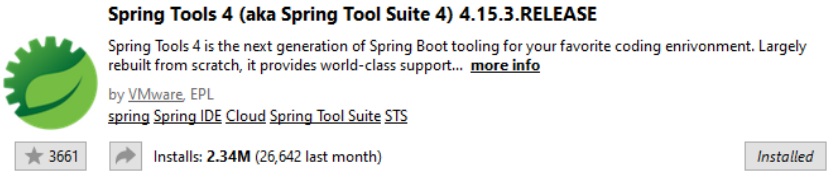
Tài liệu thực hành 3.1

Hướng dẫn 1: Tạo Rest API bằng Java Spring Boot

Công cụ thực hành

- Eclipse IDE [Download]

- Spring Tool Suite 4 (Tại thanh công cụ của Eclipse IDE > Help > Eclipse Marketplace > Gõ "Spring Tools" > Install)



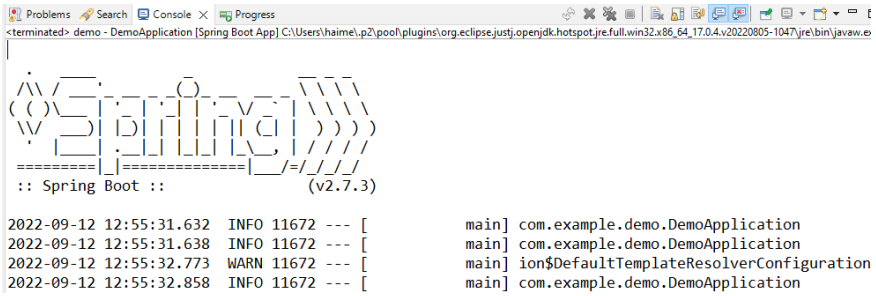
- MySQL Community Server, MySQL Workbench [Download]

- Postman Desktop [Download]

Tạo project

https://start.spring.io/	Spring Tool Suite
B1. Truy cập https://start.spring.io/ B2. Tùy chọn loại project (Maven), ngôn ngữ (Java), phiên bản Spring Boot và project metadata. Thêm các dependency bao gồm: Spring Web, Spring Data JPA, MySQL Driver, Spring Boot DevTools. B3. Ấn GENERATE và giải nén file vừa tải về. Tại giao diện Eclipse, chọn File > Open projects from file system > Chọn thư mục vừa giải nén.	B1. Trên thanh công cụ của Eclipse, chọn File > New > Spring Starter Project. B2. Tại biểu mẫu nhập thông tin project, chúng ta tùy chọn nhập trường name, group và artifact, sau đó nhấn Next. B3. Thêm các dependency như "JPA", "MySQL", "Web", … vào project. Sau đó nhấn Finish.

B4. Chuột phải vào project, chọn Run As > Spring Boot App và xem kết quả ở màn hình Console.



Kết nối database

B1. Khởi động MySQL Workbench và chọn kết nối tới local instance.

B2. Chạy câu lệnh SQL query sau để tạo database, tạo bảng và thêm dữ liệu vào bảng.

```
CREATE DATABASE TEST
USE TEST
CREATE TABLE Department (
    departmentID int(10) not null auto_increment PRIMARY key,
    name nvarchar(100)
)
INSERT INTO Department VALUES (1, “CNPM”)
INSERT INTO Department VALUES (2, “KTMT”)
INSERT INTO Department VALUES (3, “KHMT”)
```

B3. Tại thư mục src/main/resources, file application.properties, chúng ta khai báo thông tin về database cho project.

application.properties

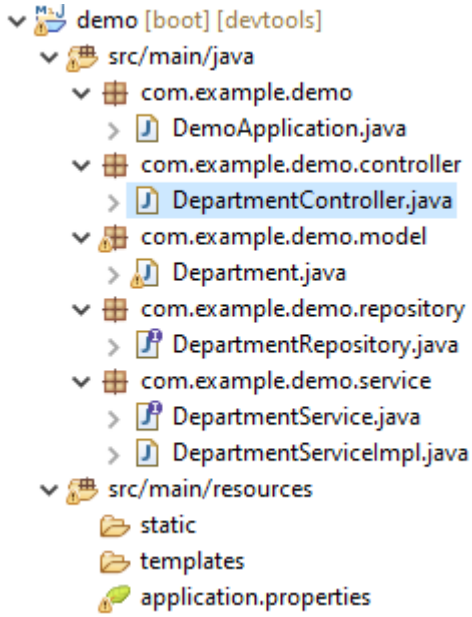
```
spring.datasource.url=jdbc:mysql://localhost:3306/test
spring.datasource.username=root
spring.datasource.password=root
spring.datasource.dbcp.test-while-idle=true
spring.datasource.dbcp.validation-query=SELECT 1
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.hibernate.naming.strategy=org.hibernate.cfg.ImprovedNamingStrategy
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect
```

Lưu ý khai báo username và password kết nối đến database:

```
spring.datasource.username=root
spring.datasource.password=root
```

Tạo project Spring MVC [Code mẫu [3]]

Tạo lần lượt các package Model, Repository, Service và Controller và thêm các class theo như thư mục:



model/Department.java

```
package com.example.demo.model;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.Table;
import org.springframework.stereotype.Component;
@Entity
@Table(name = "Department")
@Component
public class Department {
	@Id
	@GeneratedValue
	@Column(name = "departmentID")
	private Long departmentID;
	@Column(name = "name")
	private String name;
```

```
    public Department() {
        this.name = "";
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public long getId() {
        return this.departmentID;
    }

    public void setId(long id) {
        this.departmentID = id;
    }
}
```

repository/DepartmentRepository.java

```
package com.example.demo.repository;
import org.springframework.data.repository.CrudRepository;
import com.example.demo.model.Department;
public interface DepartmentRepository extends CrudRepository<Department,
Long> { }
```

service/DepartmentService.java

```
package com.example.demo.service;
```

```
import java.util.List;
import com.example.demo.model.Department;

public interface DepartmentService {
	Department saveDepartment(Department department);
	List<Department> fetchDepartmentList();
	Department updateDepartment(Department department, Long departmentId);
	void deleteDepartmentById(Long departmentId);
}
```

service/DepartmentServiceImpl.java

```
package com.example.demo.service;
import java.util.List;
import java.util.Objects;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.example.demo.model.Department;
import com.example.demo.repository.DepartmentRepository;

@Service
public class DepartmentServiceImpl implements DepartmentService {

	@Autowired
	private DepartmentRepository departmentRepository;

	@Override
	public Department saveDepartment(Department department) {
		return departmentRepository.save(department);
	}

	@Override
	public List<Department> fetchDepartmentList() {
```

```
            return (List<Department>) departmentRepository.findAll();
        }

        @Override
        public Department updateDepartment(Department department, Long
departmentId) {
            Department depDB =
departmentRepository.findById(departmentId).get();

            if (Objects.nonNull(department.getName()) &&
!"".equalsIgnoreCase(department.getName())) {
                depDB.setName(department.getName());
            }
            return departmentRepository.save(depDB);
        }

        @Override
        public void deleteDepartmentById(Long departmentId) {
            departmentRepository.deleteById(departmentId);
        }
}
```

controller/DepartmentController.java

```
package com.example.demo.controller;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;
```

```
import com.example.demo.model.Department;
import com.example.demo.service.DepartmentService;

@RestController
public class DepartmentController {
	@Autowired
	private DepartmentService departmentService;

	@PostMapping("/departments")
	public Department saveDepartment(@RequestBody Department department)
{
		return departmentService.saveDepartment(department);
	}

	@GetMapping("/departments")
	public List<Department> fetchDepartmentList() {
		return departmentService.fetchDepartmentList();
	}

	@PutMapping("/departments/{id}")
	public Department updateDepartment(@RequestBody Department
department, @PathVariable("id") Long departmentId) {
		return departmentService.updateDepartment(department,
departmentId);
	}

	@DeleteMapping("/departments/{id}")
	public String deleteDepartmentById(@PathVariable("id") Long
departmentId) {
		departmentService.deleteDepartmentById(departmentId);
		return "Deleted Successfully";
	}
}
```

Thử nghiệm

B1. Khởi động phần mềm Postman, sau đó tạo collection và thêm các request tương ứng với các API đã khai báo trong controller.

URL	http://localhost:8080/departments
Body type	Raw (JSON)

B2. Tương ứng với mỗi API:

	GET	**POST**	**PUT**	**DELETE**
Parameter	None	None	/{id}	/{id}

Body mẫu	None	{
		“name”: “AB”
		}
			{
			“name”: “AB”
			}	

Ấn Send và chờ kết quả.

Hướng dẫn 2: Tạo Rest API bằng ASP.NET Core Web Application

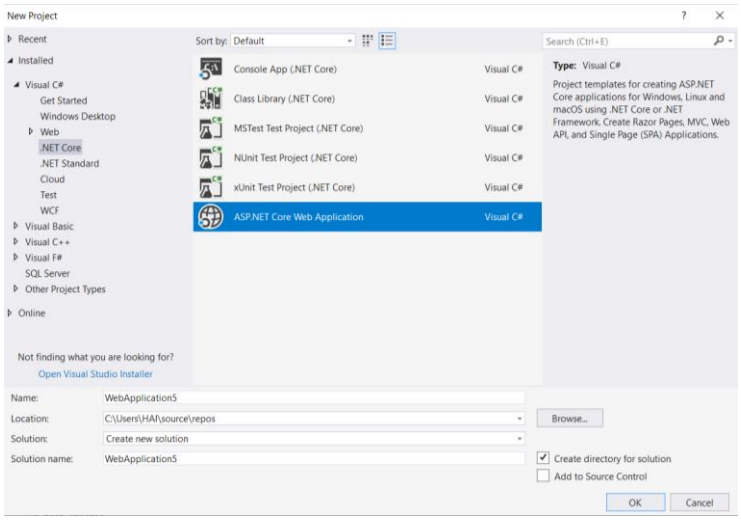
Công cụ thực hành

- Visual Studio IDE [Download]

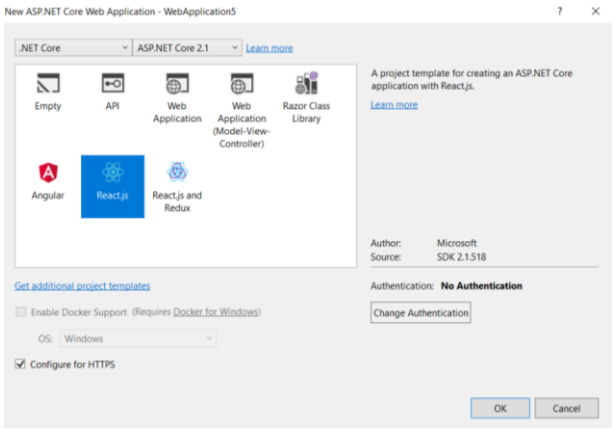
Tạo project

.NET Core là framework giúp chúng ta tạo các web application đa hệ điều hành, nên chúng ta sẽ ưu tiên sử dụng chúng hơn .NET Framework.

B1. Để tạo một project ASP.NET Core, chúng ta khởi động Visual Studio IDE, sau đó chọn File > New > Project và chọn ASP.NET Core Web Application.



B2. Nhấn OK và chọn kiểu Client, chúng ta chọn Empty nếu muốn source code Front-end và Back-end tách rời và chọn các kiểu còn lại, đặc biệt là các framework SPA thông dụng hiện nay như Angular và ReactJS nếu ngược lại.



B3. Project sẽ theo kiến trúc MVC nên source code chính sẽ nằm trong thư mục Controller, theo mẫu thì lớp Controller và Model nằm chung một file nhưng chúng ta có thể tách ra để cấu trúc project rõ ràng hơn.

B4. Để chạy ứng dụng, chúng ta click vào mũi tên màu xanh (IIS Express) và đợi Visual Studio tải các package liên quan. Ứng dụng sẽ hoạt động ở một port nào đó, chúng ta có thể thay đổi tại **Properties \ launchSetting.json.**

Tài liệu tham khảo

[1] ASP.NET Core MVC

[2] Spring Boot

[3] Source code mẫu