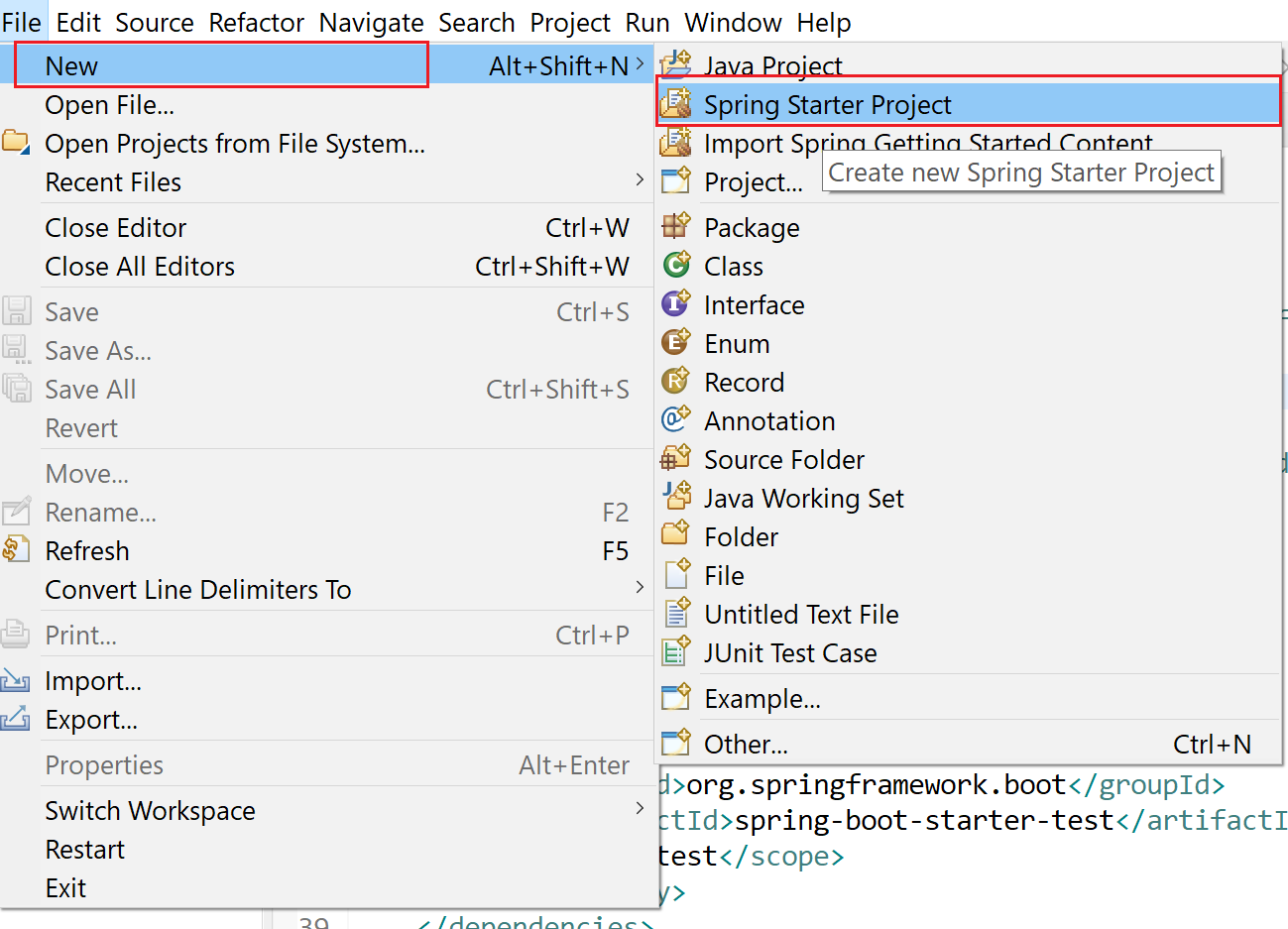
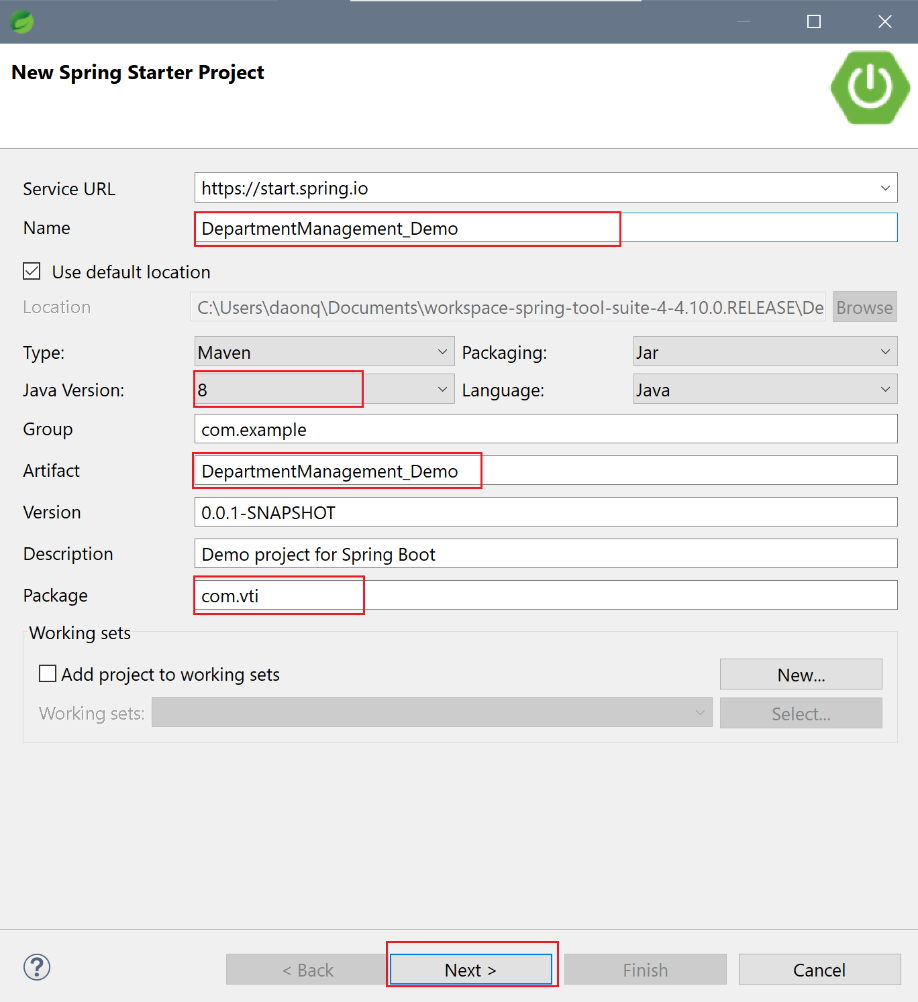
# Lab\_3: Tạo Project trả ra API Quản lý Account, Department, Possition trong Database TestingSystem:

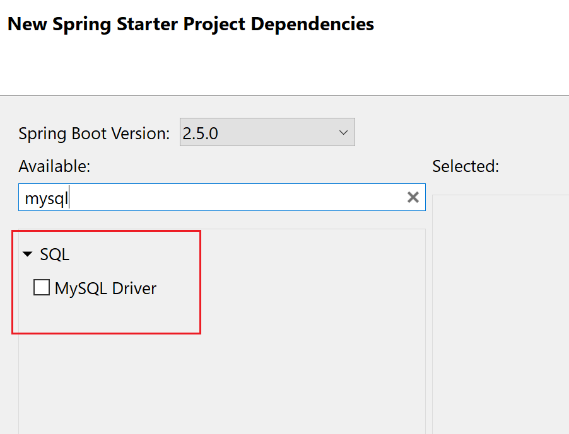
* Thực hiện các chức năng xem, thêm, sửa xóa tạo dữ liệu API với bảng Account, chức năng xem với 2 bảng Department và Possition.

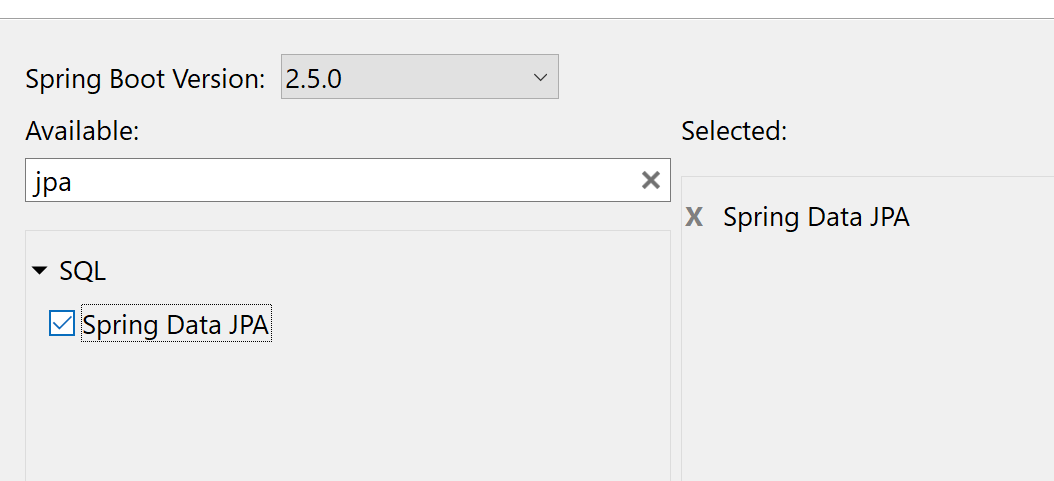
# Tạo mới 1 Project Spring Boot :

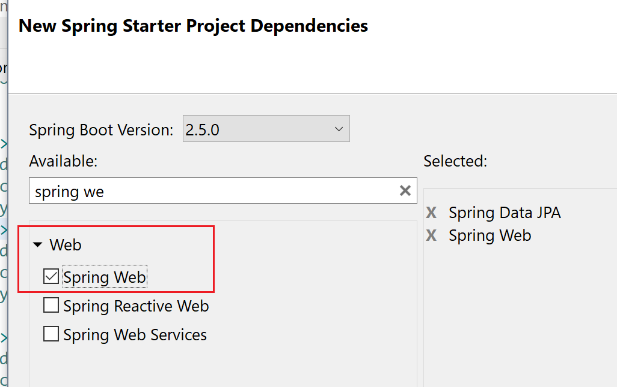
* Tạo mới Project Spring: Account\_Management\_DEMO 
* Tạo mới Project có tên: Account\_Management\_DEMO (Hình minh họa)

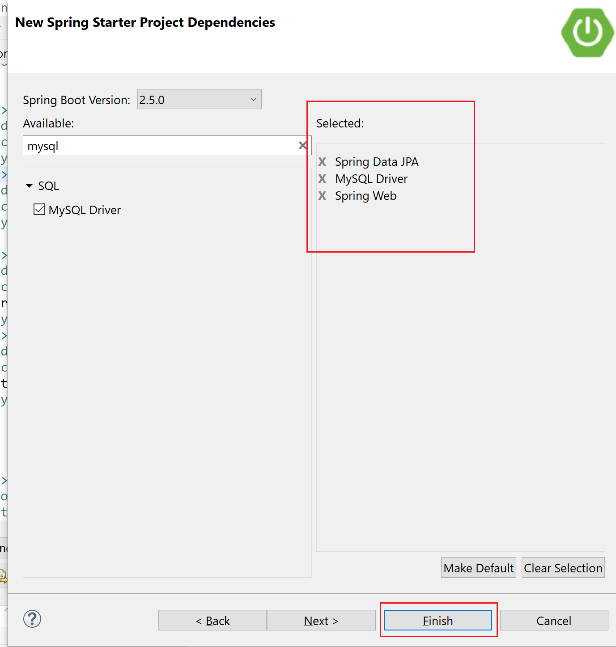


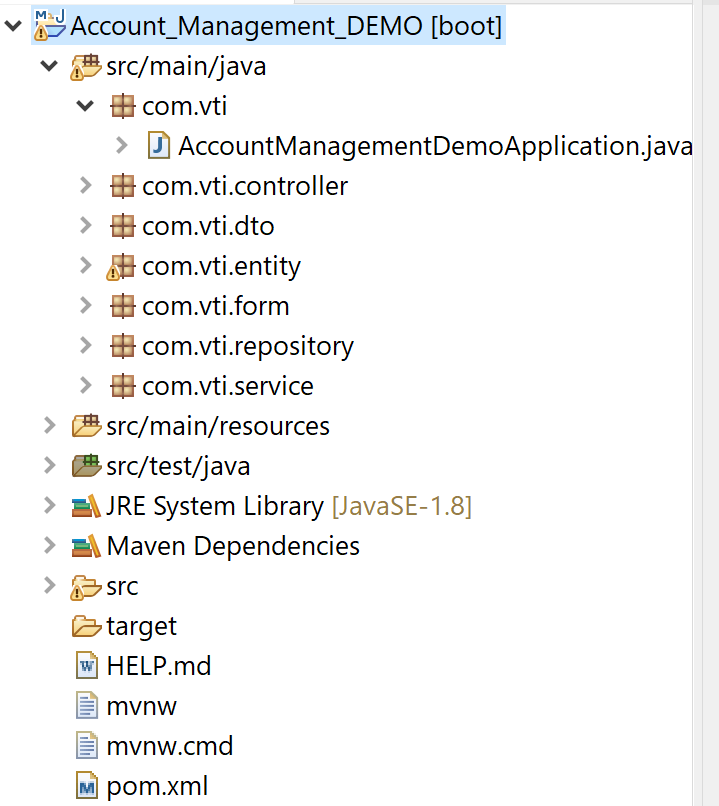
* Lựa chọn các công nghệ để thêm vào Project:



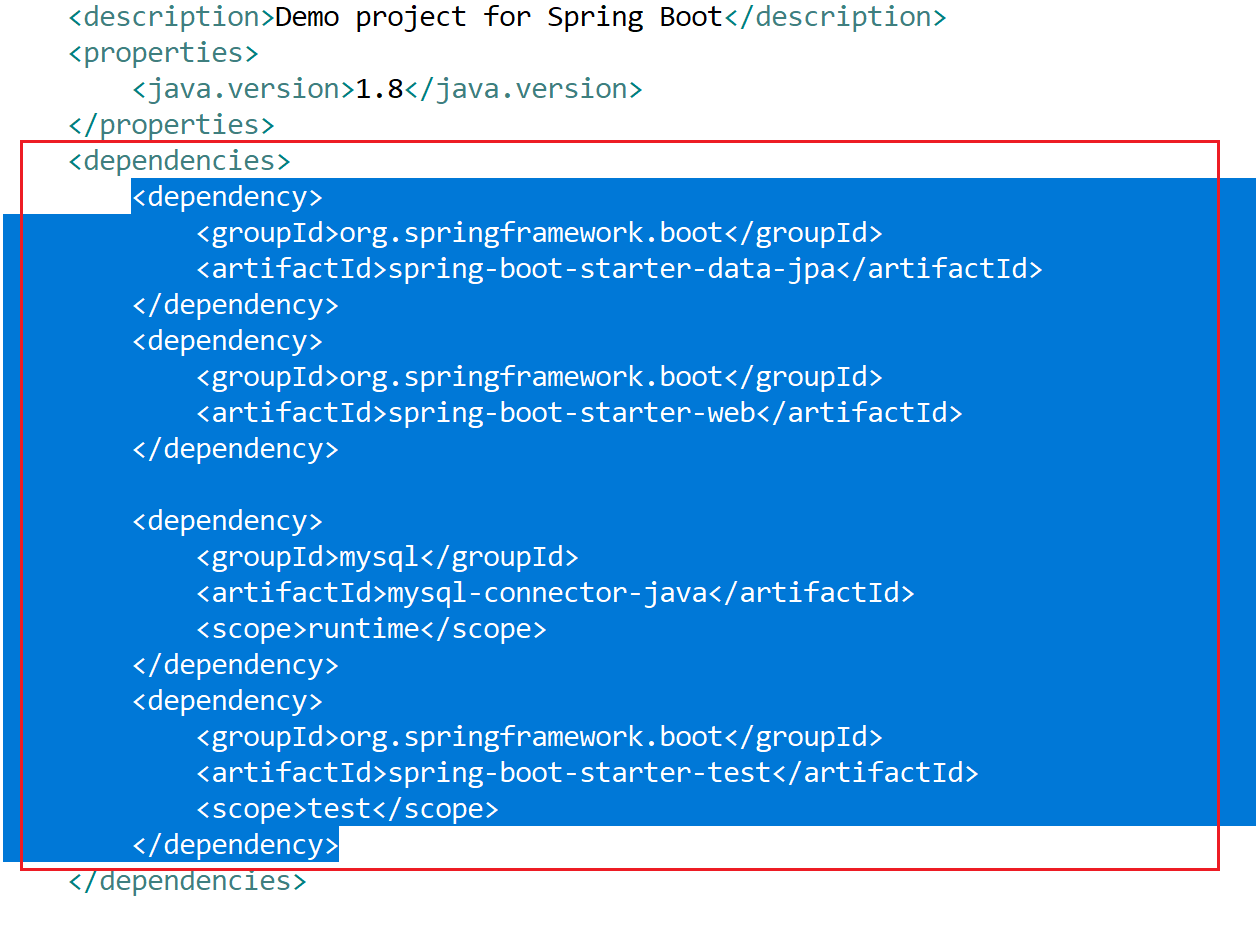




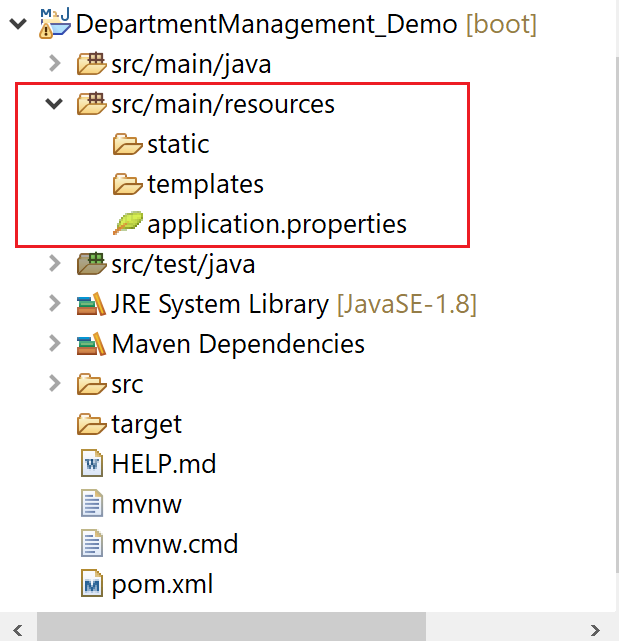




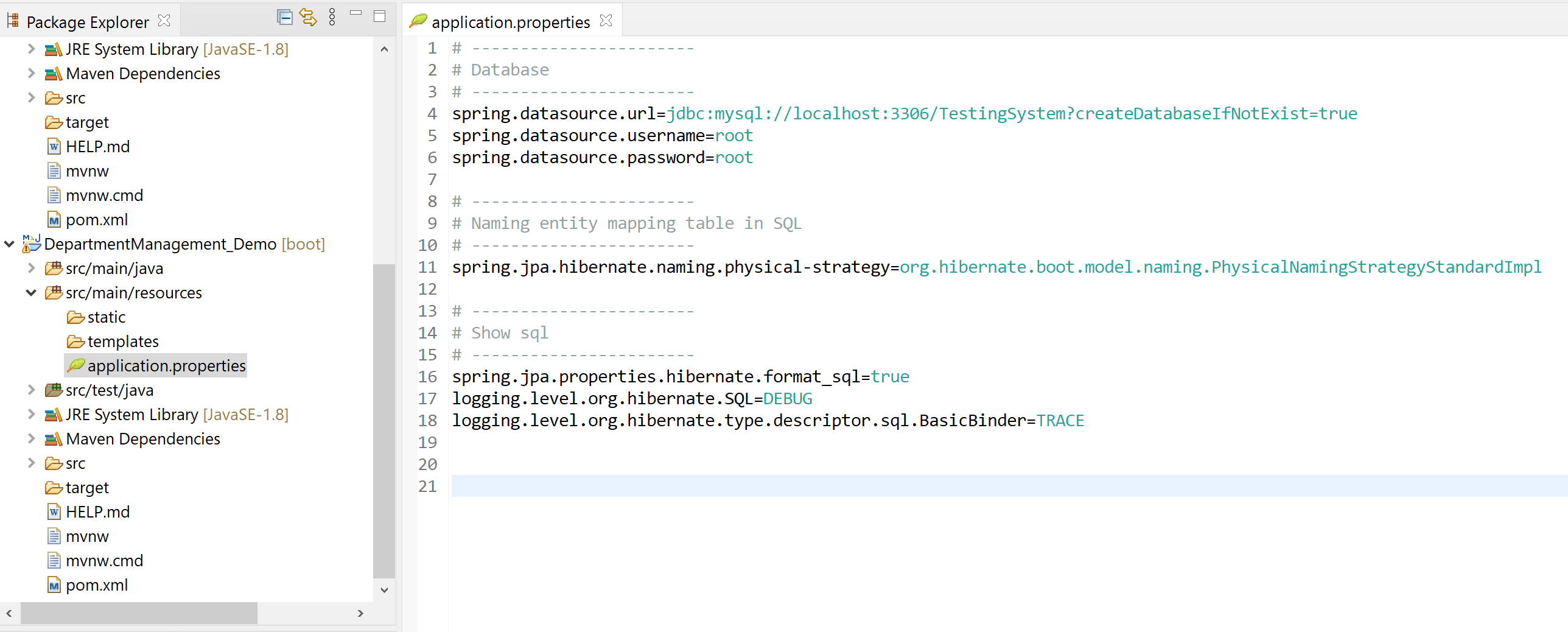
* Kiểm tra lại các Dependecy đã được thêm vào file pom.xml



* Cấu hình lại file application.properties để lưu các thông tin kết nối tới file Database, khi tạo mới project, file này nội dung sẽ null.



|  |
| --- |
| # -----------------------  # Database  # -----------------------  spring.datasource.url=jdbc:mysql://localhost:3306/TestingSystem?createDatabaseIfNotExist=true  spring.datasource.username=root  spring.datasource.password=root  # -----------------------  # Naming entity mapping table in SQL  # -----------------------  spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl  # -----------------------  # Show sql  # -----------------------  spring.jpa.properties.hibernate.format\_sql=true  logging.level.org.hibernate.SQL=DEBUG  logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE |

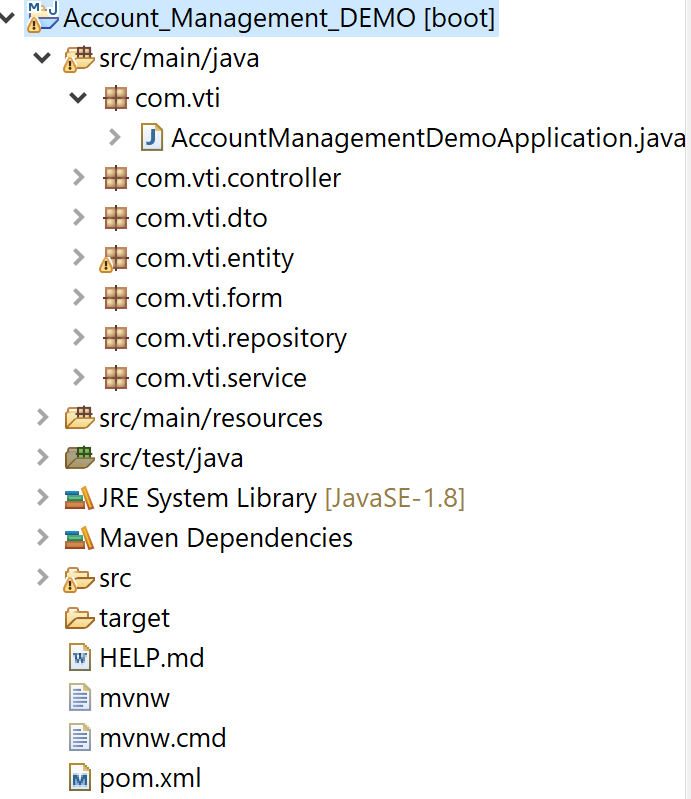


# Tạo DB TestingSystem trong MySqlWorkbench

* Trong phần này thực hiện Demo trên 3 bảng Account, Department, Possition

|  |
| --- |
| DROP DATABASE IF EXISTS TestingSystem;  CREATE DATABASE TestingSystem;  USE TestingSystem;  DROP TABLE IF EXISTS Department;  CREATE TABLE Department (  DepartmentID TINYINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,  DepartmentName NVARCHAR(30) NOT NULL UNIQUE KEY  );  DROP TABLE IF EXISTS Position;  CREATE TABLE `Position` (  PositionID TINYINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,  PositionName ENUM('Dev', 'Test', 'Scrum\_Master', 'PM') NOT NULL UNIQUE KEY  );  DROP TABLE IF EXISTS `Account`;  CREATE TABLE `Account` (  AccountID TINYINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,  Email VARCHAR(50) NOT NULL UNIQUE KEY,  Username VARCHAR(50) NOT NULL UNIQUE KEY,  FullName NVARCHAR(50) NOT NULL,  DepartmentID TINYINT UNSIGNED NOT NULL,  PositionID TINYINT UNSIGNED NOT NULL,  CreateDate DATETIME DEFAULT NOW(),  FOREIGN KEY (DepartmentID)  REFERENCES Department (DepartmentID),  FOREIGN KEY (PositionID)  REFERENCES `Position` (PositionID)  ); |

# Tạo cấu trúc Project



* Tạo các Package: **com.vti.entity, com.vti.repository, com.vti.service, com.vti.controller, com.vti.form, com.vti.dto**

# Tạo Class Department trong entity

* Trong class này chú ý khai báo để kết nối Mapping với Hibernate.

|  |
| --- |
| **package** com.vti.entity;  **import** java.io.Serializable;  **import** java.util.List;  **import** javax.persistence.Column;  **import** javax.persistence.Entity;  **import** javax.persistence.FetchType;  **import** javax.persistence.GeneratedValue;  **import** javax.persistence.GenerationType;  **import** javax.persistence.Id;  **import** javax.persistence.OneToMany;  **import** javax.persistence.Table;  **import** org.hibernate.annotations.Cascade;  **import** org.hibernate.annotations.CascadeType;  @Entity  @Table(name = "Department", catalog = "TestingSystem")  **public** **class** Department **implements** Serializable {  @Column(name = "DepartmentID")  @Id  @GeneratedValue(strategy = GenerationType.***IDENTITY***)  **private** **short** id;  @Column(name = "DepartmentName", length = 30, nullable = **false**, unique = **true**)  **private** String name;  @OneToMany(mappedBy = "department", fetch = FetchType.***EAGER***)  @Cascade(value = { CascadeType.***REMOVE***, CascadeType.***SAVE\_UPDATE*** })  **private** List<Account> account;  **public** Department() {  **super**();  }  /\*\*  \* **@return** the id  \*/  **public** **short** getId() {  **return** id;  }  /\*\*  \* **@param** id the id to set  \*/  **public** **void** setId(**short** id) {  **this**.id = id;  }  /\*\*  \* **@return** the name  \*/  **public** String getName() {  **return** name;  }  /\*\*  \* **@param** name the name to set  \*/  **public** **void** setName(String name) {  **this**.name = name;  }  /\*\*  \* **@return** the account  \*/  **public** List<Account> getAccount() {  **return** account;  }  /\*\*  \* **@param** account the account to set  \*/  **public** **void** setAccount(List<Account> account) {  **this**.account = account;  }  @Override  **public** String toString() {  **return** "Department [id=" + id + ", name=" + name + "]";  }  } |

# Tạo Class Position trong entity

* Trong class này chú ý khai báo để kết nối Mapping với Hibernate.

|  |
| --- |
| **package** com.vti.entity;  **import** java.io.Serializable;  **import** java.util.List;  **import** javax.persistence.Column;  **import** javax.persistence.Entity;  **import** javax.persistence.EnumType;  **import** javax.persistence.Enumerated;  **import** javax.persistence.GeneratedValue;  **import** javax.persistence.GenerationType;  **import** javax.persistence.Id;  **import** javax.persistence.OneToMany;  **import** javax.persistence.Table;  @Entity  @Table(name = "Position", catalog = "TestingSystem")  **public** **class** Position **implements** Serializable {  **private** **static** **final** **long** ***serialVersionUID*** = 1L;  @Column(name = "PositionID")  @Id  @GeneratedValue(strategy = GenerationType.***IDENTITY***)  **private** **short** id;  @Column(name = "PositionName", nullable = **false**, unique = **true**)  @Enumerated(EnumType.***STRING***)  **private** PositionName name;  **public** **enum** PositionName {  ***Dev***, ***Test***, ***Scrum\_Master***, ***PM***  }  @OneToMany(mappedBy = "position")  List<Account> accounts;  **public** Position() {  **super**();  }  /\*\*  \* **@return** the id  \*/  **public** **short** getId() {  **return** id;  }  /\*\*  \* **@param** id the id to set  \*/  **public** **void** setId(**short** id) {  **this**.id = id;  }  /\*\*  \* **@return** the name  \*/  **public** PositionName getName() {  **return** name;  }  /\*\*  \* **@param** name the name to set  \*/  **public** **void** setName(PositionName name) {  **this**.name = name;  }  /\*\*  \* **@return** the accounts  \*/  **public** List<Account> getAccounts() {  **return** accounts;  }  /\*\*  \* **@param** accounts the accounts to set  \*/  **public** **void** setAccounts(List<Account> accounts) {  **this**.accounts = accounts;  }  } |

# Tạo Class Account trong entity

* Trong class này chú ý khai báo để kết nối Mapping với Hibernate.

|  |
| --- |
| **package** com.vti.entity;  **import** java.io.Serializable;  **import** java.util.Date;  **import** javax.persistence.Column;  **import** javax.persistence.Entity;  **import** javax.persistence.GeneratedValue;  **import** javax.persistence.GenerationType;  **import** javax.persistence.Id;  **import** javax.persistence.JoinColumn;  **import** javax.persistence.ManyToOne;  **import** javax.persistence.Table;  **import** javax.persistence.Temporal;  **import** javax.persistence.TemporalType;  **import** org.hibernate.annotations.Cascade;  **import** org.hibernate.annotations.CascadeType;  **import** org.hibernate.annotations.CreationTimestamp;  @Entity  @Table(name = "`Account`", catalog = "TestingSystem")  **public** **class** Account **implements** Serializable {  **private** **static** **final** **long** ***serialVersionUID*** = 1L;  @Column(name = "AccountID")  @Id  @GeneratedValue(strategy = GenerationType.***IDENTITY***)  **private** **short** id;  @Column(name = "Email", length = 50, nullable = **false**, unique = **true**, updatable = **false**)  **private** String email;  @Column(name = "Username", length = 50, nullable = **false**, unique = **true**, updatable = **false**)  **private** String username;  @Column(name = "FullName", length = 50, nullable = **false**)  **private** String fullname;  @ManyToOne  @JoinColumn(name = "DepartmentID", nullable = **false**)  @Cascade(value = { CascadeType.***REMOVE***, CascadeType.***SAVE\_UPDATE*** })  **private** Department department;  @ManyToOne  @JoinColumn(name = "PositionID", nullable = **false**)  **private** Position position;    @Column(name = "CreateDate")  @Temporal(TemporalType.***TIMESTAMP***)  @CreationTimestamp  **private** Date createDate;  **public** Account() {  **super**();  }  /\*\*  \* **@return** the id  \*/  **public** **short** getId() {  **return** id;  }  /\*\*  \* **@param** id the id to set  \*/  **public** **void** setId(**short** id) {  **this**.id = id;  }  /\*\*  \* **@return** the email  \*/  **public** String getEmail() {  **return** email;  }  /\*\*  \* **@param** email the email to set  \*/  **public** **void** setEmail(String email) {  **this**.email = email;  }  /\*\*  \* **@return** the username  \*/  **public** String getUsername() {  **return** username;  }  /\*\*  \* **@param** username the username to set  \*/  **public** **void** setUsername(String username) {  **this**.username = username;  }  /\*\*  \* **@return** the fullname  \*/  **public** String getFullname() {  **return** fullname;  }  /\*\*  \* **@param** fullname the fullname to set  \*/  **public** **void** setFullname(String fullname) {  **this**.fullname = fullname;  }  /\*\*  \* **@return** the department  \*/  **public** Department getDepartment() {  **return** department;  }  /\*\*  \* **@param** department the department to set  \*/  **public** **void** setDepartment(Department department) {  **this**.department = department;  }  /\*\*  \* **@return** the createDate  \*/  **public** Date getCreateDate() {  **return** createDate;  }  /\*\*  \* **@param** createDate the createDate to set  \*/  **public** **void** setCreateDate(Date createDate) {  **this**.createDate = createDate;  }  /\*\*  \* **@return** the position  \*/  **public** Position getPosition() {  **return** position;  }  /\*\*  \* **@param** position the position to set  \*/  **public** **void** setPosition(Position position) {  **this**.position = position;  }  @Override  **public** String toString() {  **return** "Account [id=" + id + ", email=" + email + ", username=" + username + ", fullname=" + fullname  + ", department=" + department + ", createDate=" + createDate + "]";  }  } |

# Tạo Class DepartmentDto

* Class này dùng để chuyển đổi giữa dữ liệu lấy được từ DB sang dữ hiển thị đẩy lên Frontend.

|  |
| --- |
| **package** com.vti.dto;  **public** **class** DepartmentDto {  **private** **short** id;  **private** String name;  **public** DepartmentDto(**short** id, String name) {  **this**.id = id;  **this**.name = name;  }  **public** **short** getId() {  **return** id;  }  **public** String getName() {  **return** name;  }  } |

# Tạo Class PossitonDto

* Class này dùng để chuyển đổi giữa dữ liệu lấy được từ DB sang dữ hiển thị đẩy lên Frontend.

|  |
| --- |
| **package** com.vti.dto;  **public** **class** PossitonDto {  **private** **short** id;  **private** String name;  **public** PossitonDto(**short** id, String name) {  **this**.id = id;  **this**.name = name;  }  **public** **short** getId() {  **return** id;  }  **public** String getName() {  **return** name;  }  } |

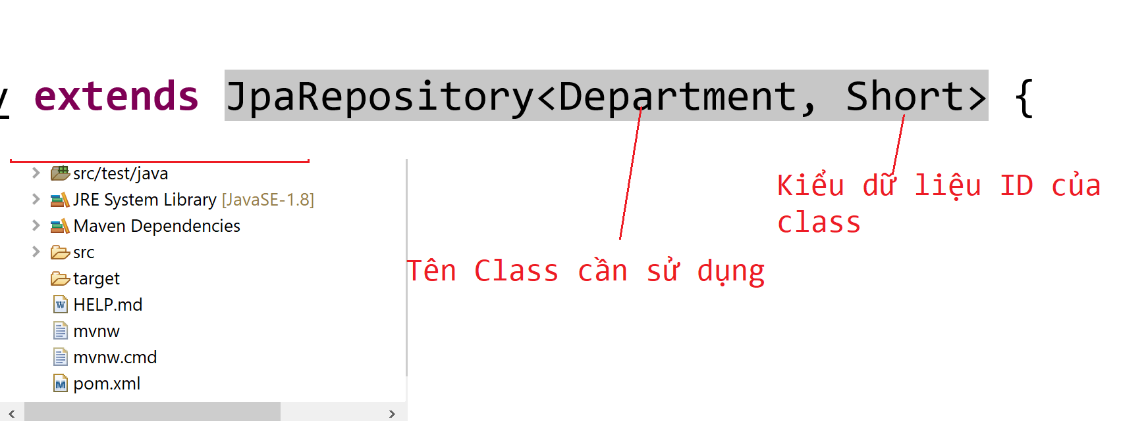
# Tạo Class AccontDto

* Class này dùng để chuyển đổi giữa dữ liệu lấy được từ DB sang dữ hiển thị đẩy lên Frontend.

|  |
| --- |
| **package** com.vti.dto;  **import** java.util.Date;  **import** com.fasterxml.jackson.annotation.JsonFormat;  **public** **class** AccontDto {  **private** **short** id;  **private** String email;  **private** String username;  **private** String fullname;  **private** String department;  **private** String position;  @JsonFormat(pattern="yyyy-MM-dd")  **private** Date createDate;  **public** AccontDto(**short** id, String email, String username, String fullname, String department, String position,  Date createDate) {  **super**();  **this**.id = id;  **this**.email = email;  **this**.username = username;  **this**.fullname = fullname;  **this**.department = department;  **this**.position = position;  **this**.createDate = createDate;  }  **public** **short** getId() {  **return** id;  }  **public** **void** setId(**short** id) {  **this**.id = id;  }  **public** String getEmail() {  **return** email;  }  **public** **void** setEmail(String email) {  **this**.email = email;  }  **public** String getUsername() {  **return** username;  }  **public** **void** setUsername(String username) {  **this**.username = username;  }  **public** String getFullname() {  **return** fullname;  }  **public** **void** setFullname(String fullname) {  **this**.fullname = fullname;  }    **public** String getDepartment() {  **return** department;  }  **public** **void** setDepartment(String department) {  **this**.department = department;  }  **public** String getPosition() {  **return** position;  }  **public** **void** setPosition(String position) {  **this**.position = position;  }  **public** Date getCreateDate() {  **return** createDate;  }  **public** **void** setCreateDate(Date createDate) {  **this**.createDate = createDate;  }  } |

# Tạo Interface IDepartmentRepository

* Trong phần này chú ý Extend JpaRepository của Spring data JPA, chỉ cần Extend mà k cần làm thêm gì, do 1 số phương thức hay dùng JPA đã hỗ trợ sẵn.



|  |
| --- |
| **package** com.vti.repository;  **import** org.springframework.data.jpa.repository.JpaRepository;  **import** com.vti.entity.Department;  **public** **interface** IDepartmentRepository **extends** JpaRepository<Department, Short> {  **public** Department findByName(String name);  **public** **boolean** existsByName(String name);  } |

# Tạo Interface IPossitionRepository

* Trong phần này chú ý Extend JpaRepository của Spring data JPA, chỉ cần Extend mà k cần làm thêm gì, do 1 số phương thức hay dùng JPA đã hỗ trợ sẵn.

|  |
| --- |
| **package** com.vti.repository;  **import** org.springframework.data.jpa.repository.JpaRepository;  **import** com.vti.entity.Position;  **public** **interface** IPossitionRepository **extends** JpaRepository<Position, Short>{  } |

# Tạo Interface IAccountRepository

* Trong phần này chú ý Extend JpaRepository của Spring data JPA, chỉ cần Extend mà k cần làm thêm gì, do 1 số phương thức hay dùng JPA đã hỗ trợ sẵn.

|  |
| --- |
| **package** com.vti.repository;  **import** org.springframework.data.jpa.repository.JpaRepository;  **import** com.vti.entity.Account;  **public** **interface** IAccountRepository **extends** JpaRepository<Account, Short> {  } |

# Tạo Interface IDepartmentService

|  |
| --- |
| **package** com.vti.service;  **import** java.util.List;  **import** com.vti.entity.Department;  **public** **interface** IDepartmentService {  **public** List<Department> getAllDepartments();  } |

# Tạo Interface IPossitionService

|  |
| --- |
| **package** com.vti.service;  **import** java.util.List;  **import** com.vti.entity.Position;  **public** **interface** IPossitionService {  **public** List<Position> getAllPossition();  } |

# Tạo Interface IAccountService

|  |
| --- |
| **package** com.vti.service;  **import** java.util.List;  **import** com.vti.entity.Account;  **import** com.vti.form.AccountFormForCreating;  **import** com.vti.form.AccountFormForUpdating;  **public** **interface** IAccountService {  **public** List<Account> getAllAccount();  **public** Account getAccountById(**short** id);  //  // public Account getAccountByName(String name);  //  **public** **void** createAccount(AccountFormForCreating form);  **public** **void** updateAccount(**short** id, AccountFormForUpdating form);  **public** **void** deleteAccount(**short** id);  //  // public boolean isAccountExistsByID(short id);  //  // public boolean isAccountExistsByName(String name);  } |

# Tạo Class DepartmentService

* Trong Class này sẽ gọi các phương thức của JPA đã xây dựng sẵn thông qua: IDepartmentRepository.

|  |
| --- |
| **package** com.vti.service;  **import** java.util.List;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.stereotype.Service;  **import** com.vti.entity.Department;  **import** com.vti.repository.IDepartmentRepository;  @Service  **public** **class** DepartmentService **implements** IDepartmentService {  @Autowired  **private** IDepartmentRepository departmentRepository;  **public** List<Department> getAllDepartments() {  **return** departmentRepository.findAll();  }  } |

# Tạo Class PossitionService

* Trong Class này sẽ gọi các phương thức của JPA đã xây dựng sẵn thông qua: IPossitionRepository.

|  |
| --- |
| **package** com.vti.service;  **import** java.util.List;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.stereotype.Service;  **import** com.vti.entity.Position;  **import** com.vti.repository.IPossitionRepository;  @Service  **public** **class** PossitionService **implements** IPossitionService {  @Autowired  **private** IPossitionRepository possitionRepository;  @Override  **public** List<Position> getAllPossition() {  // **TODO** Auto-generated method stub  **return** possitionRepository.findAll();  }  } |

# Tạo Class AccountService

* Trong Class này sẽ gọi các phương thức của JPA đã xây dựng sẵn thông qua: IAccountRepository.

|  |
| --- |
| **package** com.vti.service;  **import** java.util.List;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.stereotype.Service;  **import** com.vti.entity.Account;  **import** com.vti.entity.Department;  **import** com.vti.entity.Position;  **import** com.vti.form.AccountFormForCreating;  **import** com.vti.form.AccountFormForUpdating;  **import** com.vti.repository.IAccountRepository;  **import** com.vti.repository.IDepartmentRepository;  **import** com.vti.repository.IPossitionRepository;  @Service  **public** **class** AccountService **implements** IAccountService {  @Autowired  **private** IAccountRepository accountRepository;  @Autowired  **private** IDepartmentRepository departmentRepository;  @Autowired  **private** IPossitionRepository possitionRepository;  @Override  **public** List<Account> getAllAccount() {  // **TODO** Auto-generated method stub  **return** accountRepository.findAll();  }  @Override  **public** Account getAccountById(**short** id) {  // **TODO** Auto-generated method stub  **return** accountRepository.getById(id);  }  @Override  **public** **void** deleteAccount(**short** id) {  accountRepository.deleteById(id);  }  @Override  **public** **void** createAccount(AccountFormForCreating form) {  Account account = **new** Account();  Department department = departmentRepository.getById(form.getDepartmentId());  Position position = possitionRepository.getById(form.getPositionId());  account.setEmail(form.getEmail());  account.setUsername(form.getUsername());  account.setFullname(form.getFullname());  account.setDepartment(department);  account.setPosition(position);  accountRepository.save(account);  }  @Override  **public** **void** updateAccount(**short** id, AccountFormForUpdating form) {  Account account = accountRepository.getById(id);  Department department = departmentRepository.getById(form.getDepartmentId());  Position position = possitionRepository.getById(form.getPositionId());  account.setFullname(form.getFullname());  account.setDepartment(department);  account.setPosition(position);  accountRepository.save(account);  }  } |

# Tạo Class AccountFormForCreating trong Package form

* Class này dùng để nhận dữ liệu từ FrontEnd đẩy xuống, do các trường dữ liệu người dùng nhập vào có thể không giống hoàn toàn với dữ liệu bảng Account nên cần tạo ra 1 class này sử dụng cho việc tạo mới 1 phòng ban.

|  |
| --- |
| **package** com.vti.form;  **public** **class** AccountFormForCreating {  **private** String email;  **private** String username;  **private** String fullname;  **private** **short** departmentId;  **private** **short** positionId;  **public** AccountFormForCreating() {  }  **public** String getEmail() {  **return** email;  }  **public** **void** setEmail(String email) {  **this**.email = email;  }  **public** String getUsername() {  **return** username;  }  **public** **void** setUsername(String username) {  **this**.username = username;  }  **public** String getFullname() {  **return** fullname;  }  **public** **void** setFullname(String fullname) {  **this**.fullname = fullname;  }  **public** **short** getDepartmentId() {  **return** departmentId;  }  **public** **void** setDepartmentId(**short** departmentId) {  **this**.departmentId = departmentId;  }  **public** **short** getPositionId() {  **return** positionId;  }  **public** **void** setPositionId(**short** positionId) {  **this**.positionId = positionId;  }  } |

# Tạo Class AccountFormForUpdating trong Package form

* Class này dùng để nhận dữ liệu từ FrontEnd đẩy xuống, do các trường dữ liệu người dùng nhập vào có thể không giống hoàn toàn với dữ liệu bảng Account nên cần tạo ra 1 class này sử dụng cho việc tạo mới 1 phòng ban.

|  |
| --- |
| **package** com.vti.form;  **public** **class** AccountFormForUpdating {  **private** String fullname;  **private** **short** departmentId;  **private** **short** positionId;  **public** AccountFormForUpdating() {  **super**();  }  **public** String getFullname() {  **return** fullname;  }  **public** **void** setFullname(String fullname) {  **this**.fullname = fullname;  }  **public** **short** getDepartmentId() {  **return** departmentId;  }  **public** **void** setDepartmentId(**short** departmentId) {  **this**.departmentId = departmentId;  }  **public** **short** getPositionId() {  **return** positionId;  }  **public** **void** setPositionId(**short** positionId) {  **this**.positionId = positionId;  }  } |

# Tạo Class DepartmentController

* Trong Class này có Anotation: **@CrossOrigin("\*"),** cái này dùng để khai báo cho Server biết các ứng dụng có thể truy cập vào API

|  |
| --- |
| **package** com.vti.controller;  **import** java.util.ArrayList;  **import** java.util.List;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.http.HttpStatus;  **import** org.springframework.http.ResponseEntity;  **import** org.springframework.web.bind.annotation.CrossOrigin;  **import** org.springframework.web.bind.annotation.GetMapping;  **import** org.springframework.web.bind.annotation.RequestMapping;  **import** org.springframework.web.bind.annotation.RestController;  **import** com.vti.dto.DepartmentDto;  **import** com.vti.entity.Department;  **import** com.vti.service.IDepartmentService;  @RestController  @RequestMapping(value = "api/v1/departments")  @CrossOrigin("\*")  **public** **class** DepartmentController {  @Autowired  **private** IDepartmentService departmentService;  @GetMapping()  **public** ResponseEntity<?> getAllDepartments() {  List<Department> entities = departmentService.getAllDepartments();  List<DepartmentDto> dtos = **new** ArrayList<>();  // convert entities --> dtos  for (Department entity : entities) {  DepartmentDto dto = new DepartmentDto(entity.getId(), entity.getName());  dtos.add(dto);  }  return new ResponseEntity<>(dtos, HttpStatus.OK);  }  } |

# Tạo Class PossitionController

* Trong Class này có Anotation: **@CrossOrigin("\*"),** cái này dùng để khai báo cho Server biết các ứng dụng có thể truy cập vào API

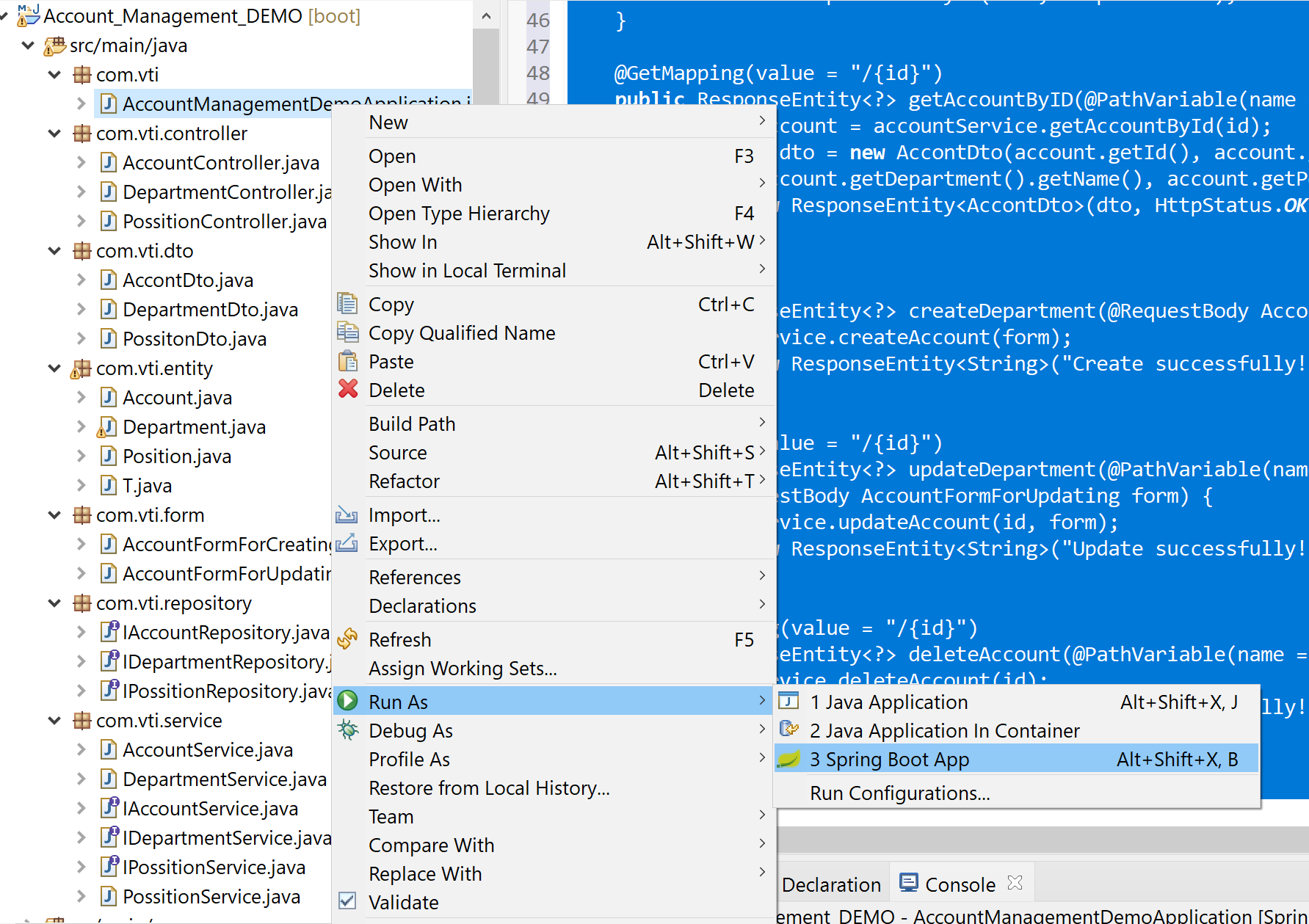
|  |
| --- |
| **package** com.vti.controller;  **import** java.util.ArrayList;  **import** java.util.List;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.http.HttpStatus;  **import** org.springframework.http.ResponseEntity;  **import** org.springframework.web.bind.annotation.CrossOrigin;  **import** org.springframework.web.bind.annotation.GetMapping;  **import** org.springframework.web.bind.annotation.RequestMapping;  **import** org.springframework.web.bind.annotation.RestController;  **import** com.vti.dto.PossitonDto;  **import** com.vti.entity.Position;  **import** com.vti.service.IPossitionService;  @RestController  @RequestMapping(value = "api/v1/possitions")  @CrossOrigin("\*")  **public** **class** PossitionController {  @Autowired  **private** IPossitionService possitionService;  @GetMapping()  **public** ResponseEntity<?> getAllPossitons() {  List<Position> entities = possitionService.getAllPossition();  List<PossitonDto> dtos = **new** ArrayList<>();  **for** (Position entity : entities) {  PossitonDto dto = **new** PossitonDto(entity.getId(), entity.getName().toString());  dtos.add(dto);  }  **return** **new** ResponseEntity<>(dtos, HttpStatus.OK);  }  } |

# Tạo Class AccountController

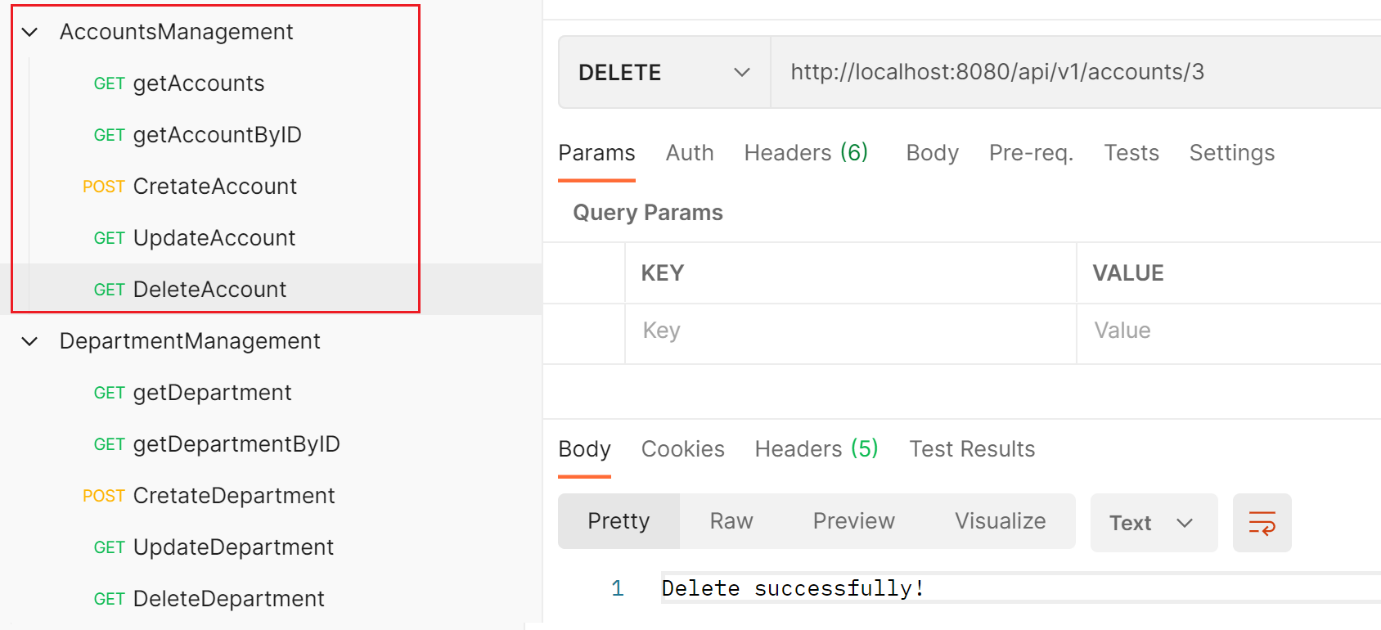
* Trong Class này có Anotation: **@CrossOrigin("\*"),** cái này dùng để khai báo cho Server biết các ứng dụng có thể truy cập vào API

|  |
| --- |
| **package** com.vti.controller;  **import** java.util.ArrayList;  **import** java.util.List;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.http.HttpStatus;  **import** org.springframework.http.ResponseEntity;  **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.RequestMapping;  **import** org.springframework.web.bind.annotation.RestController;  **import** com.vti.dto.AccontDto;  **import** com.vti.entity.Account;  **import** com.vti.form.AccountFormForCreating;  **import** com.vti.form.AccountFormForUpdating;  **import** com.vti.service.IAccountService;  @RestController  @RequestMapping(value = "api/v1/accounts")  **public** **class** AccountController {  @Autowired  **private** IAccountService accountService;  @GetMapping()  **public** ResponseEntity<?> getAllAccount() {  List<Account> entities = accountService.getAllAccount();  List<AccontDto> dtos = **new** ArrayList<>();  // convert entities --> dtos  **for** (Account account : entities) {  AccontDto dto = **new** AccontDto(account.getId(), account.getEmail(), account.getUsername(),  account.getFullname(), account.getDepartment().getName(),  account.getPosition().getName().toString(), account.getCreateDate());  dtos.add(dto);  }  **return** **new** ResponseEntity<>(dtos, HttpStatus.OK);  }  @GetMapping(value = "/{id}")  **public** ResponseEntity<?> getAccountByID(@PathVariable(name = "id") **short** id) {  Account account = accountService.getAccountById(id);  AccontDto dto = **new** AccontDto(account.getId(), account.getEmail(), account.getUsername(), account.getFullname(),  account.getDepartment().getName(), account.getPosition().getName().toString(), account.getCreateDate());  **return** **new** ResponseEntity<AccontDto>(dto, HttpStatus.OK);  }  @PostMapping()  **public** ResponseEntity<?> createDepartment(@RequestBody AccountFormForCreating form) {  accountService.createAccount(form);  **return** **new** ResponseEntity<String>("Create successfully!", HttpStatus.CREATED);  }  @PutMapping(value = "/{id}")  **public** ResponseEntity<?> updateDepartment(@PathVariable(name = "id") **short** id,  @RequestBody AccountFormForUpdating form) {  accountService.updateAccount(id, form);  **return** **new** ResponseEntity<String>("Update successfully!", HttpStatus.OK);  }  @DeleteMapping(value = "/{id}")  **public** ResponseEntity<?> deleteAccount(@PathVariable(name = "id") **short** id) {  accountService.deleteAccount(id);  **return** **new** ResponseEntity<String>("Delete successfully!", HttpStatus.OK);  }  } |

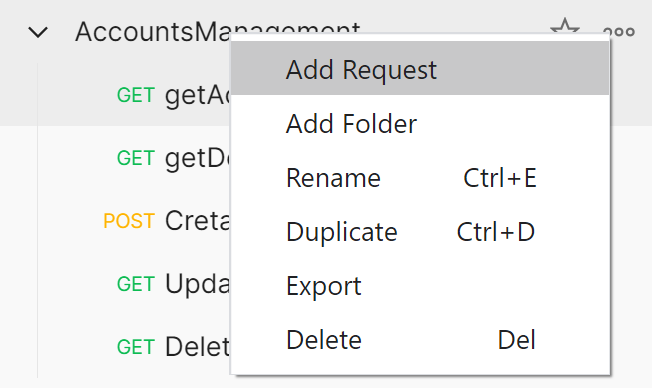
# Chạy chương trình Demo kết quả



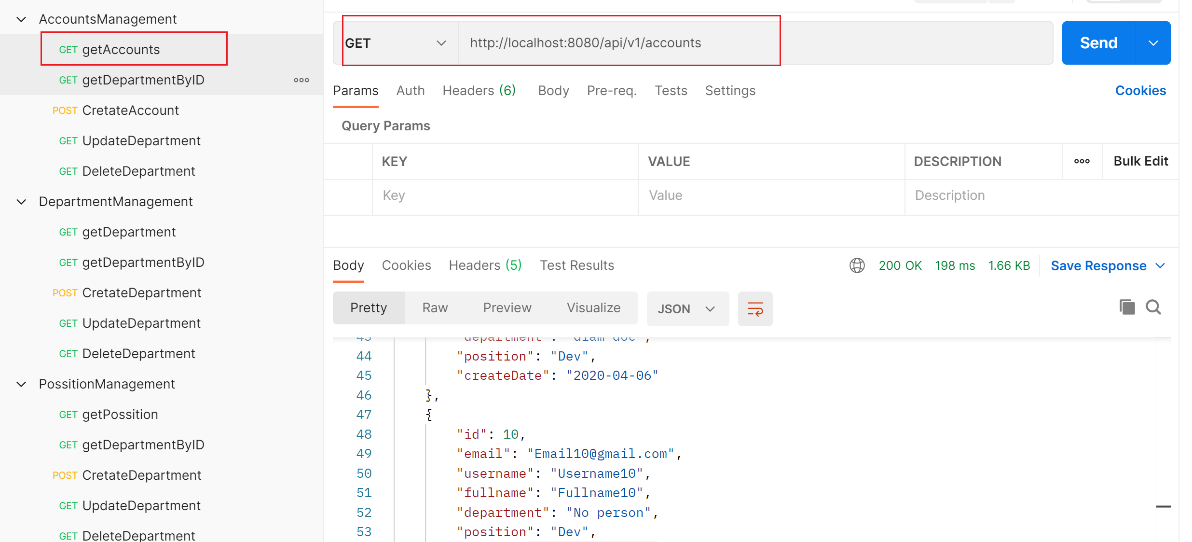
# Sử dụng Postman để Demo kết quả API với Account



* Thực hiện Add các Request:

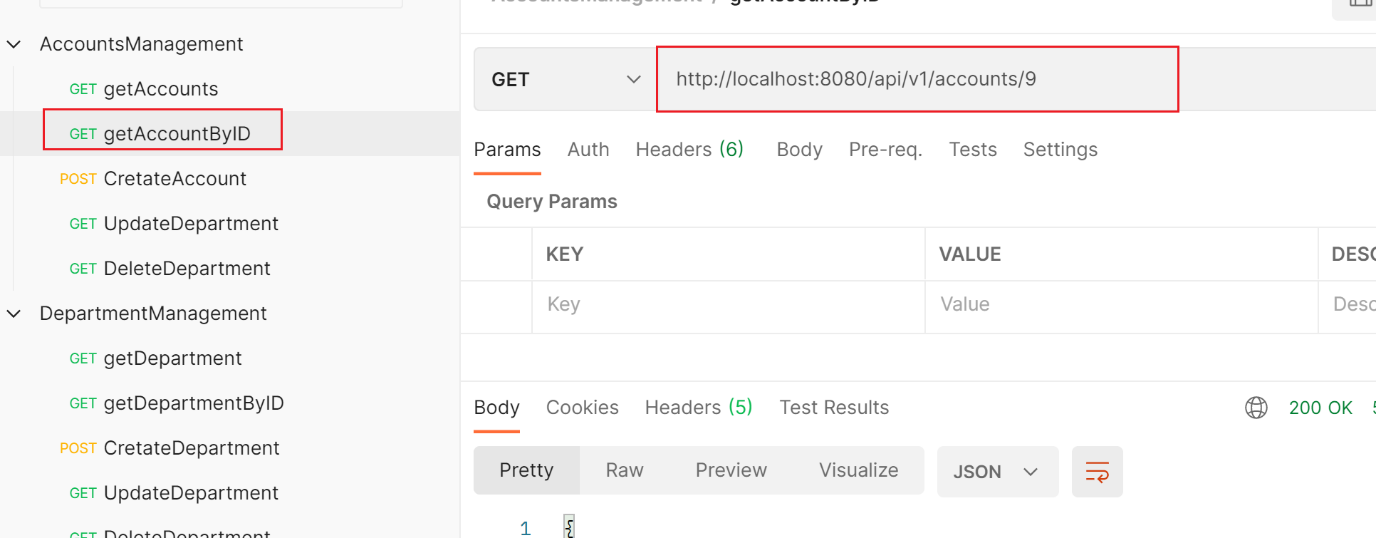


* Get All Account:



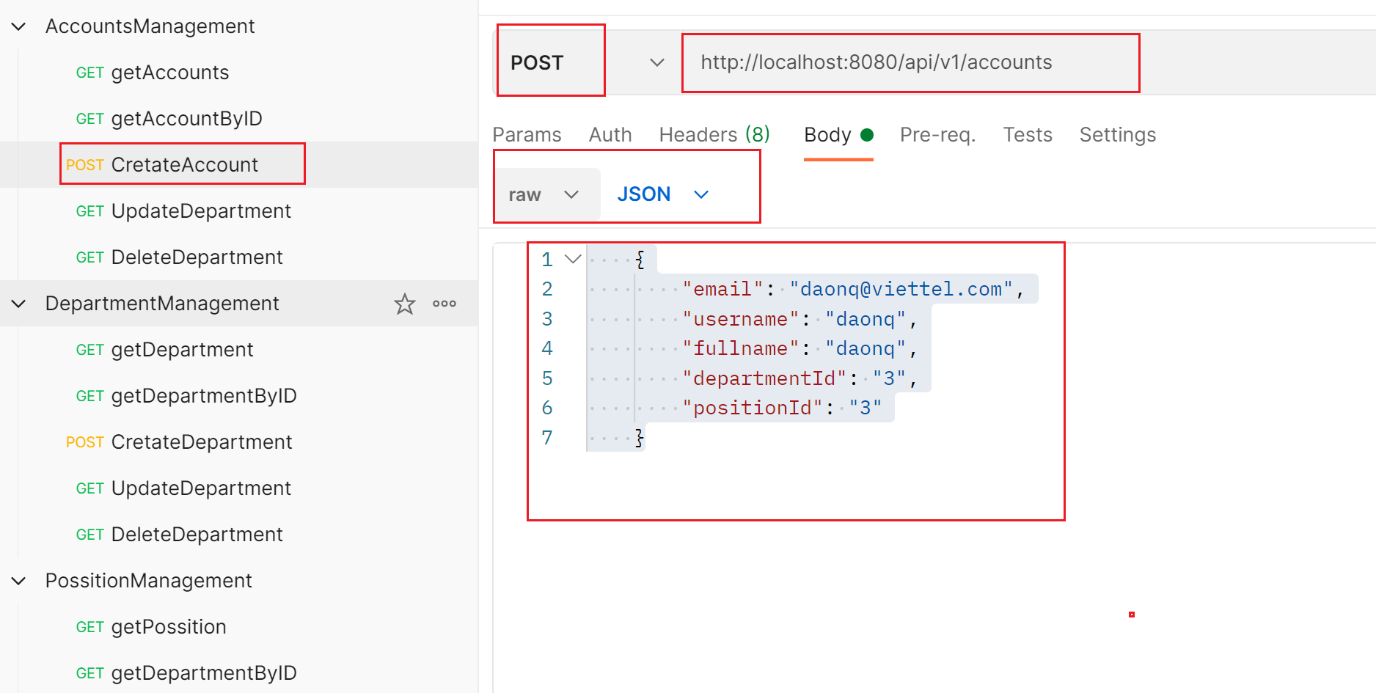
|  |
| --- |
| http://localhost:8080/api/v1/accounts |

* Get getAccountByID



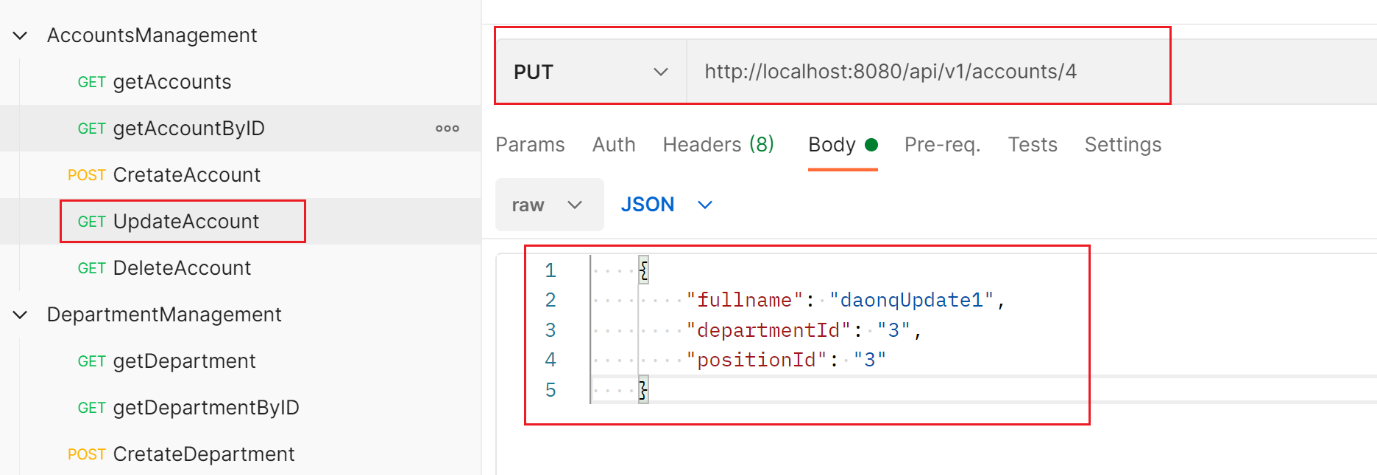
|  |
| --- |
| http://localhost:8080/api/v1/accounts/9 |

* Create Account



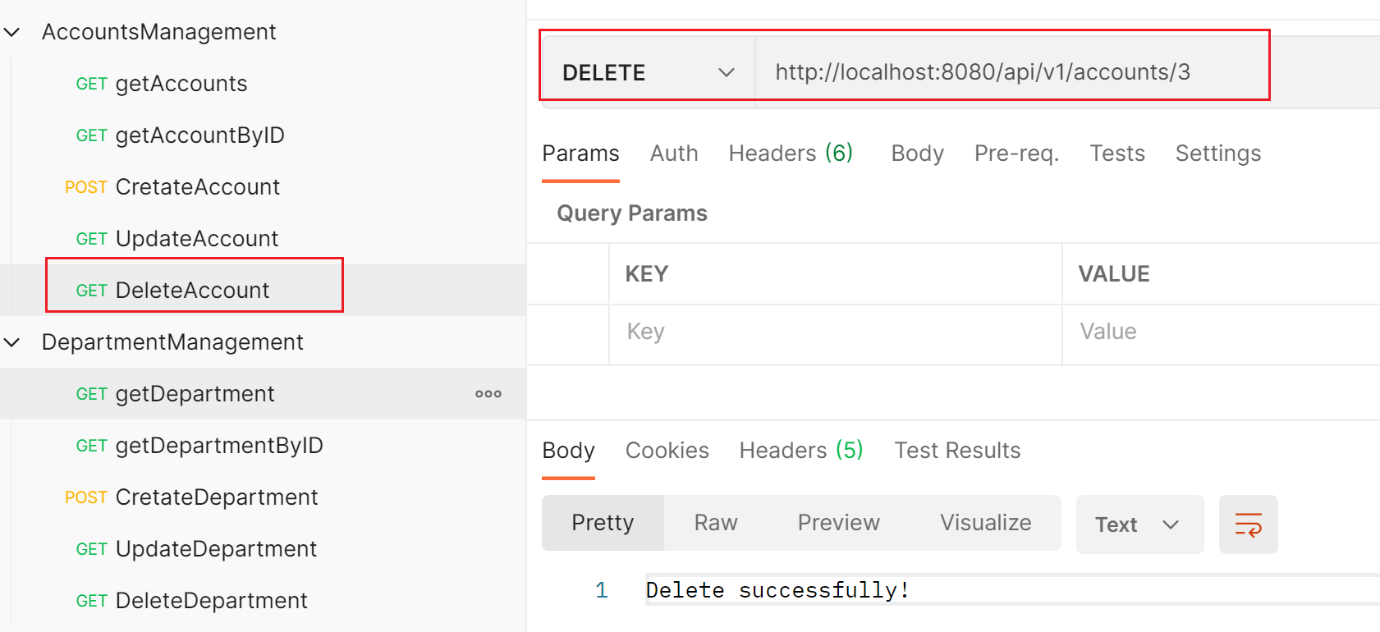
|  |
| --- |
| <http://localhost:8080/api/v1/accounts>      {          "email": "daonq@viettel.com",          "username": "daonq",          "fullname": "daonq",          "departmentId": "3",          "positionId": "3"      } |

* Update Account



|  |
| --- |
| http://localhost:8080/api/v1/accounts/4      {          "fullname": "daonqUpdate1",          "departmentId": "3",          "positionId": "3"      } |

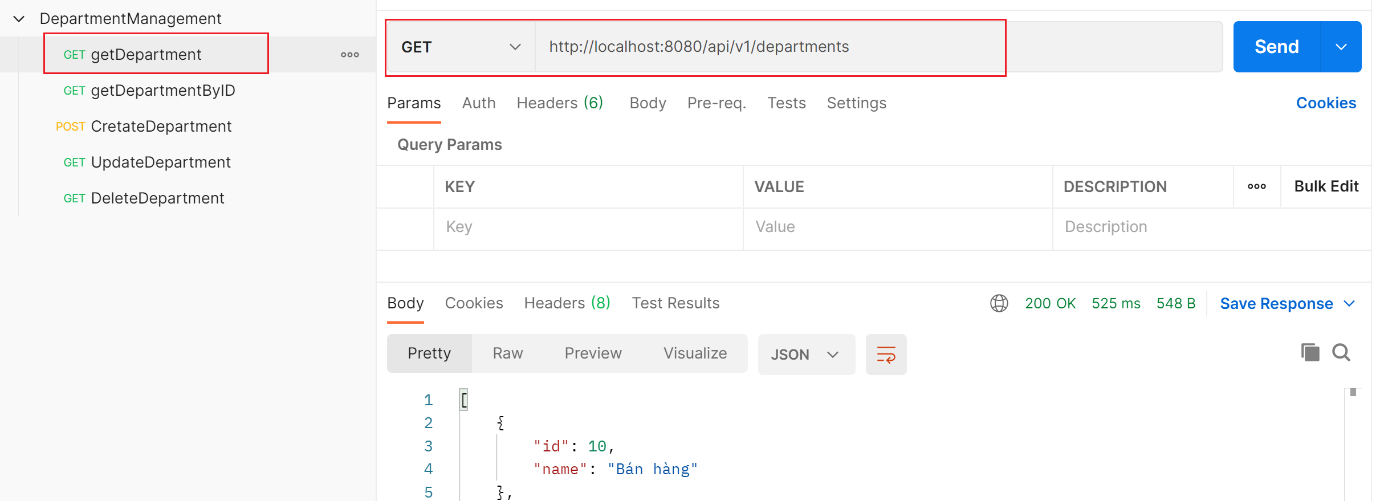
* Delete Account



|  |
| --- |
| http://localhost:8080/api/v1/accounts/3 |

# Sử dụng Postman để Demo kết quả API với Department

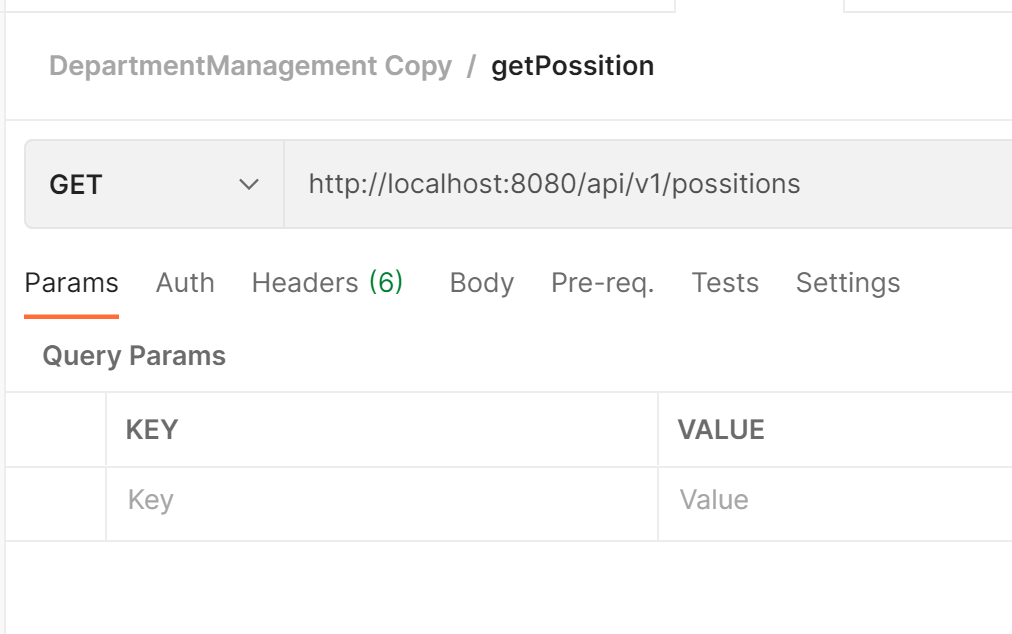
* Get All Department



|  |
| --- |
| <http://localhost:8080/api/v1/departments> |

# Sử dụng Postman để Demo kết quả API với Possition

* Get All Possition



|  |
| --- |
| http://localhost:8080/api/v1/possitions |