

Day12

SSM整合 (XML)

SSM 整合

整合步骤

项目结构搭建

新建Maven项目

创建分层

配置 web.xml

Spring 整合 MyBatis

配置spring-mvc.xml

编写模型层

编写数据层

编写业务层

编写控制层

1. 封装返回结果
2. code编码常量
3. 自定义异常
4. 异常处理器

测试

课堂作业

SSM整合 (XML)

SSM 整合

整合步骤

1. Spring
2. MyBatis
3. Spring整合MyBatis

- 4. SpringMVC
- 5. Spring整合SpringMVC

项目结构搭建

- 项目基础结构搭建
- 创建项目，组织项目结构，创建包
- 创建表与实体类
- 创建三层架构对应的模块、接口与实体类，建立关联关系
 - 数据层接口（代理自动创建实现类）
 - 业务层接口+业务层实现类
 - 表现层类

表结构：

```
1 CREATE TABLE `user` (  
2   `uuid` int NOT NULL AUTO_INCREMENT,  
3   `username` varchar(100) DEFAULT NULL,  
4   `password` varchar(100) DEFAULT NULL,  
5   `real_name` varchar(100) DEFAULT NULL,  
6   `gender` int DEFAULT NULL,  
7   `birthday` date DEFAULT NULL,  
8   PRIMARY KEY (`uuid`))  
9 )
```

SQL | 复制代码

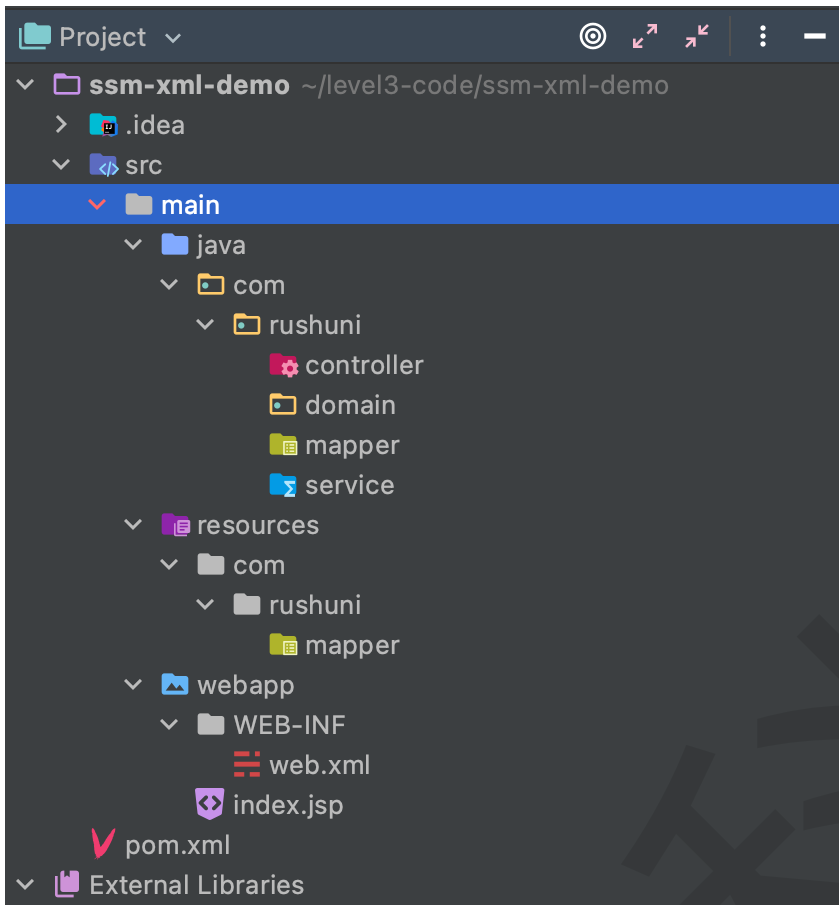
新建Maven项目

添加依赖：

```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <project xmlns="http://maven.apache.org/POM/4.0.0"
4   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
6   http://maven.apache.org/xsd/maven-4.0.0.xsd">
7   <modelVersion>4.0.0</modelVersion>
8
9   <groupId>com.rushuni</groupId>
10  <artifactId>ssm-xml-demo</artifactId>
11  <version>1.0-SNAPSHOT</version>
12  <packaging>war</packaging>
13
14  <name>ssm-xml-demo Maven Webapp</name>
15  <!-- FIXME change it to the project's website -->
16  <url>http://www.example.com</url>
17
18  <properties>
19    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
20    <maven.compiler.source>11</maven.compiler.source>
21    <maven.compiler.target>11</maven.compiler.target>
22  </properties>
23
24  <dependencies>
25    <dependency>
26      <groupId>org.springframework</groupId>
27      <artifactId>spring-webmvc</artifactId>
28      <version>5.3.9</version>
29    </dependency>
30    <dependency>
31      <groupId>org.mybatis</groupId>
32      <artifactId>mybatis</artifactId>
33      <version>3.5.7</version>
34    </dependency>
35    <dependency>
36      <groupId>com.github.pagehelper</groupId>
37      <artifactId>pagehelper</artifactId>
38      <version>5.2.1</version>
39    </dependency>
40    <dependency>
41      <groupId>mysql</groupId>
42      <artifactId>mysql-connector-java</artifactId>
43      <version>8.0.25</version>
44    </dependency>
45    <dependency>
46      <groupId>com.alibaba</groupId>
47      <artifactId>druid</artifactId>
48      <version>1.2.6</version>
```

```
47     </dependency>
48     <dependency>
49         <groupId>org.mybatis</groupId>
50         <artifactId>mybatis-spring</artifactId>
51         <version>2.0.6</version>
52     </dependency>
53     <dependency>
54         <groupId>org.springframework</groupId>
55         <artifactId>spring-jdbc</artifactId>
56         <version>5.3.9</version>
57     </dependency>
58     <dependency>
59         <groupId>commons-fileupload</groupId>
60         <artifactId>commons-fileupload</artifactId>
61         <version>1.4</version>
62     </dependency>
63     <dependency>
64         <groupId>ch.qos.logback</groupId>
65         <artifactId>logback-classic</artifactId>
66         <version>1.2.3</version>
67     </dependency>
68     <!-- jackson begging -->
69     <dependency>
70         <groupId>com.fasterxml.jackson.core</groupId>
71         <artifactId>jackson-annotations</artifactId>
72         <version>2.12.4</version>
73     </dependency>
74
75     <dependency>
76         <groupId>com.fasterxml.jackson.core</groupId>
77         <artifactId>jackson-core</artifactId>
78         <version>2.12.4</version>
79     </dependency>
80
81     <dependency>
82         <groupId>com.fasterxml.jackson.core</groupId>
83         <artifactId>jackson-databind</artifactId>
84         <version>2.12.4</version>
85     </dependency>
86     <!-- jackson end -->
87
88 </dependencies>
89
90 </project>
```

创建分层



配置 web.xml

1. 配置读取 spring 配置文件的路径: applicationContext.xml
2. 监听 spring 运行环境
3. 配置编码过滤器
4. 配置 DispatcherServlet


```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
3         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
5         http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
6         version="3.1">
7
8     <context-param>
9         <param-name>contextConfigLocation</param-name>
10        <param-value>classpath*:applicationContext.xml</param-value>
11    </context-param>
12
13    <!--启动服务器时，通过监听器加载spring运行环境-->
14    <listener>
15        <listener-
16class>org.springframework.web.context.ContextLoaderListener</listener-class>
17    </listener>
18
19    <filter>
20        <filter-name>CharacterEncodingFilter</filter-name>
21        <filter-
22class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
23        <init-param>
24            <param-name>encoding</param-name>
25            <param-value>UTF-8</param-value>
26        </init-param>
27    </filter>
28    <filter-mapping>
29        <filter-name>CharacterEncodingFilter</filter-name>
30        <url-pattern>/*</url-pattern>
31    </filter-mapping>
32
33    <servlet>
34        <servlet-name>DispatcherServlet</servlet-name>
35        <servlet-
36class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
37        <init-param>
38            <param-name>contextConfigLocation</param-name>
39            <param-value>classpath*:spring-mvc.xml</param-value>
40        </init-param>
41    </servlet>
42    <servlet-mapping>
43        <servlet-name>DispatcherServlet</servlet-name>
44        <url-pattern>/</url-pattern>
45    </servlet-mapping>
46</web-app>
```

Spring 整合 MyBatis

1. 配置jdbc.properties

```
1 jdbc.driver=com.mysql.cj.jdbc.Driver
2 jdbc.url=jdbc:mysql://localhost:3306/xxx
3 jdbc.username=xxx
4 jdbc.password=xxx
```

Plain Text

 复制代码

2. 配置 applicationContext.xml

- a. 开启 bean 扫描，排除 controller
- b. 开启事务驱动
- c. 加载 properties 文件
- d. 配置数据库连接池
- e. 整合 mybatis
 - i. 注入数据库连接池
 - ii. 配置实体类的别名
 - iii. 配置分页插件 pagehelper
- f. 扫描 mapper 接口
- g. 配置事务管理器。

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <beans xmlns="http://www.springframework.org/schema/beans"
3         xmlns:context="http://www.springframework.org/schema/context"
4         xmlns:tx="http://www.springframework.org/schema/tx"
5         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6         xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx.xsd">
9
10     <!-- a.开启bean注解扫描 -->
11     <context:component-scan base-package="com.rushuni">
12         <context:exclude-filter type="annotation"
expression="org.springframework.stereotype.Controller"/>
13     </context:component-scan>
14
15     <!-- b.开启注解式事务 -->
16     <tx:annotation-driven transaction-manager="txManager"/>
17
18     <!-- c.加载properties文件 -->
19     <context:property-placeholder location="classpath*:jdbc.properties"/>
20
21     <!-- d.配置数据库连接池 -->
22     <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource">
23         <property name="driverClassName" value="${jdbc.driver}"/>
24         <property name="url" value="${jdbc.url}"/>
25         <property name="username" value="${jdbc.username}"/>
26         <property name="password" value="${jdbc.password}"/>
27     </bean>
28
29     <!-- e.整合mybatis到spring中-->
30     <bean class="org.mybatis.spring.SqlSessionFactoryBean">
31         <property name="dataSource" ref="dataSource"/>
32         <property name="typeAliasesPackage" value="com.rushuni.domain"/>
33         <!--分页插件-->
34         <property name="plugins">
35             <array>
36                 <bean class="com.github.pagehelper.PageInterceptor">
37                     <property name="properties">
38                         <props>
39                             <prop key="helperDialect">mysql</prop>
40                             <prop key="reasonable">true</prop>
41                         </props>
42                     </property>
43                 </bean>
44             </array>

```



```

45         </property>
46     </bean>
47
48     <!-- f.映射扫描-->
49     <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
50         <property name="basePackage" value="com.rushuni.mapper"/>
51     </bean>
52
53     <!-- g.事务管理器-->
54     <bean id="txManager"
55         class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
56         <property name="dataSource" ref="dataSource"/>
57     </bean>
58 </beans>

```

配置spring-mvc.xml

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <beans xmlns="http://www.springframework.org/schema/beans"
3         xmlns:context="http://www.springframework.org/schema/context"
4         xmlns:mvc="http://www.springframework.org/schema/mvc"
5         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6         xsi:schemaLocation="http://www.springframework.org/schema/beans
7             http://www.springframework.org/schema/beans/spring-beans.xsd
8             http://www.springframework.org/schema/mvc
9             http://www.springframework.org/schema/mvc/spring-mvc.xsd
10            http://www.springframework.org/schema/context
11            http://www.springframework.org/schema/context/spring-context.xsd">
12
13     <mvc:annotation-driven/>
14
15     <context:component-scan base-package="com.rushuni.controller"/>
16
17 </beans>

```

XML

复制代码

编写模型层

```
1 public class User implements Serializable {  
2     private Integer uuid;  
3     private String userName;  
4     private String password;  
5     private String realName;  
6     private Integer gender;  
7     private Date birthday;  
8     //...  
9 }
```

编写数据层

接口：

```
1 package com.rushuni.mapper;
2
3 import com.rushuni.domain.User;
4 import org.apache.ibatis.annotations.Param;
5
6 import java.util.List;
7
8 /**
9  * @author rushuni
10  * @date 2021年07月26日 10:20 上午
11  */
12 public interface UserMapper {
13     /**
14      * 添加用户
15      * @param user
16      * @return
17      */
18     boolean save(User user);
19
20     /**
21      * 修改用户
22      * @param user
23      * @return
24      */
25     boolean update(User user);
26
27     /**
28      * 删除用户
29      * @param uuid
30      * @return
31      */
32     boolean delete(Integer uuid);
33
34     /**
35      * 查询单个用户信息
36      * @param uuid
37      * @return
38      */
39     User get(Integer uuid);
40
41     /**
42      * 查询全部用户信息
43      * @return
44      */
45     List<User> listAll();
46
47
48     /**
```

```
49      * 根据用户名密码查询个人信息
50      * @param userName 用户名
51      * @param password 密码信息
52      * @return
53      */
54      // 注意：数据层操作不需要和业务层操作的名称一样，通常数据层仅反映与数据库间的信息交换，
    不体现业务逻辑
55      User getByUserNameAndPassword(@Param("userName") String userName,
    @Param("password") String password);
56  }
57
```

实现：

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE mapper
3      PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
4      "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
5  <mapper namespace="com.rushuni.mapper.UserMapper">
6
7      <!--添加-->
8      <insert id="save" parameterType="user">
9          insert into user(userName,password,realName,gender,birthday)values(#{
10              userName},#{password},#{realName},#{gender},#{birthday})
11      </insert>
12
13      <!--删除-->
14      <delete id="delete" parameterType="int">
15          delete from user where uuid = #{uuid}
16      </delete>
17
18      <!--修改-->
19      <update id="update">
20          update user set userName=#{userName},password=#{password},realName=
21          {realName},gender=#{gender},birthday=#{birthday} where uuid=#{uuid}
22      </update>
23
24      <!--查询单个-->
25      <select id="get" resultType="user">
26          select * from user where uuid = #{uuid}
27      </select>
28
29      <!--分页查询-->
30      <select id="listAll" resultType="user">
31          select * from user
32      </select>
33
34      <!--登录-->
35      <select id="getByUserNameAndPassword" resultType="user" >
36          select * from user where userName=#{userName} and password=
37          {password}
38      </select>
39  </mapper>

```

编写业务层

接口：

```
1 package com.rushuni.service;
2
3 import com.github.pagehelper.PageInfo;
4 import com.rushuni.domain.User;
5 import org.springframework.transaction.annotation.Transactional;
6
7 /**
8  * @author rushuni
9  * @date 2021年07月26日 10:22 上午
10 */
11 @Transactional(readOnly = true)
12 public interface UserService {
13     /**
14      * 添加用户
15      * @param user
16      * @return
17      */
18     @Transactional(readOnly = false)
19     public boolean save(User user);
20
21     /**
22      * 修改用户
23      * @param user
24      * @return
25      */
26     @Transactional(readOnly = false)
27     public boolean update(User user);
28
29     /**
30      * 删除用户
31      * @param uuid
32      * @return
33      */
34     @Transactional(readOnly = false)
35     public boolean delete(Integer uuid);
36
37     /**
38      * 查询单个用户信息
39      * @param uuid
40      * @return
41      */
42     public User get(Integer uuid);
43
44     /**
45      * 查询全部用户信息
46      * @return
47      */
48     public PageInfo<User> getAll(int page, int size);
```

```
49
50     /**
51      * 根据用户名密码进行登录
52      * @param userName
53      * @param password
54      * @return
55      */
56     public User login(String userName,String password);
57 }
```

实现：

```
1 package com.rushuni.service.impl;
2
3 import com.github.pagehelper.PageHelper;
4 import com.github.pagehelper.PageInfo;
5 import com.rushuni.domain.User;
6 import com.rushuni.mapper.UserMapper;
7 import org.springframework.beans.factory.annotation.Autowired;
8 import org.springframework.stereotype.Service;
9
10 import java.util.List;
11
12 /**
13  * @author rushuni
14  * @date 2021年07月26日 10:25 上午
15  */
16 @Service
17 public class UserService implements com.rushuni.service.UserService {
18
19     @Autowired
20     private UserMapper userMapper;
21
22     @Override
23     public boolean save(User user) {
24         return userMapper.save(user);
25     }
26
27     @Override
28     public boolean update(User user) {
29         return userMapper.update(user);
30     }
31
32     @Override
33     public boolean delete(Integer uuid) {
34         return userMapper.delete(uuid);
35     }
36
37     @Override
38     public User get(Integer uuid) {
39         return userMapper.get(uuid);
40     }
41
42     @Override
43     public PageInfo<User> getAll(int page, int size) {
44         PageHelper.startPage(page, size);
45         List<User> all = userMapper.listAll();
46         return new PageInfo<>(all);
47     }
48 }
```



```
49     @Override
50     public User login(String userName, String password) {
51         return userMapper.getByUserNameAndPassword(userName,password);
52     }
53 }
```

编写控制层

1. 封装返回结果

```
1 package com.rushuni.controller.result;
2
3 /**
4  * @author rushuni
5  * @date 2021年07月26日 10:32 上午
6  */
7 public class Results {
8     // 操作结果编码
9     private Integer code;
10    // 操作数据结果
11    private Object data;
12    // 消息
13    private String message;
14
15    public Results(Integer code) {
16        this.code = code;
17    }
18
19    public Results(Integer code, Object data) {
20        this.code = code;
21        this.data = data;
22    }
23
24    @Override
25    public String toString() {
26        return "Result{" +
27            "code=" + code +
28            ", data=" + data +
29            ", message='" + message + '\'' +
30            '}';
31    }
32
33    public Integer getCode() {
34        return code;
35    }
36
37    public void setCode(Integer code) {
38        this.code = code;
39    }
40
41    public Object getData() {
42        return data;
43    }
44
45    public void setData(Object data) {
46        this.data = data;
47    }
48}
```

```

49     public String getMessage() {
50         return message;
51     }
52
53     public void setMessage(String message) {
54         this.message = message;
55     }
56 }
57

```

2. code编码常量

```

1  package com.rushuni.controller.result;
2
3  /**
4   * @author rushuni
5   * @date 2021年07月26日 10:32 上午
6   */
7  public class Codes {
8
9      // 操作结果编码
10     public static final Integer SAVE_OK = 20011;
11     public static final Integer UPDATE_OK = 20021;
12     public static final Integer DELETE_OK = 20031;
13     public static final Integer GET_OK = 20041;
14
15     public static final Integer SAVE_ERROR = 20010;
16     public static final Integer UPDATE_ERROR = 20020;
17     public static final Integer DELETE_ERROR = 20030;
18     public static final Integer GET_ERROR = 20040;
19 }

```

Java  复制代码

3. 自定义异常

单独放在 system 包下

```
1 package com.rushuni.system.exception;
2
3 /**
4  * @author rushuni
5  * @date 2021年07月26日 10:40 上午
6  */
7 public class BusinessException extends RuntimeException {
8     //自定义异常中封装对应的错误编码，用于异常处理时获取对应的操作编码
9     private Integer code;
10
11     public Integer getCode() {
12         return code;
13     }
14
15     public void setCode(Integer code) {
16         this.code = code;
17     }
18
19     public BusinessException(Integer code) {
20         this.code = code;
21     }
22
23     public BusinessException(String message, Integer code) {
24         super(message);
25         this.code = code;
26     }
27
28     public BusinessException(String message, Throwable cause, Integer code) {
29         super(message, cause);
30         this.code = code;
31     }
32
33     public BusinessException(Throwable cause, Integer code) {
34         super(cause);
35         this.code = code;
36     }
37
38     public BusinessException(String message, Throwable cause, boolean
        enableSuppression, boolean writableStackTrace, Integer code) {
39         super(message, cause, enableSuppression, writableStackTrace);
40         this.code = code;
41     }
42 }
```

4. 异常处理器

放在 interception 包。

```
1 package com.rushuni.controller.interceptor;
2
3 import com.rushuni.controller.result.Results;
4 import com.rushuni.system.exception.BusinessException;
5 import org.springframework.stereotype.Component;
6 import org.springframework.web.bind.annotation.ControllerAdvice;
7 import org.springframework.web.bind.annotation.ExceptionHandler;
8 import org.springframework.web.bind.annotation.ResponseBody;
9
10 /**
11  * @author rushuni
12  * @date 2021年07月24日 3:56 下午
13  */
14 @Component
15 @ControllerAdvice
16 public class ProjectExceptionHandler {
17
18     @ExceptionHandler(BusinessException.class)
19     @ResponseBody
20     // 对出现异常的情况进行拦截，并将其处理成统一的页面数据结果格式
21     public Results doBusinessException(BusinessException e){
22         return new Results(e.getCode(),e.getMessage());
23     }
24
25 }
```

测试

```
localhost:8088/ssm_xml_demo_war/user/1
{
  code: 20041,
  - data: {
    uuid: 1,
    userName: "zs",
    password: "123",
    realName: null,
    gender: 1,
    birthday: 973872000000
  },
  message: "操作成功"
}
```

← → ↻ ⓘ localhost:8088/ssm_xml_demo_war/user/10

```
{  
  code: 20040,  
  data: "查询出错啦, 请重试!",  
  message: null  
}
```

课堂作业

- 完成 SSM 整合，上传到公司 git。