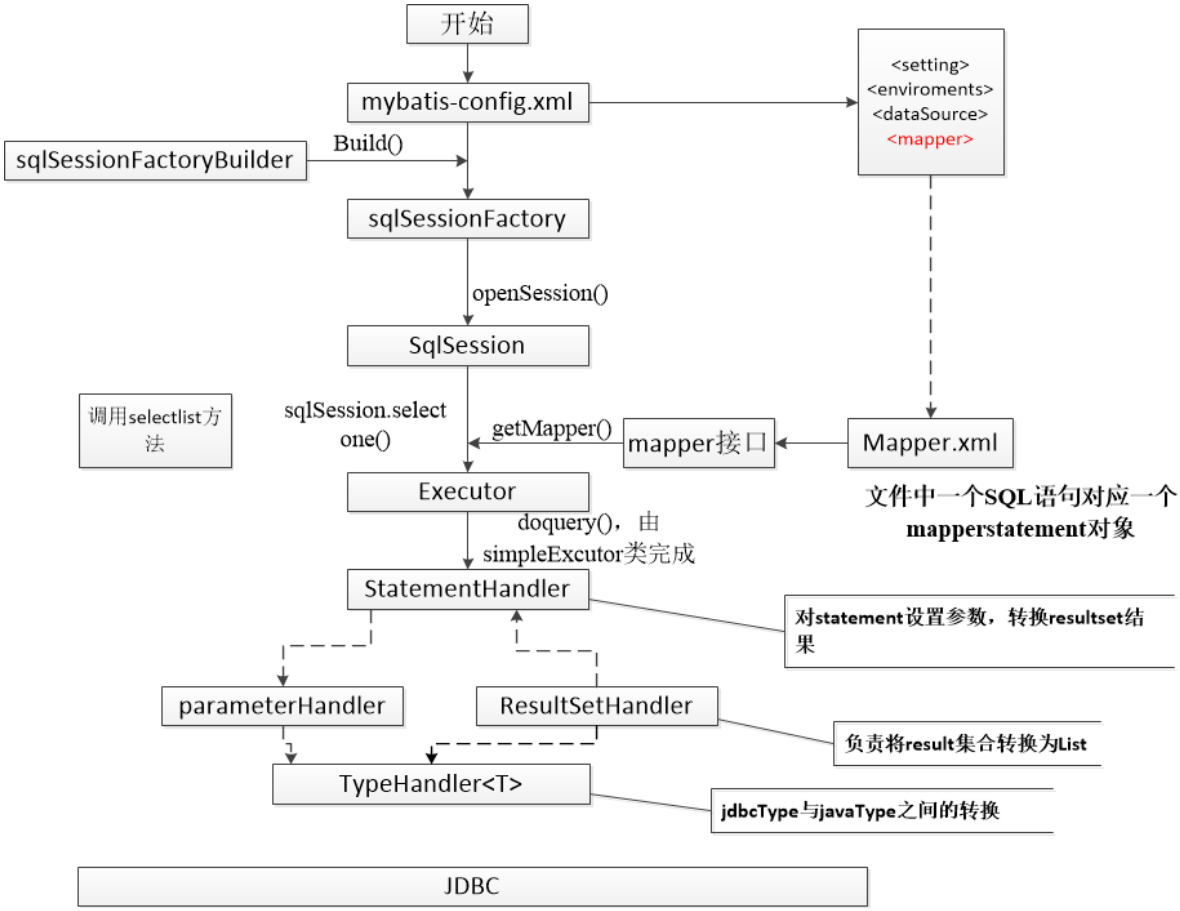
MyBatis学习

1、Mybatis工作原理

1、mybatis工作原理图：



2、步骤详解：

（1）配置XML文件

加载配置文件，XMLConfigBuilder对象会进行XML配置文件的解析，实际为configuration节点的解析操作，在configuration节点下会依次解析properties/settings/.../mappers等节点配置。在解析environments节点时，会根据transactionManager的配置来创建事务管理器，根据dataSource的配置来创建DataSource对象，这里面包含了数据库登录的相关信息。在解析mappers节点时，会读取该节点下所有的mapper文件，然后进行解析，并将解析后的结果存到configuration对象中。

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
        PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
    <settings>
        <setting name="lazyLoadingEnabled" value="false"/>
    </settings>
    <!--配置环境 -->
    <environments default="development">
        <environment id="development">
            <!-- 使用jdbc事务管理 -->
            <transactionManager type="JDBC" />
```

```
            <!-- 数据库连接池 -->
            <dataSource type="POOLED">
                <property name="driver" value="com.mysql.jdbc.Driver" />
                <property name="url"
                          value="jdbc:mysql://localhost:3306/springmybatis?
characterEncoding=utf-8" />
                <property name="username" value="root" />
                <property name="password" value="root" />
            </dataSource>
        </environment>
    </environments>

    <!--映射文件 -->
    <mappers>
        <mapper resource="com/smart/dao/userDao.xml"/>
    </mappers>
</configuration>
```

（2）、配置中的映射文件，通过namespace指定映射所在的命名空间，"com.smart.dao.UserDao"是一个接口。每个具体的映射项都有一个id，通过命名空间和映射项的id定位到具体的映射项，例如id="getUser" 对应的映射项为UserDao接口中的 getUser(User user)； parameterType指定传入的参数对象，可以是全限定类名，也可以是类的别名（在Mybatis主文件中定义）； resultType指定返回对象类型。

```
<mapper namespace="com.smart.dao.UserDao">
    <select id="getUser" parameterType="com.smart.domain.User"
            resultType="com.smart.domain.User" >
       SELECT * FROM user WHERE username=#{username} AND password=#{password}
    </select>

    <insert id="addUser" parameterType="com.smart.domain.User"
flushCache="true">
        INSERT INTO user (id,username,password) VALUES (#{id},#{username},#
{password})
    </insert>

    <update id="updateUser" parameterType="com.smart.domain.User">
        UPDATE user SET password=#{password} WHERE id=#{id}
    </update>

    <delete id="deleteUser" parameterType="int">
        DELETE FROM user WHERE id=#{id}
    </delete>
</mapper>
```

（3）、书写接口,与映射文件中的映射项的id对应

```
public interface UserDao {
    public User getUser(User user);
    public void addUser(User user);
    public void updateUser(User user);
    public void deleteUser(int UserId);
}
```

（4）、创建sqlsession

过程为：加载配置文件、通过SqlSessionFactoryBuilder对象获取SqlSessionFactory、通过SqlSessionFactory的openSession去获取sqlSession

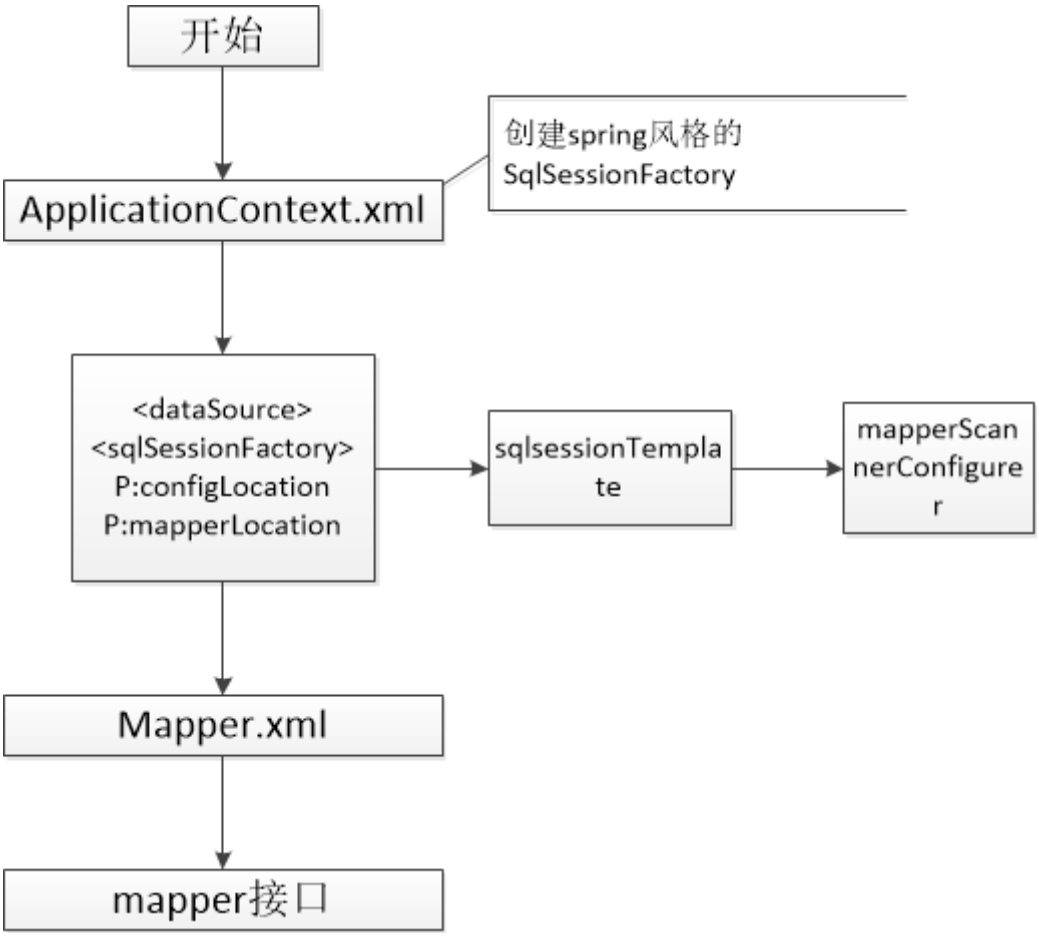
```
InputStream inputStream = Resources.getResourceAsStream("mybatisConfig.xml");
SqlSessionFactory sessionFactory = new
SqlSessionFactoryBuilder().build(inputStream);
SqlSession sqlSession = sessionFactory.openSession(true);
```

（5）、sql查询流程（使用映射接口）

```
SqlSession sqlSession = sessionFactory.openSession(true);
ForumMapper userMapper = sqlSession.getMapper(UserDao.class);
```

2、spring中配置Mybatis

简化配置



1、配置文件

```
 <!--配置数据源 -->
    <bean id="dataSource"
          class="org.apache.commons.dbcp.BasicDataSource"
          destroy-method="close"
          p:driverClassName="com.mysql.jdbc.Driver"
          p:url="jdbc:mysql://localhost:3306/springmybatis?
useUnicode=true&amp;characterEncoding=UTF-8"
          p:username="root"
          p:password="root" />
    <!-- mybatis-spring类包提供了一个sqlSessionFactory -->
    <bean id="sqlSessionFactory"
```

```
          class="org.mybatis.spring.SqlSessionFactoryBean"
          p:dataSource-ref="dataSource"
          p:configLocation="classpath:mybatisConfig.xml"
          p:mapperLocations="classpath:com/smart/dao/*.xml"/>
    <!-- mybatis-spring提供了一个模板类SqlSessionTemplate，可以通过模板类访问数据库 -->
     <bean class="org.mybatis.spring.SqlSessionTemplate">
        <constructor-arg ref="sqlSessionFactory" />
     </bean>
```

2、使用SqlSessionTemplate调用SQL映射项完成数据访问工作

```
@Repository
public class UserTemplateDao {
    @Autowired
    private SqlSessionTemplate sqlSessionTemplate;

    public User getUser(User user){
        UserDao userDao = (UserDao)
sqlSessionTemplate.getMapper(UserTemplateDao.class);
        return userDao.getUser(user);
    }
}
```

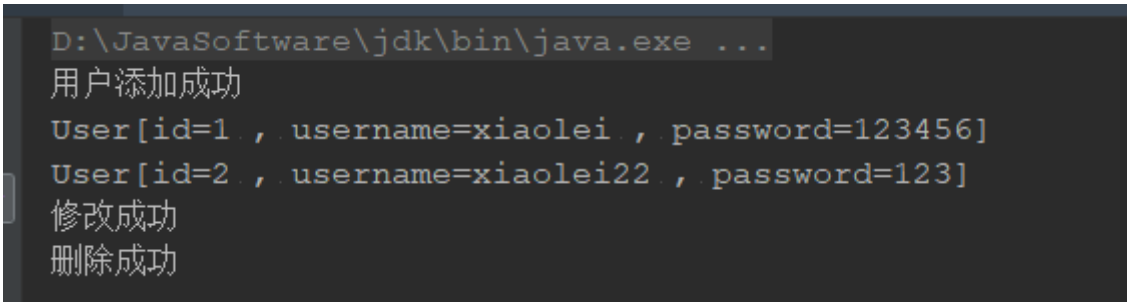
3、比调用接口更优化的方法,使用MapperScannerConfigure转换器，可以将映射接口直接转换为Spring容器中的bean，就可以在service中注入映射接口的bean。

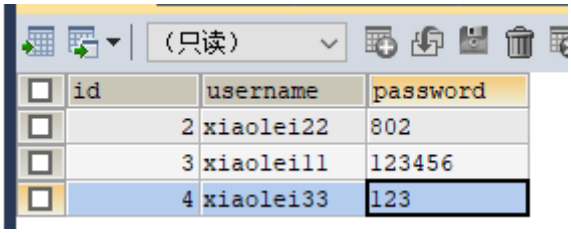
```
 <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer"
          p:sqlSessionFactory-ref="sqlSessionFactory"
          p:basePackage="com.smart.dao" />
```

4、注入service层

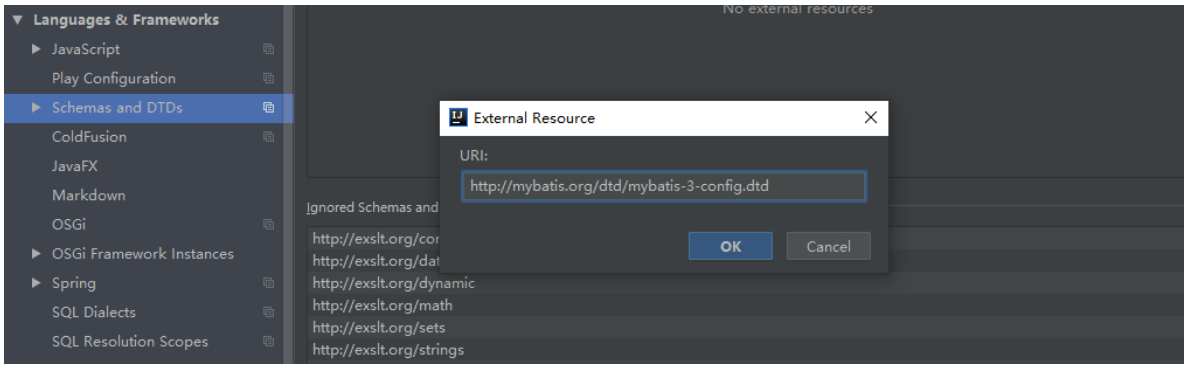
```
@Service
public class UserService {
    private UserDao  userDao;
    @Autowired
    public void setUserDao(UserDao userDao) {
        this.userDao = userDao;
    }
    public void addUser(User user){
        userDao.addUser(user);
    }
```

5、添加用户成功界面





3、在IDEA中添加mybatis配置文件



4、Mybatis分页插件PageHepler

1、引入依赖

```
<dependency>
            <groupId>com.github.pagehelper</groupId>
            <artifactId>pagehelper</artifactId>
            <version>5.1.8</version>
</dependency>
```

2、配置全局文件

```
<plugins>
        <!--这里要写成PageInterceptor， 5.0之前的版本都是写PageHelper，
         5.0之后要换成PageInterceptor-->
        <plugin interceptor="com.github.pagehelper.PageInterceptor">
            <!--reasonable：分页合理化参数，默认值为false,直接根据参数进行查询。
             当该参数设置为 true 时，pageNum<=0 时会查询第一页，
             pageNum>pages（超过总数时），会查询最后一页。-->
            <property name="reasonable" value="false"/>
        </plugin>
</plugins>
```

3、使用分页

```
@PostMapping("/getUser")
    public String getUser(int userId,int pageNum,int pageSize){
        //PageHelper.startPage()后面一定要执行SQL语句
        PageHelper.startPage(pageNum,pageSize);
        List<User> list = userService.getUser(userId);
        PageInfo<User> info = new PageInfo<User>(list);
        return JSON.toJSONString(info);
    }
```

5、mybatis缓存机制

mybatis的缓存分为两级：一级缓存、二级缓存

一级缓存是SqlSession级别的缓存，缓存的数据只在SqlSession内有效

二级缓存是mapper级别的缓存，同一个namespace公用这一个缓存，所以对SqlSession是共享的

一级缓存

mybatis的一级缓存是SqlSession级别的缓存，在操作数据库的时候需要先创建SqlSession会话对象，在对象中有一个HashMap用于存储缓存数据，此HashMap是当前会话对象私有的，别的SqlSession会话对象无法访问。

具体流程

1.第一次执行select完毕会将查到的数据写入SqlSession内的HashMap中缓存起来

2.第二次执行select会从缓存中查数据，如果select相同切传参数一样，那么就能从缓存中返回数据，不用去数据库了，从而提高了效率

注意事项

1.如果SqlSession执行了DML操作（insert、update、delete），并commit了，那么mybatis就会清空当前SqlSession缓存中的所有缓存数据，这样可以保证缓存中的存的数据永远和数据库中一致，避免出现脏读

2.当一个SqlSession结束后那么他里面的一级缓存也就不存在了，mybatis默认是开启一级缓存，不需要配置

3.mybatis的缓存是基于[namespace:sql语句:参数]来进行缓存的，SqlSession的HashMap存储缓存数据时，是使用[namespace:sql:参数]作为key，查询返回的语句作为value保存的。

二级缓存

1、二级缓存是mapper级别的缓存，也就是同一个namespace的mappe.xml，当多个SqlSession使用同一个Mapper操作数据库的时候，得到的数据会缓存在同一个二级缓存区域

2、二级缓存默认是没有开启的。

在setting全局参数中配置开启二级缓存

```
<settings>
        <setting name="cacheEnabled" value="true"/>   //默认是false：关闭二级缓存
<settings>
```

在userMapper.xml中配置：

```
//当前mapper下所有语句开启二级缓存
<cache eviction="LRU" flushInterval="60000" size="512" readOnly="true"/>

//禁用select语句的二级缓存
<select id="getCountByName" parameterType="java.util.Map" resultType="INTEGER"
statementType="CALLABLE" useCache="false">
```

具体流程

1.当一个sqlseesion执行了一次select后，在关闭此session的时候，会将查询结果缓存到二级缓存

2.当另一个sqlsession执行select时，首先会在他自己的一级缓存中找，如果没找到，就回去二级缓存中找，找到了就返回，就不用去数据库了，从而减少了数据库压力提高了性能

注意事项

如果SqlSession执行了DML操作（insert、update、delete），并commit了，那么mybatis就会清空当前mapper缓存中的所有缓存数据，这样可以保证缓存中的存的数据永远和数据库中一致，避免出现脏读.