Spring，day05，主讲：汤小洋

一、Spring整合Web

1.基本用法

1.1 创建web工程并添加依赖

```
<!-- Spring整合Web-->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-web</artifactId>
</dependency>

<!-- Java EE-->
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>javax.servlet-api</artifactId>
</dependency>
<dependency>
  <groupId>javax.servlet.jsp</groupId>
  <artifactId>jsp-api</artifactId>
</dependency>
<dependency>
  <groupId>jstl</groupId>
  <artifactId>jstl</artifactId>
</dependency>
```

```xml
<!-- tomcat插件 -->
<plugins>
  <plugin>
    <groupId>org.apache.tomcat.maven</groupId>
    <artifactId>tomcat7-maven-plugin</artifactId>
    <version>${tomcat7-maven-plugin.version}</version>
    <configuration>
      <path>/</path>
      <port>8888</port>
    </configuration>
  </plugin>
</plugins>
```

1.2 初始Spring容器

将IoC容器的初始化交给Web容器管理

```xml
<!-- 初始化Spring容器 -->
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>classpath:spring.xml</param-value>
</context-param>
<listener>
  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
```

1.3 配置依赖注入

Dao——>Service——>Action

```xml
 <!-- 扫描包 -->
<context:component-scan base-package="com.itany.dao.impl"/>
<context:component-scan base-package="com.itany.service.impl"/>

<!-- IoC容器工具类 -->
<bean class="com.itany.util.SpringBeanHolder"/>
```

2. 其他配置

2.1 关于spring的配置文件

当未指定contextConfigLocation参数时，默认会自动读取/WEB-INF/applicationContext.xml文件

2.2 解决post请求中文乱码

```
<!-- 解决POST请求中文乱码 -->
<filter>
  <filter-name>encodingFilter</filter-name>
  <filter-
class>org.springframework.web.filter.CharacterEncodingFilter</filter-
class>
  <init-param>
    <param-name>encoding</param-name>
    <param-value>utf-8</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>encodingFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

二、Spring整合JDBC

1. 基本用法

1.1 添加依赖

```
<!-- Spring整合JDBC -->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-jdbc</artifactId>
  <version>${spring.version}</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-tx</artifactId>
  <version>${spring.version}</version>
</dependency>
```

1.2 配置Dao

DataSource——>JdbcTemplate——>Dao——>Service——>Action

1.3 配置DataSource

DataSource的实现方式：

1. 使用Spring提供的数据源，没有连接池的功能，效率低
2. 使用第三方数据源，如：dbcp、c3p0、druid(德鲁伊)

   使用dbcp

```
<!-- 使用dbcp -->
<bean id="dataSource"
class="org.apache.commons.dbcp.BasicDataSource">
  <property name="driverClassName"
value="${jdbc.driverClassName}"/>
  <property name="url" value="${jdbc.url}" />
  <property name="username" value="${jdbc.username}"/>
  <property name="password" value="${jdbc.password}"/>
  <property name="initialSize" value="${jdbc.initialSize}"/>
  <property name="maxActive" value="100"/>
  <property name="minIdle" value="3"/>
  <property name="maxIdle" value="20"/>
  <property name="maxWait" value="5000"/>
</bean>
```

2.用户注册

JDBC默认是自动提交事务的，每执行完一条SQL语句就提交事务

解决：配置事务

3. 事务操作

3.1 两种方式

定义事务管理器，相当于是事务的通知

```xml
<!-- 配置事务管理器 -->
<bean
class="org.springframework.jdbc.datasource.DataSourceTransactionManager"
>
  <property name="dataSource" ref="dataSource"/>
</bean>
```

两种方式：

- 方式1：基于命名空间

```xml
<!-- 配置Advice -->
<tx:advice id="txAdvice" transaction-
manager="transactionManager">
  <!-- 配置事务属性 -->
  <tx:attributes>
    <tx:method name="login" propagation="SUPPORTS" read-
only="true" />
    <tx:method name="regist" propagation="REQUIRED"
isolation="READ_COMMITTED" no-rollback-
for="java.lang.ArithmeticException" timeout="5000"/>
  </tx:attributes>
</tx:advice>

<!-- 配置Pointcut并织入 -->
<aop:config>
  <aop:pointcut id="pc" expression="execution(*
com.itany.service.impl.*.*(..))"/>
  <aop:advisor advice-ref="txAdvice" pointcut-ref="pc"/>
</aop:config>
```

- 基于注解

```
@Transactional(propagation =Propagation.REQUIRED,rollbackFor =
Exception.class)
```

```xml
<!-- 方式2：注解驱动 -->
<tx:annotation-driven transaction-manager="transactionManager"/>
```

3.2 事务属性

五个事务属性：

1. 传播属性

propagation：定义事务的边界，用来定义当前方法是否需要事务，常用取值：

- REQUIRED 必须添加事务，如果当前没有事务，则创建一个新的事务，一般用于增删改操作
- SUPPORTS 可以没有事务，如果当前有事务则运行，如果没有事务也可以运行，一般用于查询操作

2. 隔离级别

isolation：用来解决事务并发时会出现的一些问题，四种隔离级别：

- READ_COMMITTED：已提交读——>避免脏读，但可能发生不可重复读或幻读
- READ_UNCOMMITTED 未提交读——>可能会发生脏读、不可重复读或幻读
- REPEATABLE_READ 可重复读——>避免脏读和不可重复读，但可能会发生幻读
- SERIALIZABLE 可序列化——>避免脏读、不可重复读或幻读，相当于单并发，没意义

事务并发时可能会出现的三个问题：

- 脏读：一个事务读取到另一个事务没有提交的数据，一般不会发生，如MySQL、Oracle底层默认都只读取提交的数据
- 不可重复读：一个事务已经读取数据，另一个事务在修改数据，可能导致使用的数据与数据库不同步
- 幻读或虚读：一个事务已经读取数据，另一个事务在添加或删除数据，可能导致使用的数据量与数据库不同步

  注：不可重复读和幻读是小概率事件，可以通过版本检查来解决，如Hibernate中悲观锁和乐观锁就是通过版本检查来实现的，但太麻烦

  且效率低，实际开发中一般不需要配置隔离级别，大多是通过定时任务+人工审核

3. 回滚条件

rollback：默认抛出RuntimeException时才会回滚

rollbackFor="" 表示发生该异常时回滚

noRollbackFor="" 表示发生该异常时不回滚

4. 只读优化

readOnly：在该事务中只能读取，一般用于查询

5. 超时处理

timeout：配置事务的超时时间，一般不配置

3.3 事务特性

四个事务特性：ACID 原子性、一致性、隔离性、永久性

三、Spring整合MyBatis

1. 基本用法

1.1 添加依赖

```
<!-- Spring整合MyBatis-->
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis</artifactId>
</dependency>
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis-spring</artifactId>
</dependency>
```

1.2 创建Dao实现类

对于MyBatis而言，可以创建Dao实现类，也可以不创建，一般都不创建Dao实现类

DataSource——>SqlSessionFactory——>Dao——>Service——>Action

1.3 创建映射文件

```
<!-- 配置SqlSessionFactory -->
<bean id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">
  <!-- 可以指定单独的mybatis配置文件，但整合时一般不再使用 -->
  <!--<property name="configLocation" value="classpath:mybatis-
cofnig.mxl"/>-->
  <property name="dataSource" ref="dataSource"/>
  <!-- 指定映射文件的路径 -->
  <property name="mapperLocations"
value="classpath:com/itany/mapper/*Mapper.xml"/>
  <!-- 为映射类指定别名 -->
  <property name="typeAliasesPackage" value="com.itany.entity"/>
</bean>
```

补充：slf4j是一组日志接口，但并未提供任何实现，实际开发中整个系统中可能使用了不同的日志框架，推荐面向slf4j写日志代码，可以处理不同不同日志框架

2. 不创建Dao实现类

```
<!-- 通过反射创建Dao实现，并添加到IoC容器中 -->
<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
  <!-- 指定sqlSessionFactory -->
  <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory"/>
  <!-- 指定接口所在的包 -->
  <property name="basePackage" value="com.itany.dao"/>
</bean>
```