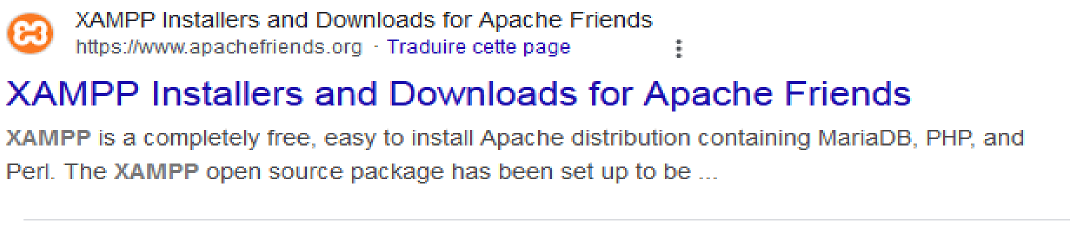
PW 5: Connect The Spring Boot Application To Mysql Database (03h)

To extend the example to use a MySQL database, we'll modify the repository to interact with the database using **Spring Data JPA**. This involves **configuring MySQL as the data source** and **adding JPA annotations to the model class** to map it to a database table.

Step 1: Download and Install XAMPP

1. **Download XAMPP**:
   - Go to the XAMPP website.
   - Click on the **Download** button for the appropriate version (Windows, macOS, or Linux).

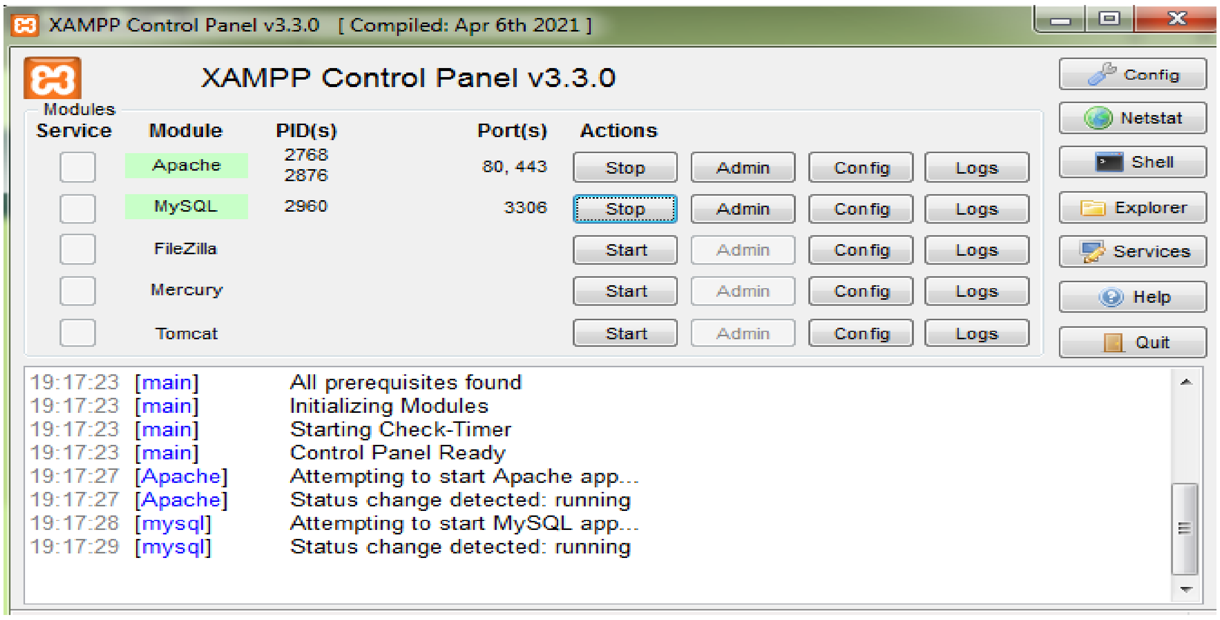


XAMPP Installers and Downloads for Apache Friends
https://www.apachefriends.org · Traduire cette page

XAMPP Installers and Downloads for Apache Friends

XAMPP is a completely free, easy to install Apache distribution containing MariaDB, PHP, and Perl. The **XAMPP** open source package has been set up to be ...

2. **Install XAMPP**:
   - Run the downloaded installer.
   - Follow the installation wizard:
     - Choose components (select Apache and MySQL; PHP is optional).
     - Choose the installation folder (default is fine).
     - Click on **Next** and then **Finish** after the installation is complete.
3. **Start XAMPP**:
   - Open the **XAMPP Control Panel**.
   - Start the **Apache** and **MySQL** services by clicking the **Start** buttons next to each service.
   - Ensure that both services are running (indicated by green lights).


XAMPP Control Panel v3.3.0 [Compiled: Apr 6th 2021]

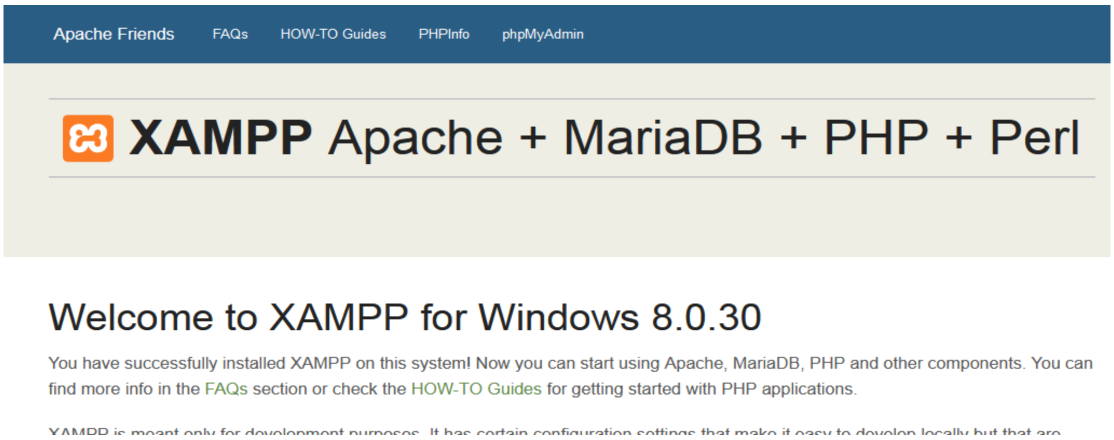
XAMPP Control Panel v3.3.0

Service	Module	PID(s)	Port(s)	Actions			
	Apache	2768 2876	80, 443	Stop	Admin	Config	Logs
	MySQL	2960	3306	Stop	Admin	Config	Logs
	FileZilla			Start	Admin	Config	Logs
	Mercury			Start	Admin	Config	Logs
	Tomcat			Start	Admin	Config	Logs

```
19:17:23  [main]      All prerequisites found
19:17:23  [main]      Initializing Modules
19:17:23  [main]      Starting Check-Timer
19:17:23  [main]      Control Panel Ready
19:17:27  [Apache]    Attempting to start Apache app...
19:17:27  [Apache]    Status change detected: running
19:17:28  [mysql]     Attempting to start MySQL app...
19:17:29  [mysql]     Status change detected: running
```

Step 2: Create a Database

1. **Access phpMyAdmin**:
   - Open a web browser and navigate to http://localhost/phpmyadmin.

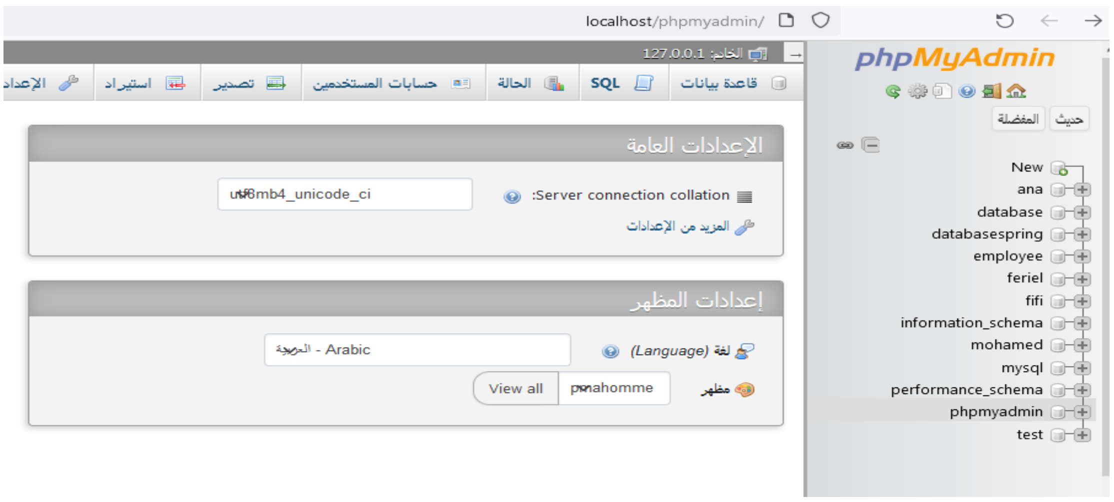


XAMPP Apache + MariaDB + PHP + Perl

Welcome to XAMPP for Windows 8.0.30

You have successfully installed XAMPP on this system! Now you can start using Apache, MariaDB, PHP and other components. You can find more info in the FAQs section or check the HOW-TO Guides for getting started with PHP applications.

2. **Create a New Database and define a user and a password**:
   - Click on the **Databases** tab.
   - In the **Create database** field, enter a name for your database:’’ **student_db’’**
   - Click on **Create**.



Define a user name, and a password of your Db as you like.

3. Create a Table:

- Click on the newly created database (`student_db`).
- Under the **Create table** section, enter a name for the table (**e.g.,** `students`).
- Define the number of columns (e.g., `4`).
- Click on **Go**.

4. Define Table Structure:

- Define columns as follows:
  - **id:** INT, AUTO_INCREMENT, PRIMARY KEY
  - **name:** VARCHAR(100)
  - **email**: VARCHAR(100)
  - **age:** INT
- Click **Save** to create the table.

Step 3: implementation

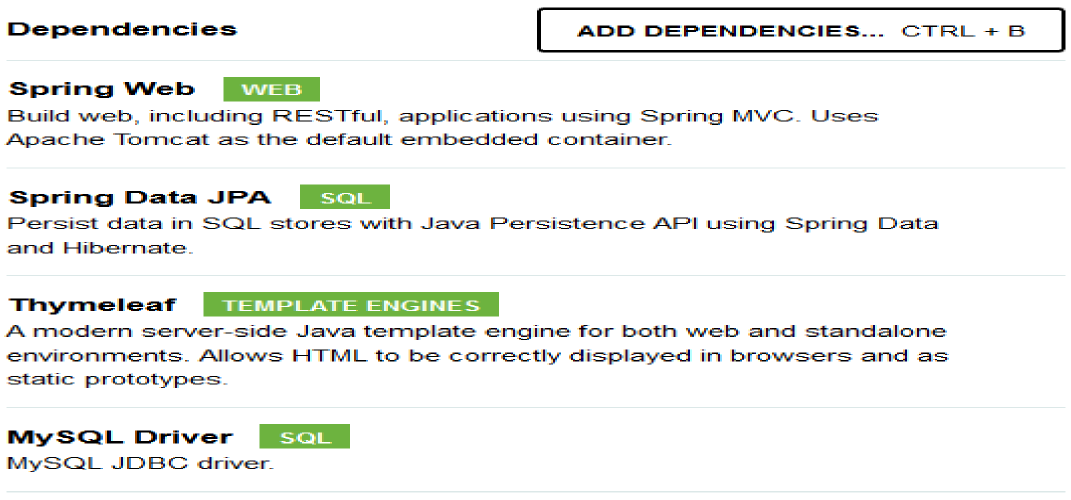
The project's structure:

```
src/
└── main/
    ├── java/
    │   └── com.example.demo/
    │       ├── controller/
    │       │   └── StudentController.java
    │       ├── model/
    │       │   └── Student.java
    │       ├── service/
    │       │   └── StudentService.java
    │       └── repository/
    │           └── StudentRepository.java
    └── resources/
        ├── templates/
        │   └── student-form.html
        │       Student-list.html
        └── application.properties
```

1. Dependencies (pom.xml)

Add the necessary dependencies in the `pom.xml` for Spring Data JPA and MySQL.

- **web**
- **Thymeleaf**
- **Data**
- **Mysql driver**



Dependencies ADD DEPENDENCIES... CTRL + B

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

Thymeleaf TEMPLATE ENGINES
A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.

MySQL Driver SQL
MySQL JDBC driver.

2. MySQL Configuration (application.properties)

Configure the connection to your MySQL database in `application.properties`.

```
spring.datasource.url=jdbc:mysql://localhost:3306/student-db
spring.datasource.username=name
spring.datasource.password=motdepasse
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

3. Model Class (Student.java)

Add JPA annotations to map the `Student` class to a database table.

```
@Entity
@Table(name = "students")
public class Student {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
}
```

4. Repository Interface (StudentRepository.java)

Spring Data JPA provides a repository interface that automatically handles basic CRUD operations:

```
public interface StudentRepository extends JpaRepository<Student,
Long> {
}
```

5. Service Class (StudentService.java)

The service class will interact with the repository to retrieve and store data in the database.

```java
@Service
public class StudentService {

    @Autowired
    private StudentRepository studentRepository;

    public List<Student> getAllStudents() {
        return studentRepository.findAll();
    }

    public void saveStudent(Student student) {
        studentRepository.save(student);
    }

    public Student getStudentById(Long id) {
        return studentRepository.findById(id).orElse(null);
    }

    public void deleteStudent(Long id) {
        studentRepository.deleteById(id);
    }
}
```

6. Controller Class (StudentController.java)

The controller remains mostly the same, but now when adding or retrieving students, they will be stored in the MySQL database.

```java
@Controller
@RequestMapping("/…")
public class StudentController {
    @Autowired
    private StudentService studentService;

    @GetMapping
    public String listStudents(Model model) {
        List<Student> students = studentService.getAllStudents();
        model.addAttribute("students", students);
        return "student-list";
    }

    @GetMapping("/add")
    public String showAddForm(Model model) {
        model.addAttribute("student", new Student());
        return "student-form";
    }

    @PostMapping("/save")
    public String saveStudent(@ModelAttribute("student") Student
student) {
        studentService.saveStudent(student);
        return "redirect:/students";
    }

    @GetMapping("/edit/{id}")
    public String showEditForm(@PathVariable Long id, Model
model) {
        Student student = studentService.getStudentById(id);
        model.addAttribute("student", student);
        return "student-form";
    }

    @GetMapping("/delete/{id}")
    public String deleteStudent(@PathVariable Long id) {
        studentService.deleteStudent(id);
        return "redirect:/students";
    }
}
```

7. View Template (students.html)

Form-list.html is for the form, it containts 03 textfields for :Name, Age, Email and a submit button

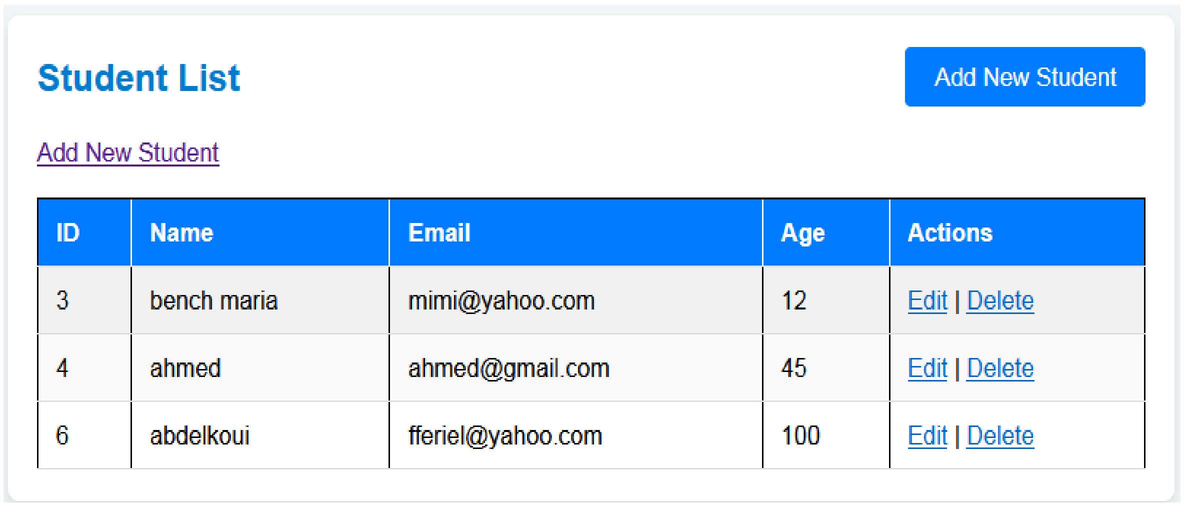
```
<h1 th:text="${student.id != null} ? 'Edit Student':
'Add Student'"></h1>

<form th:action="@{/students/save}" th:object="${student}"
method="post">

  <input type="hidden" th:field="*{id}"/>

  <label>Name:</label>
  <input type="text" th:field="*{name}" required="required"/>
.
      .//also textfield for :  email,  age
.

.
.        <button type="submit">Submit</button>
    </form>
```

Student-list.html

```
        <tbody>
            <tr th:each="student : ${students}">
                <td th:text="${student.id}"></td>
                <td th:text="${student.name}"></td>
                <td th:text="${student.email}"></td>
                <td th:text="${student.age}"></td>
                <td>
                    <a
th:href="@{/students/edit/{id}(id=${student.id})}">Edit</a> |
                    <a
th:href="@{/students/delete/{id}(id=${student.id})}"
```

Run the application and view the results.

The results of executing our application should be:

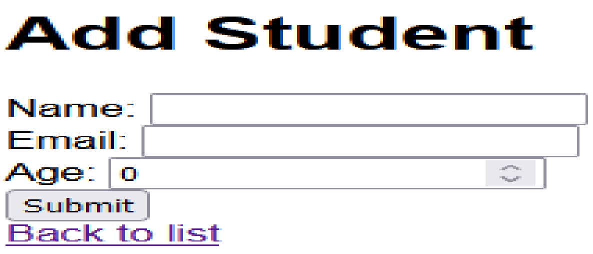


Student List

Add New Student

Add New Student

ID	Name	Email	Age	Actions
3	bench maria	mimi@yahoo.com	12	Edit | Delete
4	ahmed	ahmed@gmail.com	45	Edit | Delete
6	abdelkoui	fferiel@yahoo.com	100	Edit | Delete

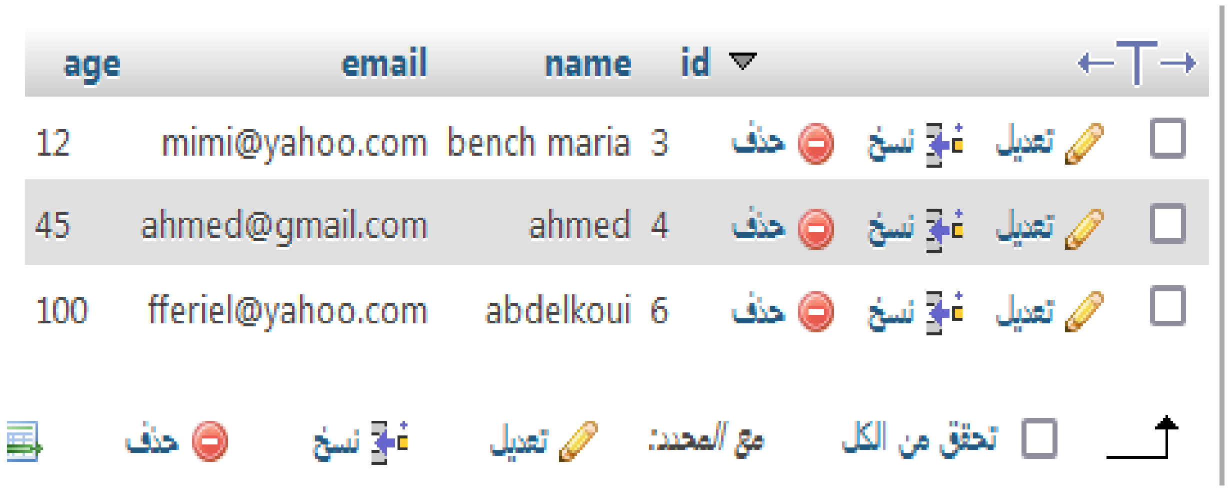


Add Student

Name:	
Email:	
Age:	0

Submit

Back to list



←T→				id	name	email	age
☐	تعديل	نسخ	حذف	3	bench maria	mimi@yahoo.com	12
☐	تعديل	نسخ	حذف	4	ahmed	ahmed@gmail.com	45
☐	تعديل	نسخ	حذف	6	abdelkoui	fferiel@yahoo.com	100

☐ تحقق من الكل مع المحدد: تعديل نسخ حذف