[Thực hành] Sử dụng Spring JPA Store Proceduce

Mục tiêu

***Sử dụng được Stored Proceduce với Hibernate.***

***Tạo được dự án Spring MVC* không thông qua *Jakarta EE , tự cấu hình dự án bằng class JAVA***

Điều kiện

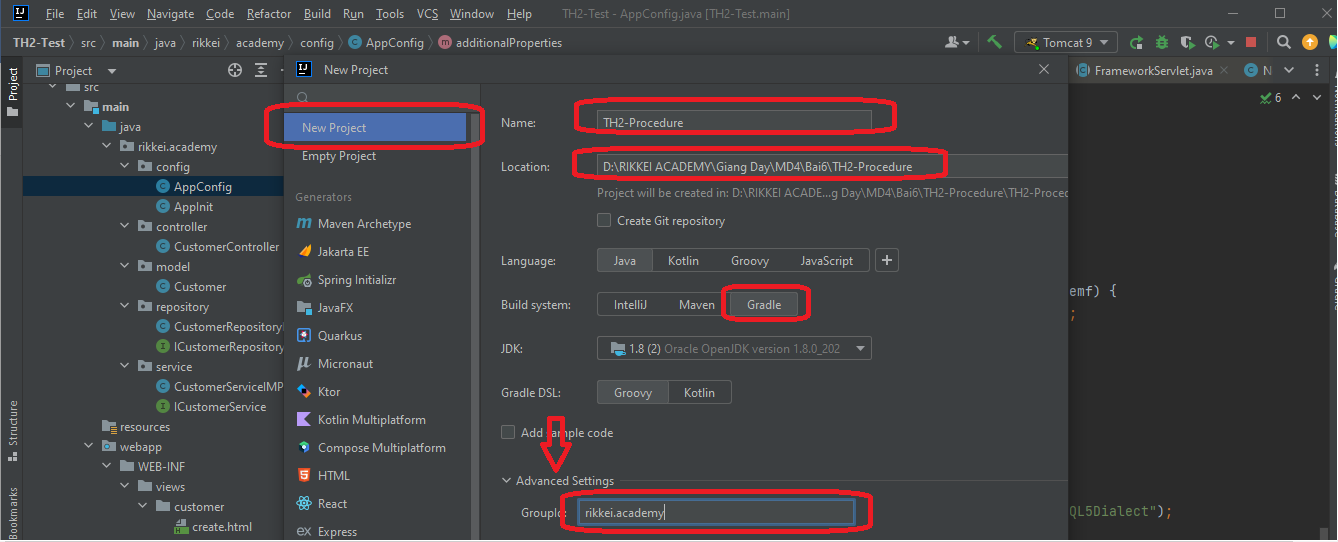
Có kiến thức căn bản về việc sử dụng JPA và Stored Proceduce.

Mô tả

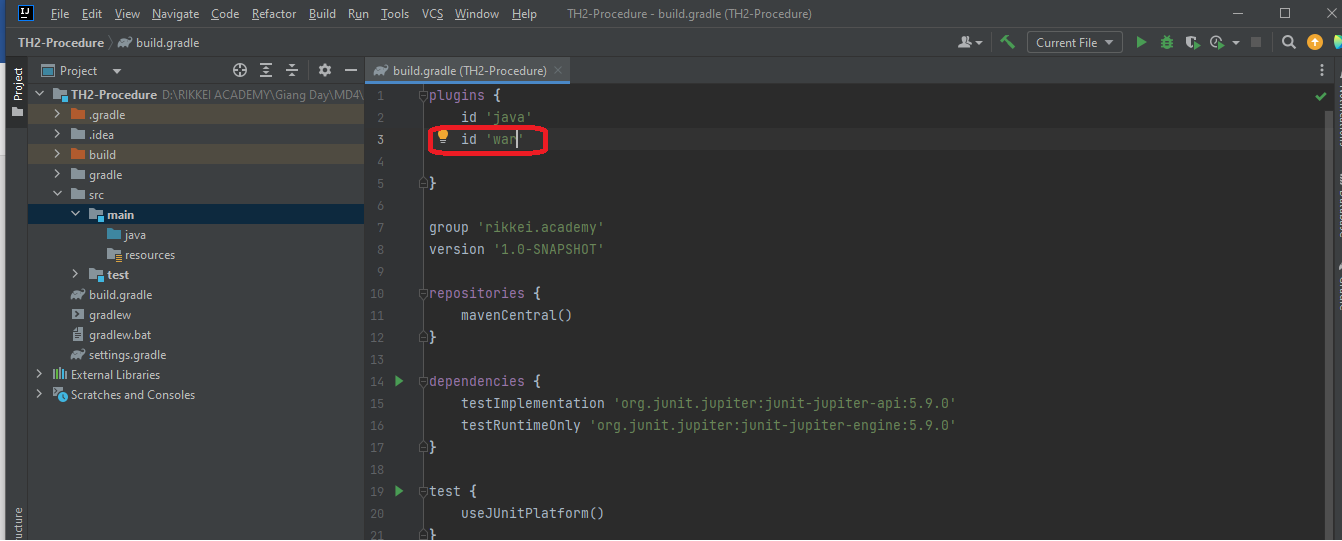
Trong phần này, chúng ta sẽ phát triển tính năng thêm mới một khách hàng bằng Store Procedure.

Hướng dẫn

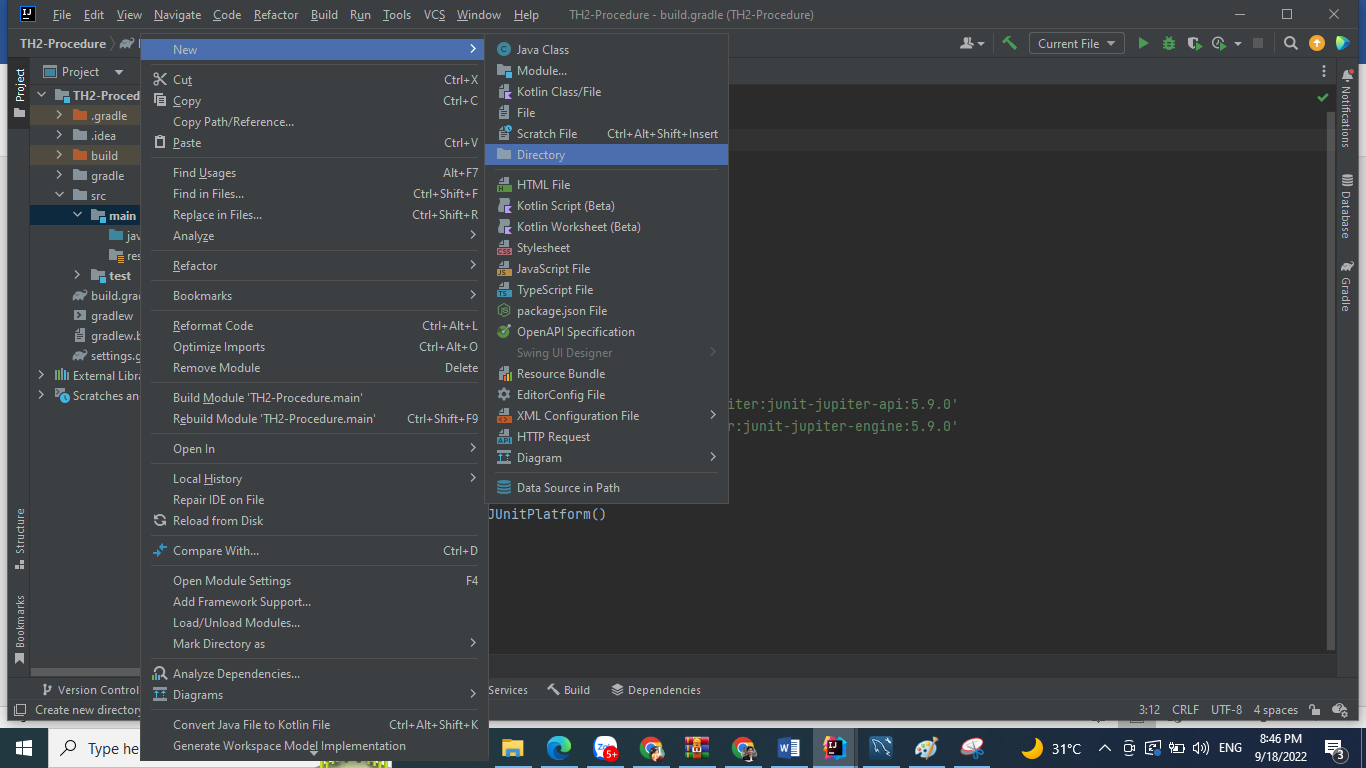
* Ở các bài trước chúng ta vẫn tạo dự án Spring MVC thông qua Jakartar EE – Trong bài này chúng ta sẽ thử tạo một dự án Spring MVC bằng cách tự thêm các cấu hình như hình ảnh bên dưới.

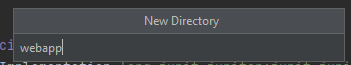


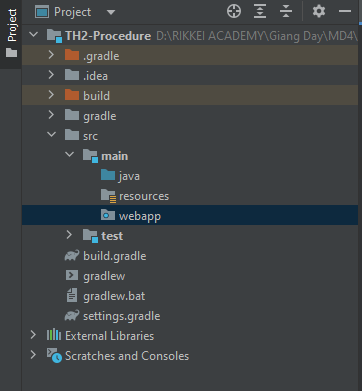
* Bước tiếp theo: thêm id ‘war’ vào build.gradle => Để tạo đường chạy xuống webapp (tạo từ thư mục main)



* Tạo thư mục **webapp** từ thư mục **main**



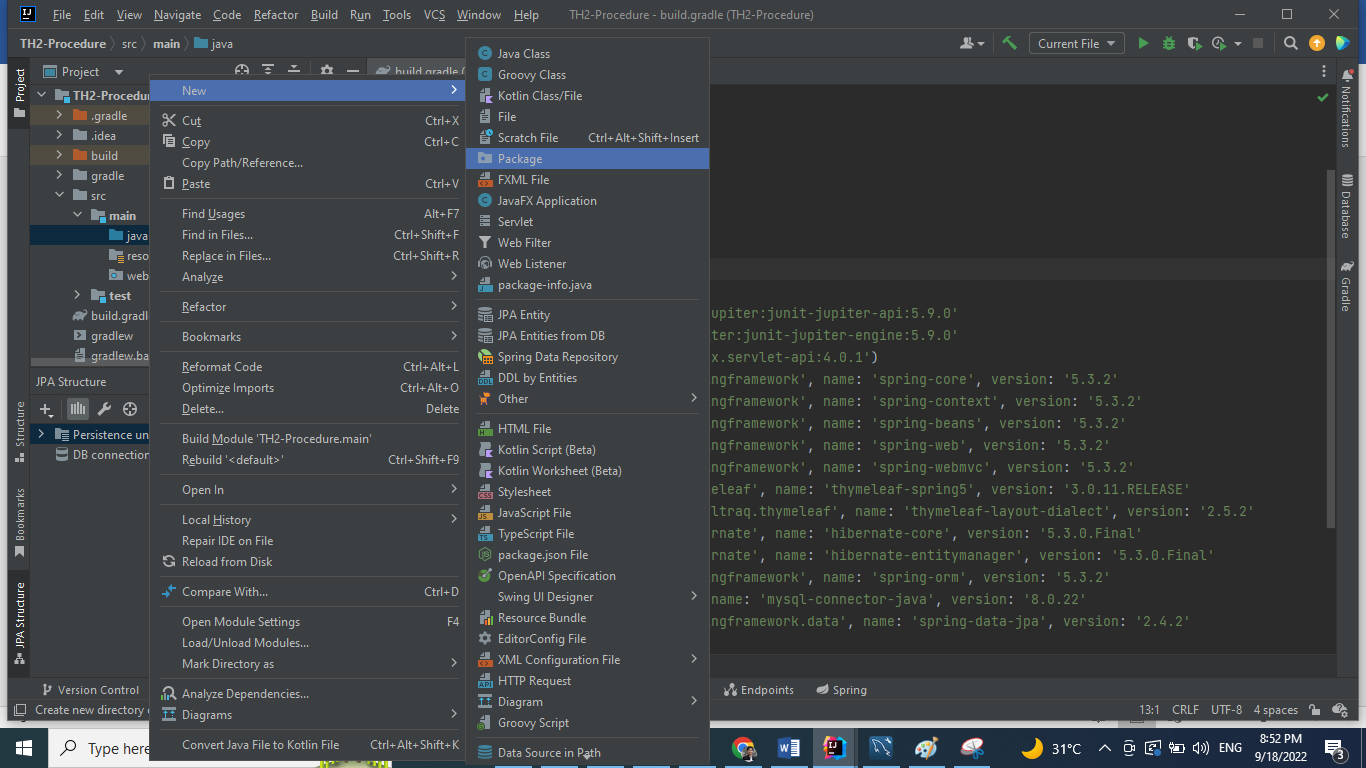




* **import**: Thư viện vào **buil.gradle:**

compileOnly('javax.servlet:javax.servlet-api:4.0.1')  
implementation group: 'org.springframework', name: 'spring-core', version: '5.3.2'  
implementation group: 'org.springframework', name: 'spring-context', version: '5.3.2'  
implementation group: 'org.springframework', name: 'spring-beans', version: '5.3.2'  
implementation group: 'org.springframework', name: 'spring-web', version: '5.3.2'  
implementation group: 'org.springframework', name: 'spring-webmvc', version: '5.3.2'  
implementation group: 'org.thymeleaf', name: 'thymeleaf-spring5', version: '3.0.11.RELEASE'  
implementation group: 'nz.net.ultraq.thymeleaf', name: 'thymeleaf-layout-dialect', version: '2.5.2'  
implementation group: 'org.hibernate', name: 'hibernate-core', version: '5.3.0.Final'  
implementation group: 'org.hibernate', name: 'hibernate-entitymanager', version: '5.3.0.Final'  
implementation group: 'org.springframework', name: 'spring-orm', version: '5.3.2'  
implementation group: 'mysql', name: 'mysql-connector-java', version: '8.0.22'  
implementation group: 'org.springframework.data', name: 'spring-data-jpa', version: '2.4.2'

* Tạo package: **rikkei.academy** từ thư mục **java:**



* Tạo package: config => Tạo 2 class: AppInit và AppConfig trong package này:
* **AppInit:** *Suối nguồn của luồng chạy trong dự án Spring MVC (Mr. ChínhNĐ- RA: 2022)*

package rikkei.academy.config;  
  
import org.springframework.web.filter.CharacterEncodingFilter;  
import org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;  
  
import javax.servlet.Filter;  
  
public class AppInit extends AbstractAnnotationConfigDispatcherServletInitializer {  
 @Override  
 protected Class<?>[] getRootConfigClasses() {  
 return new Class[]{AppConfig.class};  
 }  
  
 @Override  
 protected Class<?>[] getServletConfigClasses() {  
 return new Class[0];  
 }  
  
 @Override  
 protected String[] getServletMappings() {  
 return new String[]{"/"};  
 }  
 @Override  
 protected Filter[] getServletFilters() {  
 CharacterEncodingFilter filter = new CharacterEncodingFilter();  
 filter.setForceEncoding(true);  
 filter.setEncoding("UTF-8");  
 return new Filter[]{filter};  
 }  
}

* **AppConfig:**

package rikkei.academy.config;  
  
import org.springframework.beans.BeansException;  
import org.springframework.beans.factory.annotation.Qualifier;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.ApplicationContextAware;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.ComponentScan;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.jdbc.datasource.DriverManagerDataSource;  
import org.springframework.orm.jpa.JpaTransactionManager;  
import org.springframework.orm.jpa.JpaVendorAdapter;  
import org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean;  
import org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter;  
import org.springframework.transaction.PlatformTransactionManager;  
import org.springframework.transaction.annotation.EnableTransactionManagement;  
import org.springframework.web.servlet.config.annotation.EnableWebMvc;  
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;  
import org.thymeleaf.spring5.SpringTemplateEngine;  
import org.thymeleaf.spring5.templateresolver.SpringResourceTemplateResolver;  
import org.thymeleaf.spring5.view.ThymeleafViewResolver;  
import org.thymeleaf.templatemode.TemplateMode;  
  
  
import javax.persistence.EntityManager;  
import javax.persistence.EntityManagerFactory;  
import javax.sql.DataSource;  
import java.util.Properties;  
  
@Configuration  
@EnableWebMvc  
@EnableTransactionManagement  
@ComponentScan("rikkei.academy.controller")  
  
public class AppConfig implements WebMvcConfigurer, ApplicationContextAware {  
 private ApplicationContext applicationContext;  
  
 @Override  
 public void setApplicationContext(ApplicationContext applicationContext) throws BeansException {  
 this.applicationContext = applicationContext;  
 }  
  
 //Cấu hình Thymleaf  
 @Bean  
 public SpringResourceTemplateResolver templateResolver() {  
 SpringResourceTemplateResolver templateResolver = new SpringResourceTemplateResolver();  
 templateResolver.setApplicationContext(applicationContext);  
 templateResolver.setPrefix("/WEB-INF/views");  
 templateResolver.setSuffix(".html");  
 templateResolver.setTemplateMode(TemplateMode.*HTML*);  
 templateResolver.setCharacterEncoding("UTF-8");  
 return templateResolver;  
 }  
  
 @Bean  
 public SpringTemplateEngine templateEngine() {  
 SpringTemplateEngine templateEngine = new SpringTemplateEngine();  
 templateEngine.setTemplateResolver(templateResolver());  
 return templateEngine;  
 }  
  
 @Bean  
 public ThymeleafViewResolver viewResolver() {  
 ThymeleafViewResolver viewResolver = new ThymeleafViewResolver();  
 viewResolver.setTemplateEngine(templateEngine());  
 viewResolver.setCharacterEncoding("UTF-8");  
 viewResolver.setContentType("UTF-8");  
 return viewResolver;  
 }  
  
 //Cấu hình JPA  
 @Bean  
 @Qualifier(value = "entityManager")  
 public EntityManager entityManager(EntityManagerFactory entityManagerFactory) {  
 return entityManagerFactory.createEntityManager();  
 }  
  
 @Bean  
 public LocalContainerEntityManagerFactoryBean entityManagerFactory() {  
 LocalContainerEntityManagerFactoryBean em = new LocalContainerEntityManagerFactoryBean();  
 em.setDataSource(dataSource());  
 em.setPackagesToScan("rikkei.academy.model");  
  
 JpaVendorAdapter vendorAdapter = new HibernateJpaVendorAdapter();  
 em.setJpaVendorAdapter(vendorAdapter);  
 em.setJpaProperties(additionalProperties());  
 return em;  
 }  
  
 @Bean  
 public DataSource dataSource() {  
 DriverManagerDataSource dataSource = new DriverManagerDataSource();  
 dataSource.setDriverClassName("com.mysql.cj.jdbc.Driver");  
 dataSource.setUrl("jdbc:mysql://localhost:3306/th2\_producer");  
 dataSource.setUsername("root");  
 dataSource.setPassword("Minhtri29092014");  
 return dataSource;  
 }  
  
 @Bean  
 public PlatformTransactionManager transactionManager(EntityManagerFactory emf) {  
 JpaTransactionManager transactionManager = new JpaTransactionManager();  
 transactionManager.setEntityManagerFactory(emf);  
 return transactionManager;  
 }  
  
 public Properties additionalProperties() {  
 Properties properties = new Properties();  
 properties.setProperty("hibernate.hbm2ddl.auto", "update");  
 properties.setProperty("hibernate.dialect", "org.hibernate.dialect.MySQL5Dialect");  
 return properties;  
 }  
  
}

* Tạo package: **model** => Tạo class: **Customer:**

package rikkei.academy.model;  
  
import javax.persistence.\*;  
  
@Entity  
@Table(name = "customers")  
public class Customer {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
 private String firstName;  
 private String lastName;  
  
 public Customer() {}  
  
 public Customer(String firstName, String lastName) {  
 this.firstName = firstName;  
 this.lastName = lastName;  
 }  
  
 @Override  
 public String toString() {  
 return String.*format*("Customer[id=%d, firstName='%s', lastName='%s']", id, firstName, lastName);  
 }  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public void setFirstName(String firstName) {  
 this.firstName = firstName;  
 }  
  
 public String getLastName() {  
 return lastName;  
 }  
  
 public void setLastName(String lastName) {  
 this.lastName = lastName;  
 }  
  
}

* Tạo thêm package: **repository** => Tạo interface: **ICustomerRepository** (Tầng này có nhiệm vụ tạo các câu query để tương tác với **Database**)

package rikkei.academy.repository;  
  
import rikkei.academy.model.Customer;  
  
public interface ICustomerRepository {  
 boolean insertWithStoredProcedure(Customer customer);  
}

* Tạo class: **CustomerRepositoryIMPL** triển khai interface **ICustomerRepository**:

package rikkei.academy.repository;  
  
import rikkei.academy.model.Customer;  
  
import javax.persistence.EntityManager;  
import javax.persistence.PersistenceContext;  
import javax.persistence.Query;  
import javax.transaction.Transactional;  
  
@Transactional  
public class CustomerRepositoryIMPL implements ICustomerRepository{  
 @PersistenceContext  
 private EntityManager entityManager;  
 @Override  
 public boolean insertWithStoredProcedure(Customer customer) {  
 String sql = "CALL Insert\_Customer(:firstName, :lastName)";  
 Query query = entityManager.createNativeQuery(sql);  
 query.setParameter("firstName", customer.getFirstName());  
 query.setParameter("lastName", customer.getLastName());  
 return query.executeUpdate() == 0;  
 }  
}

* Tạo package: service => Tạo interface: **ICustomerService**

package rikkei.academy.service;  
  
import rikkei.academy.model.Customer;  
  
public interface ICustomerService {  
 boolean insertWithStoredProcedure(Customer customer);  
}

* Tạo class: **CustomerServiceIMPL** triển khai interface:

**ICustomerService:**

package rikkei.academy.service;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import rikkei.academy.model.Customer;  
import rikkei.academy.repository.ICustomerRepository;  
@Service  
public class CustomerServiceIMPL implements ICustomerService{  
 @Autowired  
 private ICustomerRepository customerRepository;  
  
 @Override  
 public boolean insertWithStoredProcedure(Customer customer) {  
 return customerRepository.insertWithStoredProcedure(customer);  
 }  
}

* Thêm đoạn code sau vào: **AppConfig**:

@Bean  
public ICustomerRepository customerRepository() {  
 return new CustomerRepositoryIMPL();  
}  
  
@Bean  
public ICustomerService customerService() {  
 return new CustomerServiceIMPL();  
}

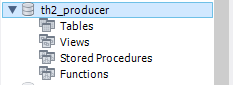
* Tạo package: controller => Tạo class: **CustomerController**

package rikkei.academy.controller;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.ModelAttribute;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.servlet.ModelAndView;  
import rikkei.academy.model.Customer;  
import rikkei.academy.service.ICustomerService;  
  
@Controller  
public class CustomerController {  
 @Autowired  
 private ICustomerService customerService;  
  
 @GetMapping("/create-customer")  
 public ModelAndView showCreateForm() {  
 ModelAndView modelAndView = new ModelAndView("/customer/create");  
 modelAndView.addObject("customer", new Customer());  
 return modelAndView;  
 }  
  
 @PostMapping("/create-customer")  
 public ModelAndView saveCustomer(@ModelAttribute("customer") Customer customer) {  
 customerService.insertWithStoredProcedure(customer);  
 ModelAndView modelAndView = new ModelAndView("/customer/create");  
 modelAndView.addObject("customer", new Customer());  
 modelAndView.addObject("message", "New customer created successfully");  
 return modelAndView;  
 }  
}

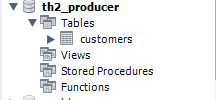
* **Tạo tầng views:** => Tạo thư mục:

WEB-INF/views/customers => Tạo file: **create.html**

* Tạo database: **th2\_producer**  trong MYSQL



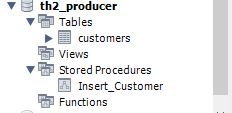
* **Add Tomcat và chạy chương trình => Table: customers đã được tạo**



* Tạo **procedure** bằng dòng lệnh sau:

DELIMITER //  
CREATE PROCEDURE *Insert\_Customer*(IN first\_name VARCHAR(255), IN last\_name VARCHAR(255))  
BEGIN  
 INSERT INTO customers(firstName,lastName) VALUES (first\_name,last\_name);  
END//  
DELIMITER ;

* Một procedures: **Isert\_Customer** đã được tạo:



* **Truy cập:** [***http://localhost:8080/create-customer***](http://localhost:8080/create-customer)
* **Create và kiểm tra kết quả trong database**

**Link git tham khảo:**

**https://github.com/nguyendongminhtri/MD4-Bai6-TH2-Call-Procedures-JPA.git**