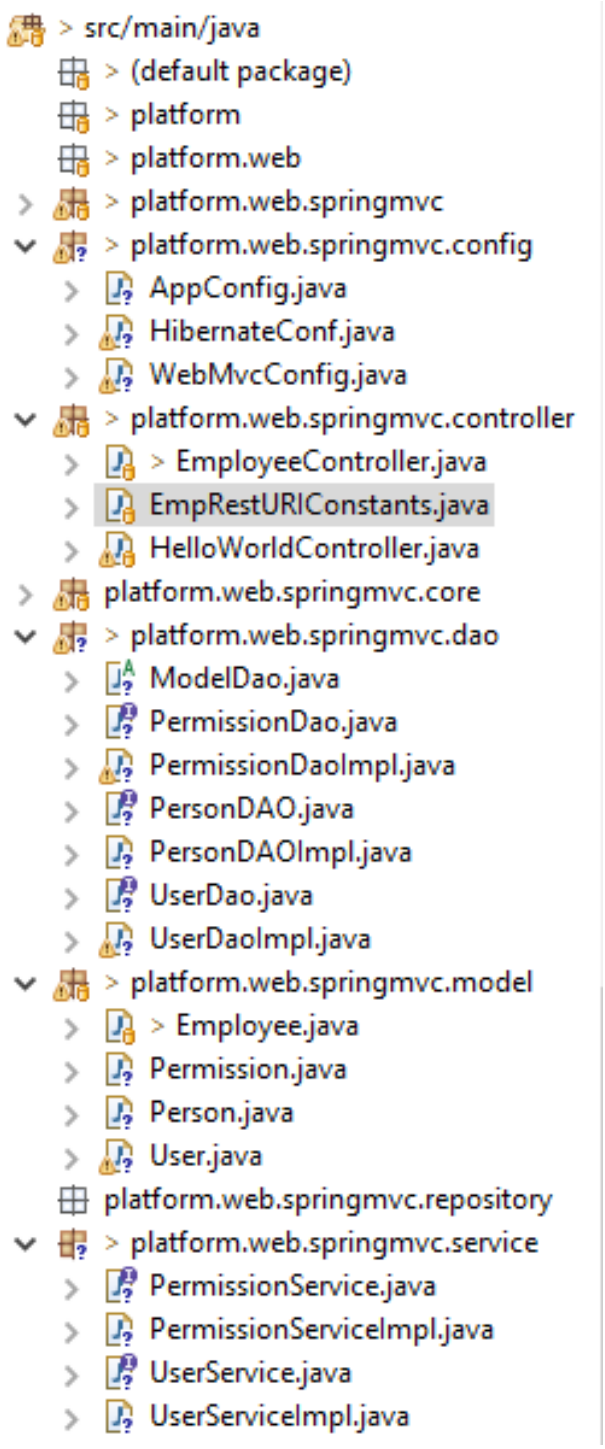
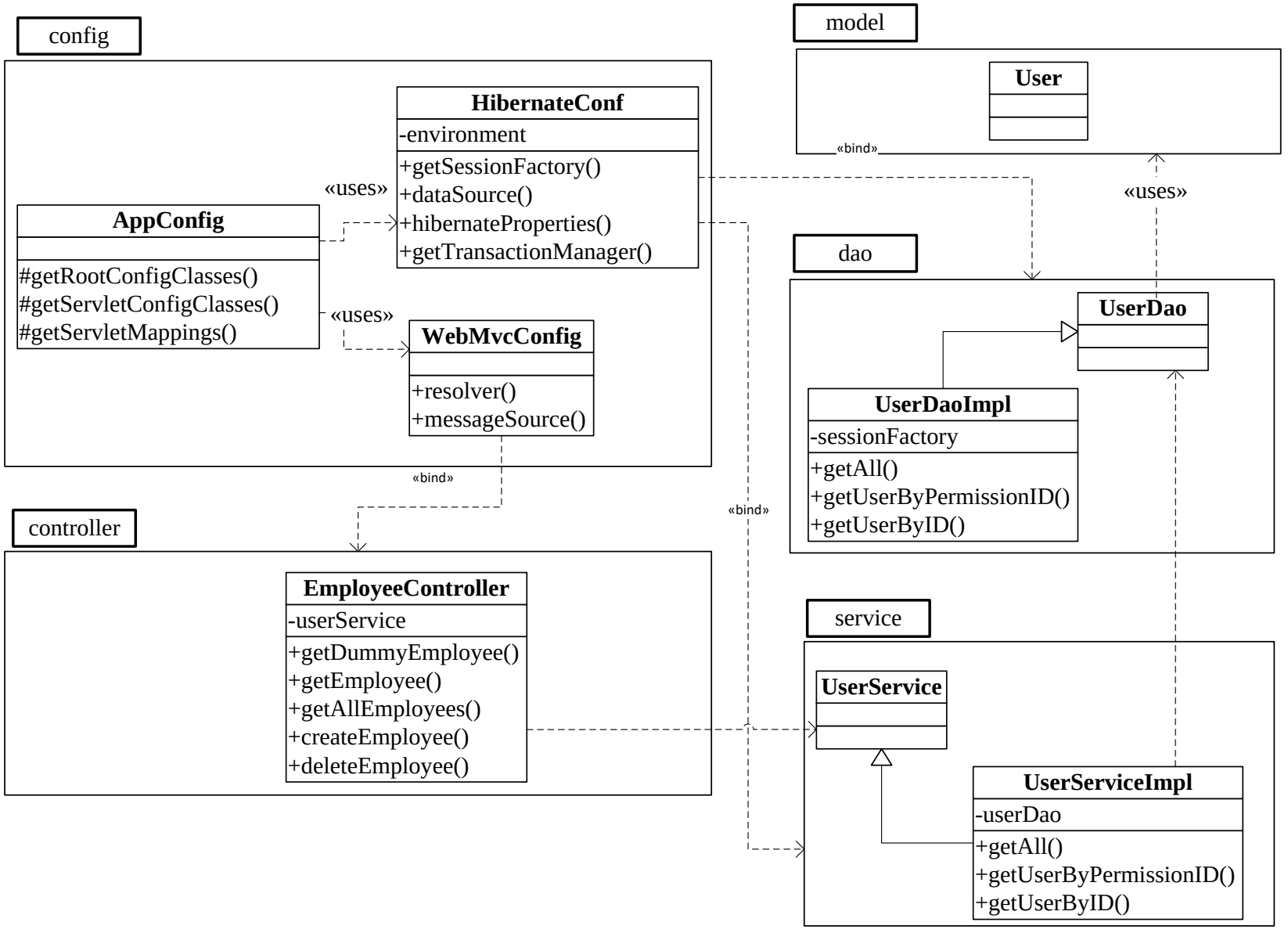
I. Spring MVC 5 – Hibernate 5 and MySQL Tutorial

1. Project Structure



2. Class Diagram



3. Maven config

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">

	<modelVersion>4.0.0</modelVersion>
	<groupId>platform.web</groupId>
	<artifactId>springmvc</artifactId>
	<packaging>war</packaging>
	<version>1.0</version>
	<name>springmvc Maven Webapp</name>
	<url>http://maven.apache.org</url>

	<properties>
		<org.slf4j-version>1.7.5</org.slf4j-version>
		<org.aspectj-version>1.7.4</org.aspectj-version>
		<spring.version>5.2.0.RELEASE</spring.version><!-- 5.1.0.RELEASE
5.2.3.RELEASE-->
	  <hibernate.version>5.2.11.Final</hibernate.version><!-- 5.3.5.Final
5.4.14.Final -->
	  <hibernate.validator>5.4.1.Final</hibernate.validator>
	  <failOnMissingWebXml>false</failOnMissingWebXml>
		<c3p0.version>0.9.5.2</c3p0.version>
		<jstl.version>1.2.1</jstl.version>
		<tld.version>1.1.2</tld.version>
		<servlets.version>3.1.0</servlets.version>
		<jsp.version>2.3.1</jsp.version>
		<hsqldb.version>1.8.0.10</hsqldb.version>
	</properties>

	<dependencies>
		<!-- <dependency>
			<groupId>tinyWorld.com</groupId>
			<artifactId>core</artifactId>
			<version>1.0</version>
		</dependency>

		<dependency>
			<groupId>tinyWorld.com</groupId>
			<artifactId>device-api</artifactId>
			<version>1.0-SNAPSHOT</version>
		</dependency>
		<dependency>
		    <groupId>org.primefaces</groupId>
		    <artifactId>primefaces</artifactId>
		    <version>${primefaces-version}</version>
	  </dependency> -->

		<dependency>
			<groupId>org.springframework</groupId>
			<artifactId>spring-core</artifactId>
```

```
			<version>${spring.version}</version>
		</dependency>
		<dependency>
			<groupId>org.springframework</groupId>
			<artifactId>spring-context</artifactId>
			<version>${spring.version}</version>
		</dependency>
		<dependency>
			<groupId>org.springframework</groupId>
			<artifactId>spring-aop</artifactId>
			<version>${spring.version}</version>
		</dependency>
		<dependency>
			<groupId>org.springframework</groupId>
			<artifactId>spring-webmvc</artifactId>
			<version>${spring.version}</version>
		</dependency>
		<dependency>
			<groupId>org.springframework</groupId>
			<artifactId>spring-web</artifactId>
			<version>${spring.version}</version>
		</dependency>
		<!-- https://mvnrepository.com/artifact/org.springframework/spring-tx --
>
		<dependency>
		    <groupId>org.springframework</groupId>
		    <artifactId>spring-tx</artifactId>
		    <version>${spring.version}</version>
		</dependency>
		<!-- https://mvnrepository.com/artifact/org.springframework/spring-orm -
->
		<dependency>
		    <groupId>org.springframework</groupId>
		    <artifactId>spring-orm</artifactId>
		    <version>${spring.version}</version>
		</dependency>

		<!-- Spring Security Config -->
		<dependency>
			<groupId>org.springframework.security</groupId>
			<artifactId>spring-security-config</artifactId>
			<version>${spring.version}</version>
		</dependency>

		<!-- Spring Security Web -->
		<dependency>
			<groupId>org.springframework.security</groupId>
			<artifactId>spring-security-web</artifactId>
			<version>${spring.version}</version>
		</dependency>
		<!--
=============================================================================
====================== -->
		<!-- Dependencies cho việc tạo và call rest-service -->
```

```
		<!--
https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind -->
		<dependency>
		    <groupId>com.fasterxml.jackson.core</groupId>
		    <artifactId>jackson-databind</artifactId>
		    <version>2.10.3</version>
		</dependency>

		<!-- AspectJ -->
		<dependency>
			<groupId>org.aspectj</groupId>
			<artifactId>aspectjrt</artifactId>
			<version>${org.aspectj-version}</version>
		</dependency>

		<!-- Logging -->
		<dependency>
			<groupId>org.slf4j</groupId>
			<artifactId>slf4j-api</artifactId>
			<version>${org.slf4j-version}</version>
		</dependency>
		<dependency>
			<groupId>org.slf4j</groupId>
			<artifactId>jcl-over-slf4j</artifactId>
			<version>${org.slf4j-version}</version>
			<scope>runtime</scope>
		</dependency>
		<dependency>
			<groupId>org.slf4j</groupId>
			<artifactId>slf4j-log4j12</artifactId>
			<version>${org.slf4j-version}</version>
			<scope>runtime</scope>
		</dependency>
		<dependency>
			<groupId>log4j</groupId>
			<artifactId>log4j</artifactId>
			<version>1.2.15</version>
			<exclusions>
				<exclusion>
					<groupId>javax.mail</groupId>
					<artifactId>mail</artifactId>
				</exclusion>
				<exclusion>
					<groupId>javax.jms</groupId>
					<artifactId>jms</artifactId>
				</exclusion>
				<exclusion>
					<groupId>com.sun.jdmk</groupId>
					<artifactId>jmxtools</artifactId>
				</exclusion>
				<exclusion>
					<groupId>com.sun.jmx</groupId>
					<artifactId>jmxri</artifactId>
				</exclusion>
			</exclusions>
```

```
			<scope>runtime</scope>
		</dependency>

		<!-- @Inject -->
		<dependency>
			<groupId>javax.inject</groupId>
			<artifactId>javax.inject</artifactId>
			<version>1</version>
		</dependency>

		<!-- Servlet -->
		<dependency>
			<groupId>javax.servlet</groupId>
			<artifactId>servlet-api</artifactId>
			<version>2.5</version>
			<scope>provided</scope>
		</dependency>
		<dependency>
			<groupId>javax.servlet.jsp</groupId>
			<artifactId>jsp-api</artifactId>
			<version>2.1</version>
			<scope>provided</scope>
		</dependency>
		<dependency>
			<groupId>javax.servlet</groupId>
			<artifactId>jstl</artifactId>
			<version>1.2</version>
		</dependency>

		<!--
================================================================================
======================== -->
		<!-- Depedencies for RestClient -->
		<!-- https://mvnrepository.com/artifact/org.codehaus.jackson/jackson-
mapper-asl -->
		<dependency>
		    <groupId>org.codehaus.jackson</groupId>
		    <artifactId>jackson-mapper-asl</artifactId>
		    <version>1.9.13</version>
		</dependency>

		<!--
================================================================================
======================== -->
		<!-- Start: Dependencies for DB Process -->
		<!-- https://mvnrepository.com/artifact/org.springframework.data/spring-
data-jpa -->
		<dependency>
		    <groupId>org.springframework.data</groupId>
		    <artifactId>spring-data-jpa</artifactId>
		    <version>2.2.6.RELEASE</version>
		</dependency>
		<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-
entitymanager -->
		<dependency>
```

```
                <groupId>org.hibernate</groupId>
                <artifactId>hibernate-entitymanager</artifactId>
                <version>${hibernate.version}</version>
            </dependency>
            <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
            <dependency>
                <groupId>org.hibernate</groupId>
                <artifactId>hibernate-core</artifactId>
                <version>${hibernate.version}</version>
            </dependency>
            <dependency>
               <groupId>org.hibernate</groupId>
               <artifactId>hibernate-validator</artifactId>
               <version>${hibernate.validator}</version>
         </dependency>
              <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
              <dependency>
                <groupId>mysql</groupId>
                <artifactId>mysql-connector-java</artifactId>
                <!-- <version>8.0.11</version> -->
                <!-- <version>5.1.9</version> -->
                <version>5.1.47</version>
            </dependency>
            <!-- https://mvnrepository.com/artifact/commons-dbcp/commons-dbcp -->
            <dependency>
                <groupId>commons-dbcp</groupId>
                <artifactId>commons-dbcp</artifactId>
                <version>1.4</version>
            </dependency>
            <dependency>
              <groupId>javax.xml.bind</groupId>
              <artifactId>jaxb-api</artifactId>
              <version>2.3.0</version>
            </dependency>
            <!-- End Dependencies for DB Process -->
            <!--
================================================================================
======================== -->

            <!-- Test -->
            <dependency>
                  <groupId>junit</groupId>
                  <artifactId>junit</artifactId>
                  <version>4.7</version>
                  <scope>test</scope>
            </dependency>
      </dependencies>

      <build>
            <finalName>springmvc</finalName>
            <sourceDirectory>src/main/java</sourceDirectory>
            <resources>
                  <resource>
                        <directory>src/main/resources</directory>
                  </resource>
```

```
            </resources>
            <plugins>

                  <plugin>
                        <groupId>org.apache.maven.plugins</groupId>
                        <artifactId>maven-war-plugin</artifactId>
                        <version>3.2.2</version>
                  </plugin>
                  <plugin>
               <artifactId>maven-compiler-plugin</artifactId>
               <version>3.5.1</version>
               <configuration>
                   <source>1.8</source>
                   <target>1.8</target>
               </configuration>
            </plugin>
                  <plugin>
                        <groupId>org.eclipse.jetty</groupId>
                        <artifactId>jetty-maven-plugin</artifactId>
                        <version>9.4.14.v20181114</version>
                        <configuration>
                              <scanIntervalSeconds>10</scanIntervalSeconds>
                              <webApp>
                                    <contextPath>/</contextPath>
                              </webApp>
                              <httpConnector>
                                    <port>8080</port>
                              </httpConnector>
                        </configuration>
                  </plugin>
                  <!-- Embedded Apache Tomcat required for testing war -->
                  <plugin>
                        <groupId>org.apache.tomcat.maven</groupId>
                        <artifactId>tomcat-maven-plugin</artifactId>
                        <version>2.2</version>
                        <configuration>
                              <path>/</path>
                        </configuration>
                  </plugin>
            </plugins>
      </build>
</project>
```

4. Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
            xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
            id="WebApp_ID" version="3.0">
      <display-name>Archetype Created Web Application</display-name>
```

```
    <!-- Context Configuration locations for Spring XML files -->
  <context-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>
          classpath:/Beans.xml
      </param-value>
  </context-param>
    <context-param>
          <param-name>contextConfigLocation</param-name>
          <param-value>/WEB-INF/dispatcher-servlet.xml</param-value>
    </context-param>

    <!-- <servlet>
          <servlet-name>dispatcher</servlet-name>
          <servlet-class>
                org.springframework.web.servlet.DispatcherServlet
          </servlet-class>
          <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
          <servlet-name>dispatcher</servlet-name>
          <url-pattern>/</url-pattern>
    </servlet-mapping> -->

    <!-- <listener>
          <listener-class>
                org.springframework.web.context.ContextLoaderListener
          </listener-class>
    </listener> -->
</web-app>
```

5. DispatcherServlet Configuration

Spring cung cấp ***SpringServletContainerInitializer*** để xử lý các lớp ***WebApplicationInitializer***. Lớp `AbstractAnnotationConfigDispatcherServletInitializer` thực thi `WebMvcConfigurer` đây là lớp thực thi `WebApplicationInitializer`. Nó đăng ký ContextLoaderListener (tùy chọn) và một DispatcherServlet cho phép chúng ta dễ dang thêm vào các configuration classes để lạp các classes cũng như áp dụng các filters vào DispatcherServlet và cung cấp servlet mapping.

Khi sử dụng cấu hình này, ta không cần thêm các đoạng đăng ký servlet vào web.xml như các ví dụ ở trên.

```
package platform.web.springmvc.config;

import
org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInit
ializer;

/*@Configuration
```

```
@ComponentScan("platform.web.springmvc")*/
public class AppConfig extends AbstractAnnotationConfigDispatcherServletInitializer {
      @Override
    protected Class < ? > [] getRootConfigClasses() {
        return new Class[] {
              HibernateConf.class
        };
    }

      @Override
    protected Class < ? > [] getServletConfigClasses() {
        return new Class[] {
              WebMvcConfig.class
        };
    }

      @Override
      protected String[] getServletMappings() {
            return new String[] { "/" };
      }

      /*
       * @Bean RestTemplate restTemplate() { RestTemplate restTemplate = new
       * RestTemplate(); MappingJacksonHttpMessageConverter converter = new
       * MappingJacksonHttpMessageConverter(); converter.setObjectMapper(new
       * ObjectMapper()); restTemplate.getMessageConverters().add(converter); return
       * restTemplate; }
       */
}
```

6. Spring WebMVC Configuration

```
package platform.web.springmvc.config;

import org.springframework.context.MessageSource;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.support.ResourceBundleMessageSource;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
import org.springframework.web.servlet.view.InternalResourceViewResolver;
import org.springframework.web.servlet.view.JstlView;

@Configuration
@EnableWebMvc
@ComponentScan(basePackages = { "platform.web.springmvc"})
public class WebMvcConfig implements WebMvcConfigurer {

  @Bean
   public InternalResourceViewResolver resolver() {
      InternalResourceViewResolver resolver = new InternalResourceViewResolver();
      resolver.setViewClass(JstlView.class);
```

```
        resolver.setPrefix("/");
        resolver.setSuffix(".jsp");
        return resolver;
    }

    @Bean
    public MessageSource messageSource() {
        ResourceBundleMessageSource source = new ResourceBundleMessageSource();
        source.setBasename("messages");
        return source;
    }

    /*public Validator getValidator() {
        LocalValidatorFactoryBean validator = new LocalValidatorFactoryBean();
        validator.setValidationMessageSource(messageSource());
        return validator;
    }*/
}
```

Lớp này định nghĩa các options cho việc tùy chỉnh hoặc thêm vào trong default Spring MVC Configuration, nó được kích hoạt thông qua @EnableWebMvc.

@EnableWebMvc:kích hoạt default Spring MVC Configuration và đăng ký các components hạ tầng mong muốn cho DispatcherServlet.

@Configuration: chỉ định một lớp khai báo một hay nhiều @Bean methods và có thể được xử lý bởi Spring Container để tạo ra bean definitions và service requests cho các bean này tại thời điểm runtime.

@ComponentScan: sử dụng để chỉ định base-packages để scan. Bất kỳ class nào được đánh dấu với @Component và @Configuration sẽ được quét.

InternalResourceViewResolver: hỗ trợ việc ánh xạc các logical view tới các file cụ thể trong thư mục được chỉ định.

LocalValidatorFactoryBean đưa một `javax.validation.ValidationFactory` và thể hiện nó trong Spring Validator interface thông qua JSR-303 validator interface và ValidatorFacotry interface.

7. Hibernate Configuration

```
package platform.web.springmvc.config;

import java.util.Properties;

import javax.sql.DataSource;
```

```
import org.hibernate.SessionFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.PropertySource;
import org.springframework.core.env.Environment;
import org.springframework.jdbc.datasource.DriverManagerDataSource;
import org.springframework.orm.hibernate5.HibernateTransactionManager;
import org.springframework.orm.hibernate5.LocalSessionFactoryBean;
import org.springframework.transaction.annotation.EnableTransactionManagement;

@Configuration
@EnableTransactionManagement
@PropertySource("classpath:hibernateconfig.properties")
@ComponentScan(basePackages = { "platform.web.springmvc" })
public class HibernateConf {
	// Using environment bean to read property configuration from
	// hibernateconfig.properties file
	@Autowired
	private Environment environment;

	@Bean
	public LocalSessionFactoryBean getSessionFactory() {
		System.out.println("\n\n\n\n>>>>>> HibernateConf.sessionFactory");
		LocalSessionFactoryBean sessionFactory = new LocalSessionFactoryBean();
		sessionFactory.setDataSource(dataSource());
		sessionFactory.setPackagesToScan(new String[] { "platform.web.springmvc"
});
		sessionFactory.setHibernateProperties(hibernateProperties());

		System.out.println(
				"-----> HibernateConfiguration--> sessionFactory-->
localSesFactory:" + sessionFactory.toString());
		/*System.out.println("-----> HibernateConfiguration--> sessionFactory--
>SessonFactory:"
				+ sessionFactory.getObject().toString());*/
		System.out.println(">>>>>>End sessionFactory\n\n\n\n");

		return sessionFactory;
	}

	@Bean
	public DataSource dataSource() {
		System.out.println(">>>>>> HibernateConf.dataSource");
		// BasicDataSource dataSource = new BasicDataSource();
		DriverManagerDataSource dataSource = new DriverManagerDataSource();

	dataSource.setDriverClassName(environment.getRequiredProperty("jdbc.driverClas
sName"));
		dataSource.setUrl(environment.getRequiredProperty("jdbc.url"));

	dataSource.setUsername(environment.getRequiredProperty("jdbc.username"));

	dataSource.setPassword(environment.getRequiredProperty("jdbc.password"));
```

```
            System.out.println(" - jdbc.driverClassName:" +
environment.getRequiredProperty("jdbc.driverClassName"));
            System.out.println(" - jdbc.url:" +
environment.getRequiredProperty("jdbc.url"));
            System.out.println(" - jdbc.username:" +
environment.getRequiredProperty("jdbc.username"));
            System.out.println(" - jdbc.password: " +
environment.getRequiredProperty("jdbc.password"));
            System.out.println(">>>>>>End dataSource");
            return dataSource;
      }

      private final Properties hibernateProperties() {
            Properties properties = new Properties();
            properties.put("hibernate.dialect",
environment.getRequiredProperty("hibernate.dialect"));
            properties.put("hibernate.show_sql",
environment.getRequiredProperty("hibernate.show_sql"));
            properties.put("hibernate.format_sql",
environment.getRequiredProperty("hibernate.format_sql"));
            properties.put("current_session_context_class",
environment.getProperty("current_session_context_class"));
            System.out.println(" - hibernate.dialect:" +
environment.getRequiredProperty("hibernate.dialect"));
            System.out.println(" - hibernate.show_sql:" +
environment.getRequiredProperty("hibernate.show_sql"));
            System.out.println(" - hibernate.format_sql" +
environment.getRequiredProperty("hibernate.format_sql"));
            System.out.println(
                        " - current_session_context_class:" +
environment.getProperty("current_session_context_class"));
            return properties;
      }

      /*@Autowired
      @Bean(name = "transactionManager")
      public HibernateTransactionManager transactionManager(SessionFactory s) {
            System.out.println(">>>>>> HibernateConfiguration.transactionManager --
>SessionFactory: " + s.toString());
            HibernateTransactionManager txManager = new
HibernateTransactionManager(s); // txManager.setSessionFactory(s);
            System.out.println("\t\t>>>>>> sessionFactory: " + s.toString());
            System.out.println(">>>>>>End transactionManager\n\n\n\n");
            return txManager;
      }*/

      @Bean
      public HibernateTransactionManager getTransactionManager() {
            System.out.println("\n\n\n\n>>>>>>
HibernateConfiguration.getTransactionManager -->SessionFactory: ");
            HibernateTransactionManager transactionManager = new
HibernateTransactionManager();
            SessionFactory sessionFactory = getSessionFactory().getObject();
```

```
            System.out.println("\t\t>>>>>> sessionFactory: " +
sessionFactory.toString());
            transactionManager.setSessionFactory(sessionFactory);
            System.out.println(">>>>>> End getTransactionManager");
            return transactionManager;
      }
}
```

- **LocalSessionFactoryBean** tạo ra một Hibernate **SessionFactory**. Đây là cách hữu ích để tạo ra một shared Hibernate SessionFactory trong Spring application context.
- **EnableTransactionManagement** kích hoạt annotation-driven transaction management của Spring.
- **HibernateTransactionManager** rang buộc một Hibernate Session từ một factory cụ thể vào thread, có khả năng cho phép một thread-bound Session trên một factory. Transaction manager này là phù hợp cho ứng dụng sử dụng một single Hibernate SessionFactory cho transactional data access, nó cũng hỗ trợ direct DataSource truy cập vào trong một transaction i.e. plain JDBC.

❖ **File hibernateconfig.properties**

jdbc.driverClassName = com.mysql.jdbc.Driver
jdbc.url = jdbc:mysql://localhost:3306/myhouse
jdbc.username = root
jdbc.password = 123456
hibernate.dialect = org.hibernate.dialect.MySQLDialect
hibernate.show_sql = true
hibernate.format_sql = false
current_session_context_class=thread

8. Spring Controller and Restful config

```
package platform.web.springmvc.controller;

import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import platform.web.springmvc.model.User;
import platform.web.springmvc.service.UserService;
```

```java
/**
 * Handles requests for the Employee service.
 */
@Controller
public class EmployeeController {
	@Autowired
	private UserService userService;

	private static final Logger logger =
LoggerFactory.getLogger(EmployeeController.class);

	@RequestMapping(value = EmpRestURIConstants.DUMMY_EMP, method =
RequestMethod.GET)
	public @ResponseBody User getDummyEmployee() {
		logger.info("Start getDummyEmployee");
		User user = userService.getUserByID(01);
		System.out.println("\n\n\n\n>>>>>> user: " + user.getUserName());
		return user;
	}

	@RequestMapping(value = EmpRestURIConstants.GET_EMP, method =
RequestMethod.GET)
	public @ResponseBody User getEmployee(@PathVariable("id") int empId) {
		logger.info("Start getEmployee. ID="+empId);

		return userService.getUserByID(empId);
	}

	@RequestMapping(value = EmpRestURIConstants.GET_ALL_EMP, method =
RequestMethod.GET)
	public @ResponseBody List<User> getAllEmployees() {
		logger.info("Start getAllEmployees.");
		return userService.getAll();
	}

	@RequestMapping(value = EmpRestURIConstants.CREATE_EMP, method =
RequestMethod.POST)
	public @ResponseBody User createEmployee(@RequestBody User emp) {
		logger.info("Start createEmployee.");
		userService.insertUser(emp);
		return emp;
	}

	@RequestMapping(value = EmpRestURIConstants.DELETE_EMP, method =
RequestMethod.PUT)
	public @ResponseBody User deleteEmployee(@PathVariable("id") int empId) {
		logger.info("Start deleteEmployee.");
		return userService.getUserByID(01);
	}

}
```

Trong ví dụ này, ta lưu tất cả dữ liệu của user từ database.

- ✓ **@RequestMapping annotation** được sử dụng để ánh xạ request URI tới handler method. Ta có thể chỉ định HTTP method được dùng bởi client application để gọi rest method.
- ✓ **@ResponseBody annotation**: được sử dụng để ánh xạ response object trong response body. Mỗi response object được trả về với handler method, MappingJackson2HttpMessageConverter đưa vào và chuyển đổi nó thành JSON response.
- ✓ **@PathVariable annotation** Đây là cacash đơn giản để lấy data từ trong rest URI và ánh xạ nó vào method argument.
- ✓ **@RequestBody annotation** Được sử dụng để ánh xạ request body JSON data vào trong Employee object, nó được thực hiện bằng MappingJackson2HttpMessageConverter mapping.

9. Service layer

❖ UserService

```
package platform.web.springmvc.service;

import java.util.List;

import platform.web.springmvc.model.User;

public interface UserService {
	public List<User> getAll();

	public List<User> getUserByPermissionID(int perId);

	public User getUserByID(int id);

	public List<User> getUserByAddress(String address);

	public User login(String userName, String pass);

	public List<User> getAllUserWithNativeSQL();

	public void insertUser(User user);
}
```

❖ UserServiceImpl

```
package platform.web.springmvc.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
```

```java
import org.springframework.transaction.annotation.Transactional;

import platform.web.springmvc.dao.UserDao;
import platform.web.springmvc.model.User;

@Service
public class UserServiceImpl implements UserService{
	@Autowired
	private UserDao userDao;

	@Transactional
	public List<User> getAll(){
		return userDao.getAll();
	}

	@Transactional
	public List<User> getUserByPermissionID(int perId){
		return userDao.getUserByPermissionID(perId);
	}

	@Transactional
	public User getUserByID(int id){
		return userDao.getUserByID(id);
	}

	@Transactional
	public List<User> getUserByAddress(String address){
		return userDao.getUserByAddress(address);
	}


	/**
	 * Login system
	 * @param userName  user-name of account
	 * @param pass            Pass word
	 * @return                User object
	 */
	@Transactional
	public User login(String userName, String pass){
		return userDao.login(userName, pass);
	}

	/**
	 * Get all users, using Native SQL
	 * @return List<User>
	 */
	@Transactional
	public List<User> getAllUserWithNativeSQL(){
		return userDao.getAllUserWithNativeSQL();
	}

	@Transactional
	public void insertUser(User user){
		userDao.insertUser(user);
	}
```

```
}
```

10. DAO layer

❖ UserDao

```
package platform.web.springmvc.dao;

import java.util.List;

import platform.web.springmvc.model.User;

public interface UserDao {
	//public Session getSession();

	public List<User> getAll();

	public List<User> getUserByPermissionID(int perId);

	public User getUserByID(int id);

	public List<User> getUserByAddress(String address);

	public User login(String userName, String pass);

	public List<User> getAllUserWithNativeSQL();

	public void insertUser(User user);
}
```

❖ UserDaoImpl

```
package platform.web.springmvc.dao;

import java.util.ArrayList;
import java.util.List;

import org.hibernate.Criteria;
import org.hibernate.Query;
import org.hibernate.SQLQuery;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.criterion.Order;
import org.hibernate.criterion.Restrictions;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;

import platform.web.springmvc.model.User;

@Repository
public class UserDaoImpl implements UserDao {
	@Autowired
```

```
	private SessionFactory sessionFactory;

	/*@Autowired
	@Qualifier("sessionFactory")
	private LocalSessionFactoryBean sessionFactory;*/

	public void testConnection(){
		//session = HibernateUtil.SESSION_FACTORY.openSession();
		Session session = sessionFactory.getCurrentSession();
		System.out.println("finish test connect DB");
	}

	public void insertUser(User user){
		/*session = HibernateUtil.SESSION_FACTORY.openSession();
		Transaction tx = session.beginTransaction();
		session.save(user);
		tx.commit();*/
		System.out.println("Insert user into db");
	}

	public List<User> getAll(){
		Session session = sessionFactory.getCurrentSession();
		List<User> listUsers = new ArrayList<User>();
		listUsers.addAll(session.createQuery("FROM User").list());
		return listUsers;
	}

	public List<User> getUserByPermissionID(int perId){
		Session session = sessionFactory.getCurrentSession();
		List<User> listUsers = new ArrayList<User>();
		listUsers.addAll(session.createQuery("FROM User WHERE
permissionId=:perId").setParameter("perId", perId).list());
		return listUsers;
	}

	public User getUserByID(int id){
		//session = HibernateUtil.SESSION_FACTORY.openSession();
		Session session = sessionFactory.getCurrentSession();
		//User user = session.load(User.class, id);

		Criteria criteria = session.createCriteria(User.class);
		criteria.add(Restrictions.eq("id", id));
		User user = (User)criteria.list().get(0);
		return user;
	}

	public List<User> getUserByAddress(String address){
		//session = HibernateUtil.SESSION_FACTORY.openSession();
		Session session = sessionFactory.getCurrentSession();
		Criteria criteria = session.createCriteria(User.class);
		criteria.add(Restrictions.like("address", "%"+address+"%"));
		criteria.addOrder(Order.asc("userName"));
		//List userList = criteria.list();
		return criteria.list();
	}
```

```
	public long countUserByAddress(String address){
		//session = HibernateUtil.SESSION_FACTORY.openSession();
		Session session = sessionFactory.getCurrentSession();
		Transaction tx = session.beginTransaction();

		String sql = "SELECT count(*) FROM User WHERE address LIKE :address";
		Query query = session.createQuery(sql);
		query.setParameter("address", "%"+address + "%");
		List resultList = query.list();
		tx.commit();
		return (Long)resultList.get(0);
	}

	/**
	 * Login system
	 * @param userName  user-name of account
	 * @param pass            Pass word
	 * @return                User object
	 */
	public User login(String userName, String pass){
		//session = HibernateUtil.SESSION_FACTORY.openSession();
		Session session = sessionFactory.getCurrentSession();
		Transaction tx = session.beginTransaction();

		String sql = "FROM User u "
				+ "WHERE userName = :userName and
password=MD5(:password)";
		Query query = session.createQuery(sql);
		query.setParameter("userName", userName).setParameter("password", pass);

		tx.commit();
		if(query.list().size() > 0)
			return (User)query.list().get(0);
		else
			return null;
	}

	/**
	 * Get all users, using Native SQL
	 * @return List<User>
	 */
	public List<User> getAllUserWithNativeSQL(){
		//session = HibernateUtil.SESSION_FACTORY.openSession();
		Session session = sessionFactory.getCurrentSession();
		Transaction tx = session.beginTransaction();

		String sql = "SELECT * FROM users";
		SQLQuery query = session.createSQLQuery(sql);
		query.addEntity(User.class);
		List<User> list = query.list();
		tx.commit();
		return list;
	}
```

```
	public static void deleteObject(){

	}

	/*
	public static void main( String[] args )
	{
		UserDaoImpl app = new UserDaoImpl();
		//insertUser();
		System.out.println(">>>>>> get all user");
		for(User user : app.getAll()){
			System.out.println(user.getId() + "\t" + user.getUserName() +
"\t" + user.getPermission().getPermissionName());
		}

		System.out.println(">>>>>> find user");
		User u = app.getUserByID(2);
		System.out.println(u.getId() + "\t" + u.getUserName());

		System.out.println(">>>>>> find user by address");
		for(User object : app.getUserByAddress("Cau Giay")){
			System.out.println(object.getUserName());
		}

		System.out.println(">>>>> Count user by address: Cau Giay");
		System.out.println(app.countUserByAddress("Cau Giay") + " users has
address at Cau Giay");

		System.out.println(">>>>>> Login:");
		System.out.println(app.login("root", "123456").getUserName());

		System.out.println(">>>>>> Get all user with Native SQL");
		for(User obj : app.getAllUserWithNativeSQL()){
			System.out.println(obj.getUserName());
		}

		System.out.println(">>>>>> get all user");
		for(User user : app.getAll()){
			System.out.println(user.getId() + "\t" + user.getUserName() +
"\t" + user.getPermission().getPermissionName());
		}

		System.out.println(">>>>>> Login:");
		System.out.println(app.login("root", "123456").getUserName());
		System.out.println(">>>>>> Finish");
	}
	*/
}
```

11. Model layer

```
package platform.web.springmvc.model;
```

```java
import java.io.Serializable;
import java.util.Date;

import javax.persistence.CascadeType;
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.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.persistence.UniqueConstraint;

@Entity
@Table(name="users", uniqueConstraints = {
            @UniqueConstraint(columnNames = "permissionId") })
public class User implements Serializable{
	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	@Column(name="id")
	private int id;
	@Column(name="firstName")
	private String firstName;
	@Column(name="lastName")
	private String lastName;
	@Column(name="email")
	private String email;
	@Column(name="city")
	private String city;
	@Column(name="country")
	private String country;
	@Column(name="userName")
	private String userName;
	@Column(name="password")
	private String password;
	@Column(name="birthDay")
	private Date birthDay;
	@Column(name="dateSingUp")
	private Date dateSingUp;
	@Column(name="address")
	private String address;
	@Column(name="website")
	private String website;
	@Column(name="avatar")
	private String avatar;
	@Column(name="phone")
	private String phone;
	@Column(name="permissionId")
	private int permissionId;

	/*@JoinColumn(name="permissionId", nullable=true)
	@ManyToOne(cascade = CascadeType.ALL, fetch = FetchType.LAZY)
	private Permission permission;*/
```

```
/*public Permission getPermission() {
	return permission;
}
public void setPermission(Permission permission) {
	this.permission = permission;
}*/

public int getId() {
	return id;
}
public void setId(int 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;
}
public String getEmail() {
	return email;
}
public void setEmail(String email) {
	this.email = email;
}
public String getCity() {
	return city;
}
public void setCity(String city) {
	this.city = city;
}
public String getCountry() {
	return country;
}
public void setCountry(String country) {
	this.country = country;
}
public int getPermissionId() {
	return permissionId;
}
public void setPermissionId(int permissionId) {
	this.permissionId = permissionId;
}
public String getUserName() {
	return userName;
}
public void setUserName(String userName) {
	this.userName = userName;
```

```
	}
	public String getPassword() {
		return password;
	}
	public void setPassword(String password) {
		this.password = password;
	}
	public Date getBirthDay() {
		return birthDay;
	}
	public void setBirthDay(Date birthDay) {
		this.birthDay = birthDay;
	}
	public Date getDateSingUp() {
		return dateSingUp;
	}
	public void setDateSingUp(Date dateSingUp) {
		this.dateSingUp = dateSingUp;
	}
	public String getAddress() {
		return address;
	}
	public void setAddress(String address) {
		this.address = address;
	}
	public String getWebsite() {
		return website;
	}
	public void setWebsite(String website) {
		this.website = website;
	}
	public String getAvatar() {
		return avatar;
	}
	public void setAvatar(String avatar) {
		this.avatar = avatar;
	}
	public String getPhone() {
		return phone;
	}
	public void setPhone(String phone) {
		this.phone = phone;
	}
}
```

12. Kiểm tra kết quả qua Web-service


localhost:8080/rest/emps
EPS - PM
EPS - Pro Docs
EPS - RnD Apps
Big-Data
PMP
My Source
JSON
Raw Data
Headers
Save
Copy
Collapse All
Expand All
Filter JSON
0:
id: 1
firstName: "root"
lastName: "demo"
email: "root@localhost.com"
city: "Ha Noi"
country: "Viet Nam"
userName: "root"
password: "e10adc3949ba59abbe56e057f20f883e"
birthDay: 466707600000
dateSingUp: 1248714000000
address: "Yen Hoa, Cau Giay,Ha Noi"
website: null
avatar: null
phone: null
permissionId: 1
1:
id: 2
firstName: "H"
lastName: "admin"
email: "admin@localhost.com"
city: "Ha Noi"
country: "Viet Name"
userName: "admin"
password: "e10adc3949ba59abbe56e057f20f883e"
birthDay: 466707600000
dateSingUp: 1248973200000
address: "Yen Hoa, Cau Giay,Ha Noi"
website: "null"
avatar: " "
phone: "null"
permissionId: 2