河北师范大学软件学院
Software College of Hebei Normal University

MyBatis

第八讲
Spring集成MyBatis

目录

基本准备

- 创建Dynamic Web Project
- 引入相关jar包
  - Spring框架相关jar包
  - MyBatis连接Spring相关jar包
  - 连接MySQL驱动包
  - JSTL标签库包

基本准备

- 创建Dynamic Web Project
- 引入相关jar包
  - Spring框架相关jar包
  - MyBatis连接Spring相关jar包
  - 连接MySQL驱动包
  - JSTL标签库包

基本准备

- 添加db.properties文件，该属性文件配置连接数据库相关信息

```
driver=com.mysql.jdbc.Driver
url=jdbc:mysql://localhost:3306/mybatis?characterEncoding=utf-8
user=root
password=
```

- 添加Spring配置文件，在src中新增applicationContext.xml，并引入该属性文件

```
<!-- 导入数据库连接信息的属性文件 -->
<context:property-placeholder location="classpath:db.properties"/>
```

基本准备

- 在Spring配置文件中添加DataSource的配置，PooledDataSource为MyBatis实现的数据库连接池

```
<!-- 配置数据源 -->
<bean id="dataSource"
class="org.apache.ibatis.datasource.pooled.PooledDataSource">
<property name="driver" value="${driver}"></property>
<property name="username" value="${user}"></property>
<property name="password" value="${password}"></property>
<property name="url" value="${url}"></property>
</bean>
```

基本准备

- 在Spring配置文件中添加SqlSessionFactoryBean，用来创建SqlSessionFactory对象
  - configLocation：用于指定MyBatis的mybatis.xml配置文件的路径
  - dataSource：用于配置数据源，该属性为必选项，可以直接引用已经配置好的dataSource数据库连接池
  - mapperLocations：扫描XML映射文件的路径

基本准备

```
<bean id="sqlSessionFactory"
    class="org.mybatis.spring.SqlSessionFactoryBean">
    <!-- 添加mybatis主配置文件的位置 -->
    <property name="configLocation" value="classpath:mybatis.xml"/>
    <!-- 需要一个数据源 -->
    <property name="dataSource" ref="dataSource"></property>
    <!-- 设置映射文件的位置 -->
    <property name="mapperLocations">
        <array>
            <value>classpath:net/onest/server/dao/*.xml</value>
        </array>
    </property>
</bean>
```

基本准备

- 在Spring配置文件中添加MapperScannerConfigurer，自动扫描所有的Mapper接口
  - basePackage：用于配置基本的包路径

```
<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
    <property name="basePackage" value="net.onest.server.dao"/>
</bean>
```

基本准备

- 按照SSH集成的web.xml，添加Spring和SpringMVC相关配置
- 创建实体类
- 创建数据库表
- 创建index.jsp

目录

开发Mapper层（Dao层）

- Mapper层也就是常说的数据访问层（Dao层）。根据配置文件中配置的自动扫描接口的包名创建映射接口和XML映射文件

```
public interface UserMapper {
    public List<User> findAllUsers();
    public int saveUser(User u);
}
```

开发Mapper层（Dao层）

```
<mapper namespace="net.onest.server.dao.UserMapper">
    <resultMap type="net.onest.server.entity.User" id="userMap">
        <id column="id" property="id"/>
        <result column="user_name" property="userName"/>
        <result column="password" property="password"/>
    </resultMap>
    <select id="findAllUsers" resultMap="userMap">
        select * from user
    </select>
    <insert id="saveUser">
        insert into user(user_name,password)
        values(#{userName},#{password})
    </insert>
</mapper>
```

目录

开发业务层（Service层）

- 添加Service层的接口和实现类

```
public interface UserService {

    public List<User> findAllUsers();
    public int saveUser(User u);
}
```

开发业务层（Service层）

```
@Service
public class UserServiceImpl implements UserService{
    @Autowired
    private UserMapper userMapper;
    @Override
    public List<User> findAllUsers() {
        return userMapper.findAllUsers();
    }
    @Override
    public int saveUser(User u) {
        return userMapper.saveUser(u);
    }
}
```

开发业务层（Service层）

- Service的实现类需要添加@Service注解，由于在Spring配置文件中配置了自动扫描Service实现类所在的包，所以Spring在初始化时就会扫描到添加了@Service注解的类
- 由于配置了自动扫描Mapper接口，所以在Service层可以使用@AutoWired注解自动注入Mapper

目录

开发控制层（Controller层）

```java
@RequestMapping("/user")
@Controller
public class UserController {
    @Autowired
    private UserService userService;

    @RequestMapping("/userList")
    public ModelAndView getUsers() {
        ModelAndView mv = new ModelAndView("userList");
        List<User> users = userService.findAllUsers();
        mv.addObject("users", users);
        return mv;
    }
    ……
}
```

开发控制层（Controller层）

```
@RequestMapping("/addUser")
public ModelAndView addUser() {
    ModelAndView mv = new ModelAndView("addUser");
    User u = new User();
    mv.addObject("user", u);
    return mv;
}
@RequestMapping("/saveUser")
public ModelAndView saveUser(User u) {
    ModelAndView mv = new ModelAndView();
    userService.saveUser(u);
    mv.setViewName("redirect:/user/userList");
    return mv;
}
```

目录

开发视图层（View层）

- 根据在SpringMVC配置文件中的视图配置需要在WebContent/WEB-INF中新建jsp目录来存放所有的jsp文件

```
<bean id="viewResolver" class=
"org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix" value="/WEB-INF/jsp/"/>
    <property name="suffix" value=".jsp"/>
</bean>
```

THANK YOU