useSSL=false&serverTimezone=UTC
4 spring.datasource.username=root
5 spring.datasource.password=
6
7 #Hibernate Configuration:
8 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
9 spring.jpa.show-sql=true
10 spring.jpa.hibernate.ddl-auto= update
```

- ✓ Le type de la colonne « **designation** » est « **varchar** » (relatif au type « **String** » de l'attribut de la classe) et sa taille est, par défaut, « **255** » :

The screenshot shows the 'Structure de table' (Table Structure) view for the 'produit' table:

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
1	<b>id</b>	bigint(20)			Non	Aucun(e)			<b>Modifier</b> <b>Supprimer</b> Plus
2	<b>designation</b>	varchar(255)	utf8mb4_general_ci		Oui	NULL			<b>Modifier</b> <b>Supprimer</b> Plus
3	<b>prix</b>	double			Non	Aucun(e)			<b>Modifier</b> <b>Supprimer</b> Plus
4	<b>quantite</b>	int(11)			Non	Aucun(e)			<b>Modifier</b> <b>Supprimer</b> Plus

Pour personnaliser la taille de cette colonne, nous revenons au code de la classe « **Produit** » pour ajouter une annotation « **@Column** » avant la déclaration de l'attribut « **designation** » comme suit (ajouter l'instruction **import** nécessaire) :

```
@Column (length=50)  
private String designation;
```

- ✓ Redémarrer Spring Boot, et remarquer le changement de la structure de la table « produit » par la modification de la taille de la colonne « designation ».



#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
1	<b>id</b> 	bigint(20)			Non	Aucun(e)		
2	<b>designation</b>	varchar(50)	utf8mb4_general_ci		Oui	NULL		
3	<b>prix</b>	double			Non	Aucun(e)		
4	<b>quantite</b>	int(11)			Non	Aucun(e)		