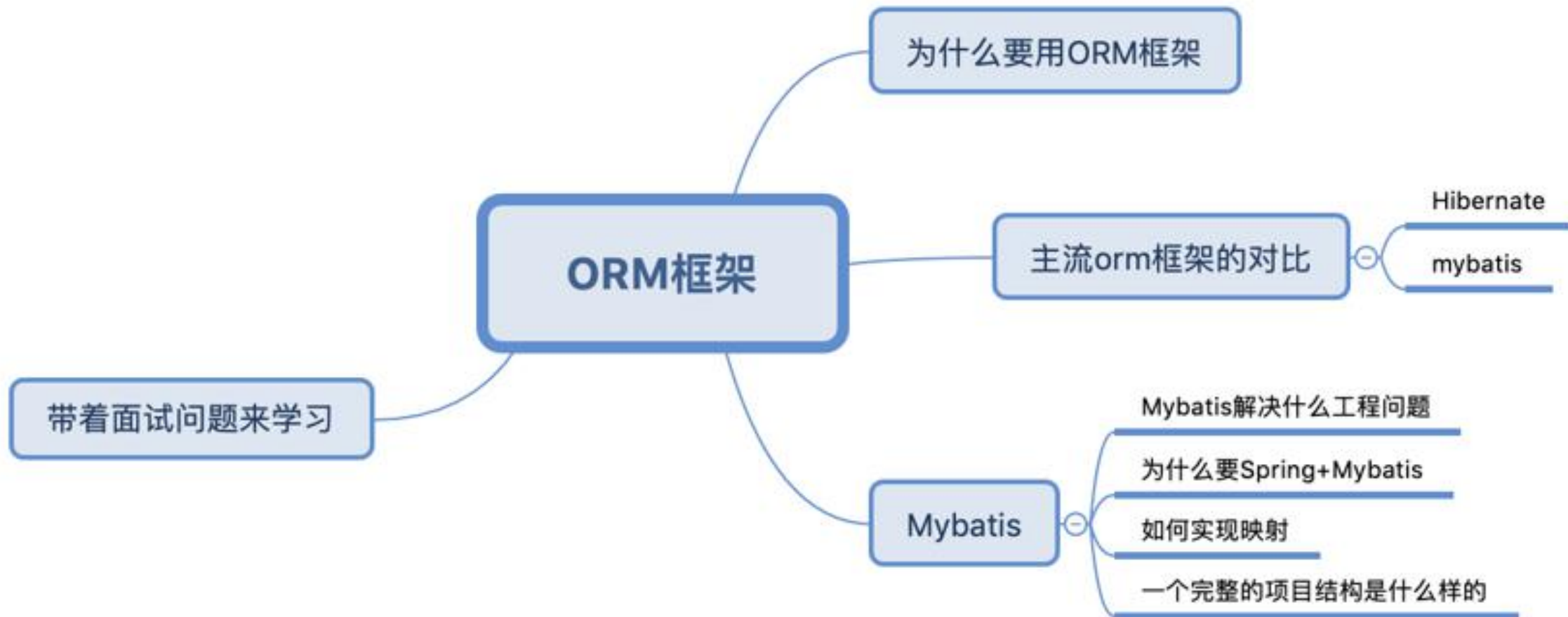


Java高级工程师

数据库终篇



在项目中为什么要使用ORM框架?

频度:高

难度:低

通过率:高

什么是ORM

Object Relational Mapping, 简称ORM

为什么要用ORM解决数据库交互问题？

为了解决面向对象与关系数据库存在的互不匹配的现象

面向对象:Bean的操作

关系数据库:关系数据元组

弥合OOM与关系数据模型的GAP

ORM是通过使用描述对象和数据库之间映射的元数据，将程序中的对象自动持久化到关系数据库中。

什么是持久化

存到数据库中

怎么实现持久化

硬编码:resultSet读出字段,给Bean中的属性赋值

缺点

持久化层缺乏弹性。一旦出现关系数据结构的变更，就必须修改持久化层的接口

- Student表

id	name
1	Jim

对应的Bean:

```
public class Student implements Serializable {  
    private String id;  
    private String name;  
}
```

```
Student stu = new Student();
```

```
String sql = "SELECT * FROM student WHERE id = ?";
```

```
PreparedStatement pstmt = conn.prepareStatement(sql);
```

```
pstmt.setString(1,"1");
```

```
//step5: 处理结果
```

```
rs = pstmt.executeQuery();
```

```
while (rs.next()){
```

```
    stu.setId(rs.getString("id"));
```

```
    stu.setName(rs.getString("name"));
```

```
}
```

id	studentname
1	Jim

```
Student stu = new Student();

String sql = "SELECT * FROM student WHERE id = ?";
PreparedStatement pstmt = conn.prepareStatement(sql);
pstmt.setString(1,"1");
//step5: 处理结果
rs = pstmt.executeQuery();

while (rs.next()){
    stu.setId(rs.getString("id"));
    stu.setName(rs.getString("name"));
}
```

这里就必须修改代码

跟着代码找答案

ORM方法论解决了数据持久层反复需要变更的问题

```
Student stu = new Student();
```

```
String sql = "SELECT * FROM student WHERE id = ?";
```

```
PreparedStatement pstmt = conn.prepareStatement(sql);
```

```
pstmt.setString(1,"1");
```

```
//step5: 处理结果
```

```
rs = pstmt.executeQuery();
```

```
while (rs.next()){
```

```
    stu.setId(rs.getString("id"));
```

```
    stu.setName(rs.getString("name"));
```

```
}
```

上层领域对象可能跟着需求发生变更,ORM框架通过映射配置的机制避免了领域层变更引发的维护成本

```
Student stu = new Student();
```

```
String sql = "SELECT * FROM student WHERE id = ?";
```

```
PreparedStatement pstmt = conn.prepareStatement(sql);
```

```
pstmt.setString(1,"1");
```

```
//step5: 处理结果
```

```
rs = pstmt.executeQuery();
```

```
while (rs.next()){
```

```
    stu.setId(rs.getString("id"));
```

```
    stu.setName(rs.getString("name"));
```

```
}
```

数据库的查询策略(sql语句)可能发生变化,将变化点抽出到配置文件中


```
Student stu = new Student();
```

```
String sql = "SELECT * FROM student WHERE id = ?";
```

```
PreparedStatement pstmt = conn.prepareStatement(sql);
```

```
pstmt.setString(1,"1");
```

```
//step5: 处理结果
```

```
rs = pstmt.executeQuery();
```

```
while (rs.next()){
```

```
    stu.setId(rs.getString("id"));
```

```
    stu.setName(rs.getString("name"));
```

```
}
```

变参注入必须去理解数据库表结构,ORM框架将其抽象为Bean的Set方法

```
Student stu = new Student();
```

```
String sql = "SELECT * FROM student WHERE id = ?";  
PreparedStatement pstmt = conn.prepareStatement(sql);
```

```
pstmt.setString(1,"1");
```

```
//step5: 处理结果
```

```
rs = pstmt.executeQuery();
```

```
while (rs.next()){  
    stu.setId(rs.getString("id"));  
    stu.setName(rs.getString("name"));  
}
```

程序必须去理解游标,要做关系模型和对象实例之间的转化,ORM通过映射配置解决了实体Bean的实例化和值注入问题

ORM框架是怎么解决这些问题的？

频度:高

难度:低

通过率:高

- 以最基本的形式建模数据。
- Java Bean
- 传达性 数据库结构被任何人都能理解的语言文档化。
- 精确性 基于数据模型创建正确标准化了的结构。
- 总结
- ORM中间件能在任何一个应用的业务逻辑层和数据库层之间充当桥梁
- 屏蔽了应用逻辑层和数据库之间因变更产生的差异传导.
- 存异是客观的,求同是我们的工程目标

面试题

你知道哪些主流的ORM框架

频度:高

难度:低

通过率:高

答案

Hibernate、mybatis

面试题

Hibernate与Mybatis对比,它们的区别是什么?

- Hibernate比mybatis更加重量级一些,开发难度高一些.
- Mybatis需要我们手动编写SQL语句, 回归最原始的方式, 所以可以按需求指定查询的字段, 提高程序的查询效率。
- Mybatis由于所有SQL都是依赖数据库书写的, 所以扩展性, 迁移性比较差。
- Hibernate与数据库具体的关联都在XML中, 所以HQL对具体是用什么数据库并不是很关心。

- **缓存机制对比**

- 相同点

- Hibernate和Mybatis的二级缓存除了采用系统默认的缓存机制外，都可以通过实现你自己的缓存或为其他第三方缓存方案，创建适配器来完全覆盖缓存行为。

- 不同点

- Hibernate的二级缓存配置在SessionFactory生成的配置文件中进行详细配置，然后再在具体的表-对象映射中配置是哪种缓存。
- MyBatis的二级缓存配置都是在每个具体的表-对象映射中进行详细配置，这样针对不同的表可以自定义不同的缓存机制。并且Mybatis可以在命名空间中共享相同的缓存配置和实例，通过Cache-ref来实现。

- 因为Hibernate对查询对象有着良好的管理机制，用户无需关心SQL。
- 所以在使用二级缓存时如果出现脏数据，系统会报出错误并提示。
- 而MyBatis在这一方面，使用二级缓存时需要特别小心。
- 如果不能完全确定数据更新操作的波及范围，避免Cache的盲目使用。
- 否则，脏数据的出现会给系统的正常运行带来很大的隐患。

- Hibernate的DAO层开发比MyBatis简单，Mybatis需要维护SQL和结果映射。
- Hibernate对对象的维护和缓存要比MyBatis好，对增删改查的对象的维护要方便。
- Hibernate数据库移植性很好，MyBatis的数据库移植性不好，不同的数据库需要写不同SQL。
- Hibernate有更好的二级缓存机制，可以使用第三方缓存。MyBatis本身提供的缓存机制不佳

- MyBatis可以进行更为细致的SQL优化，可以减少查询字段。
- MyBatis容易掌握，而Hibernate门槛较高。
- 实践篇
- 用Spring整合Mybatis实践一个简单的项目

面试题:在工程实践中,为什么常常需要mybatis和spring的整合?

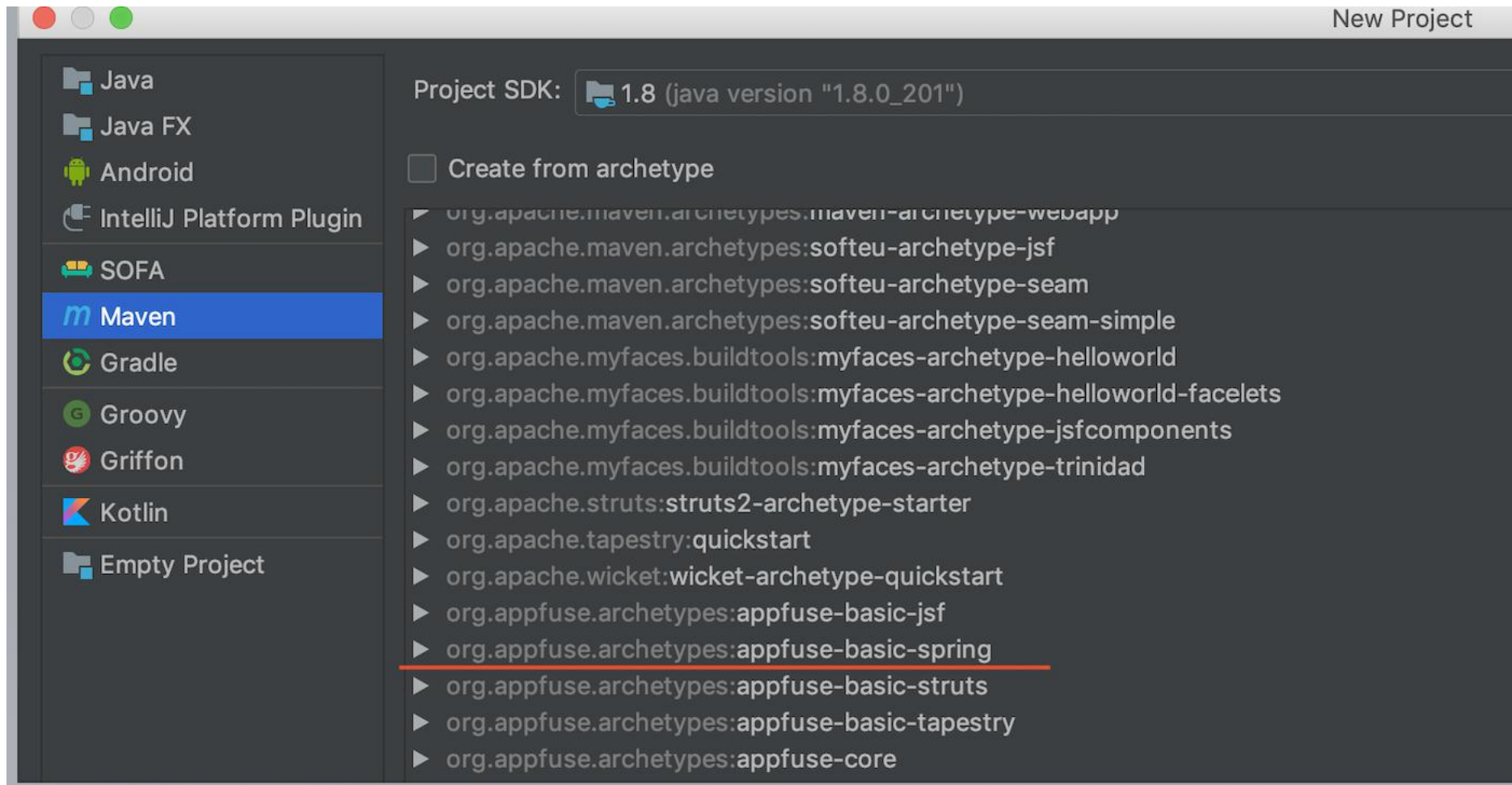
频度:高

难度:中

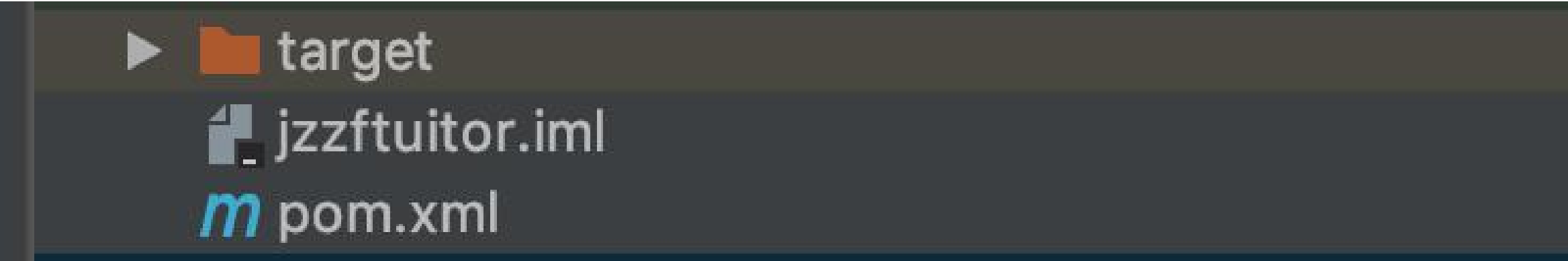
通过率:中

- spring框架在整合mybatis的时候，其实我们主要的目的就是要将原来本来应该由程序去调用mybatis代码产生Dao代理对象的过程交给了Spring;
 - MyBatis需要SqlSessionFactory对象交给Spring管理,解决了数据库会话、连接、查询实例的对象管理问题.
-
- 解析
 - 两个点
 - 管理Dao对象(Java Bean)
 - 管理SqlSessionFactory

- 用maven的能力构建一个spring项目框架



- maven其实是一个项目构建和管理的工具,主要就是提供了帮助管理 构建、文档、报告、依赖、scms、发布、分发的方法。
- 可以方便的编译代码、进行依赖管理、管理二进制库等的。



```
▶ target
  jzzftutor.iml
  m pom.xml
```

Pom文件

Maven中管理项目jar包依赖,发布版本,编译插件的配置

```
<properties>  
  <!-- spring版本号 -->  
  <spring.version>4.1.6.RELEASE</spring.version>  
  <!-- mybatis版本号 -->  
  <mybatis.version>3.2.6</mybatis.version>  
  <!-- log4j日志文件管理包版本 -->  
  <slf4j.version>1.7.7</slf4j.version>  
  <log4j.version>1.2.17</log4j.version>  
</properties>
```

```
<!-- spring核心包 -->  
<dependency>  
    <groupId>org.springframework</groupId>  
    <artifactId>spring-core</artifactId>  
    <version>${spring.version}</version>  
</dependency>
```

```
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-aop</artifactId>
  <version>${spring.version}</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-context-support</artifactId>
  <version>${spring.version}</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-test</artifactId>
  <version>${spring.version}</version>
</dependency>
```

```
<dependency>  
  <groupId>org.springframework</groupId>  
  <artifactId>spring-jdbc</artifactId>  
  <version>${spring.version}</version>  
</dependency>
```

```
<!-- mybatis核心包 -->  
<dependency>  
    <groupId>org.mybatis</groupId>  
    <artifactId>mybatis</artifactId>  
    <version>${mybatis.version}</version>  
</dependency>
```

```
<!-- mybatis/spring包 -->  
<dependency>  
  <groupId>org.mybatis</groupId>  
  <artifactId>mybatis-spring</artifactId>  
  <version>1.2.2</version>  
</dependency>
```



```
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>8.0.19</version>
</dependency>
<!-- c3p0连接池jar -->
<dependency>
  <groupId>c3p0</groupId>
  <artifactId>c3p0</artifactId>
  <version>0.9.1.2</version>
</dependency>
<!-- 导入dbcp的jar包，用来在applicationContext.xml中配置数据库 -->
<dependency>
  <groupId>commons-dbcp</groupId>
  <artifactId>commons-dbcp</artifactId>
  <version>1.2.2</version>
</dependency>
```

common-dbcp包

提供数据库连接池服务

C3P0包

C3P0是一个开源的[JDBC](#)连接池

```
public class Student implements Serializable {  
    private String id;  
    private String name;  
}
```

持久化Bean

通过orm框架将关系数据模型映射出的持久化数据模型

```
public interface StudentMapper {  
    public Student findStudentById(String id);  
}
```

Mapper接口

- Mapper接口开发方法只需要程序员编写Mapper接口
- 一个接口;一个Mapper配置文件

Mapper接口开发规范

- Mapper.xml文件中的**namespace**与**mapper接口的类路径**相同
- Mapper**接口方法名**和Mapper.xml中定义的**每个statement的id**相同
- Mapper接口方法的**输入参数类型**和mapper.xml中定义的每个sql的**parameterType**的类型相同
- Mapper接口方法的输出参数类型和mapper.xml中定义的每个sql的resultType的类型相同

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<!-- namespace: 命名空间, 对sql进行一个分类管理 -->
<!-- 注意: namespace在mapper代理时, 具有重要且特殊的作用
      对应mapper接口的全限定名
-->
<!--mybatis映射配置文件-->
<mapper namespace="com.essa.mapper.StudentMapper">
  <select id="findStudentById" parameterType="String" resultType="Student">
    select * from student where id=#{id}
  </select>
</mapper>
```

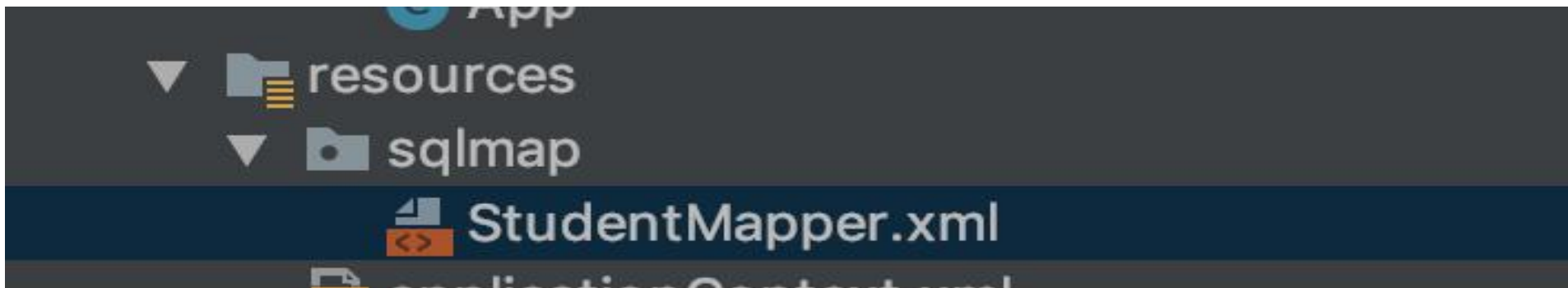

面试题:使用mybatis框架时 Mapper方式是怎么实现的

频度:中

难度:中

通过率:低

- Mybatis框架将根据接口定义创建接口的动态代理对象，代理对象的方法体实现Mapper接口中定义的方法。
- 何为动态代理
- 代理类在程序运行时创建的代理方式被成为动态代理
- 代理类并不是在Java代码中定义的，而是在运行时根据我们在Java代码中的“指示”动态生成的。



```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
    PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
    "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<!-- namespace: 命名空间, 对sql进行一个分类管理 -->
<!-- 注意: namespace在mapper代理时, 具有重要且特殊的作用
    对应mapper接口的全限定名
-->
<!--mybatis映射配置文件-->
<mapper namespace="com.essa.mapper.StudentMapper">
    <select id="findStudentById" parameterType="String" resultType="Student">
        select * from student where id=#{id}
    </select>
</mapper>
```

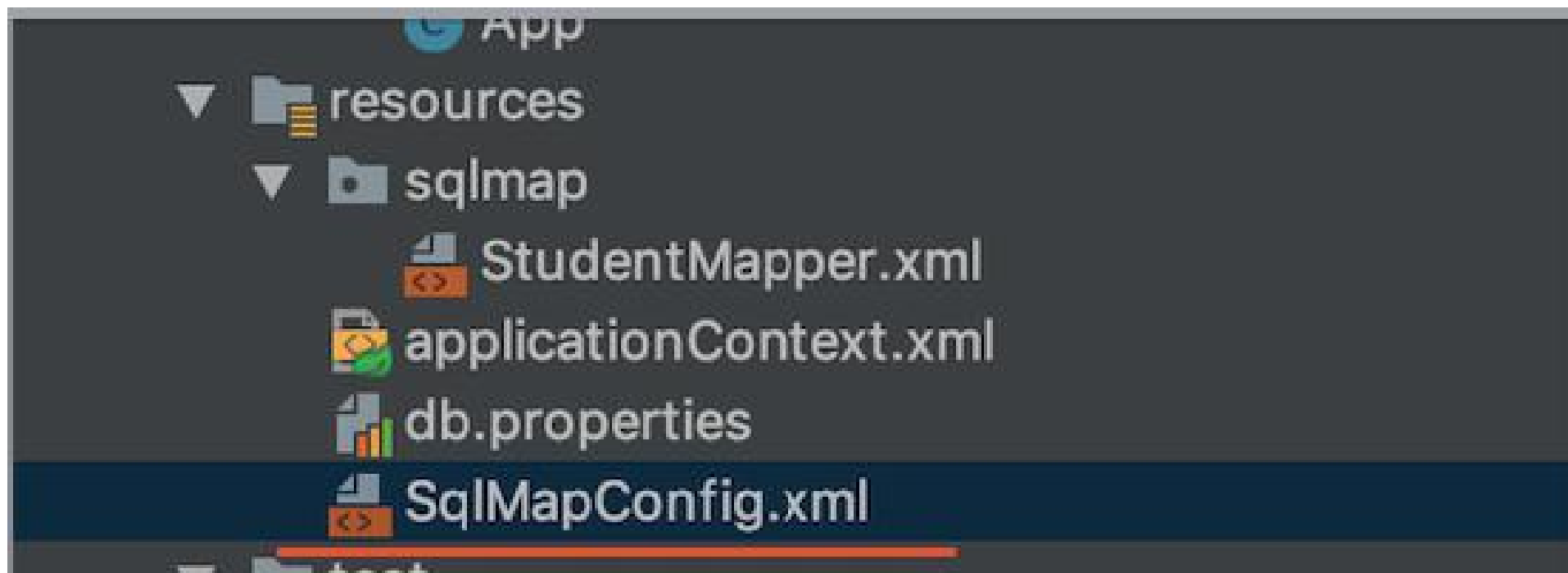
```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
    PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
    "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<!-- namespace: 命名空间, 对sql进行一个分类管理 -->
<!-- 注意: namespace在mapper代理时, 具有重要且特殊的作用
    对应mapper接口的全限定名
-->

<!--mybatis映射配置文件-->
<mapper namespace="com.essa.mapper.StudentMapper">
    <select id="findStudentById" parameterType="String" resultType="Student">
        select * from student where id=#{id}
    </select>
</mapper>
```

同样的命名空间

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
    PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
    "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<!-- namespace: 命名空间, 对sql进行一个分类管理 -->
<!-- 注意: namespace在mapper代理时, 具有重要且特殊的作用
    对应mapper接口的全限定名
-->
<!--mybatis映射配置文件-->
<mapper namespace="com.essa.mapper.StudentMapper">
    <select id="findStudentById" parameterType="String" resultType="Student">
        select * from student where id=#{id}
    </select>
</mapper>
```

同样的statementId、参数、返回
类型



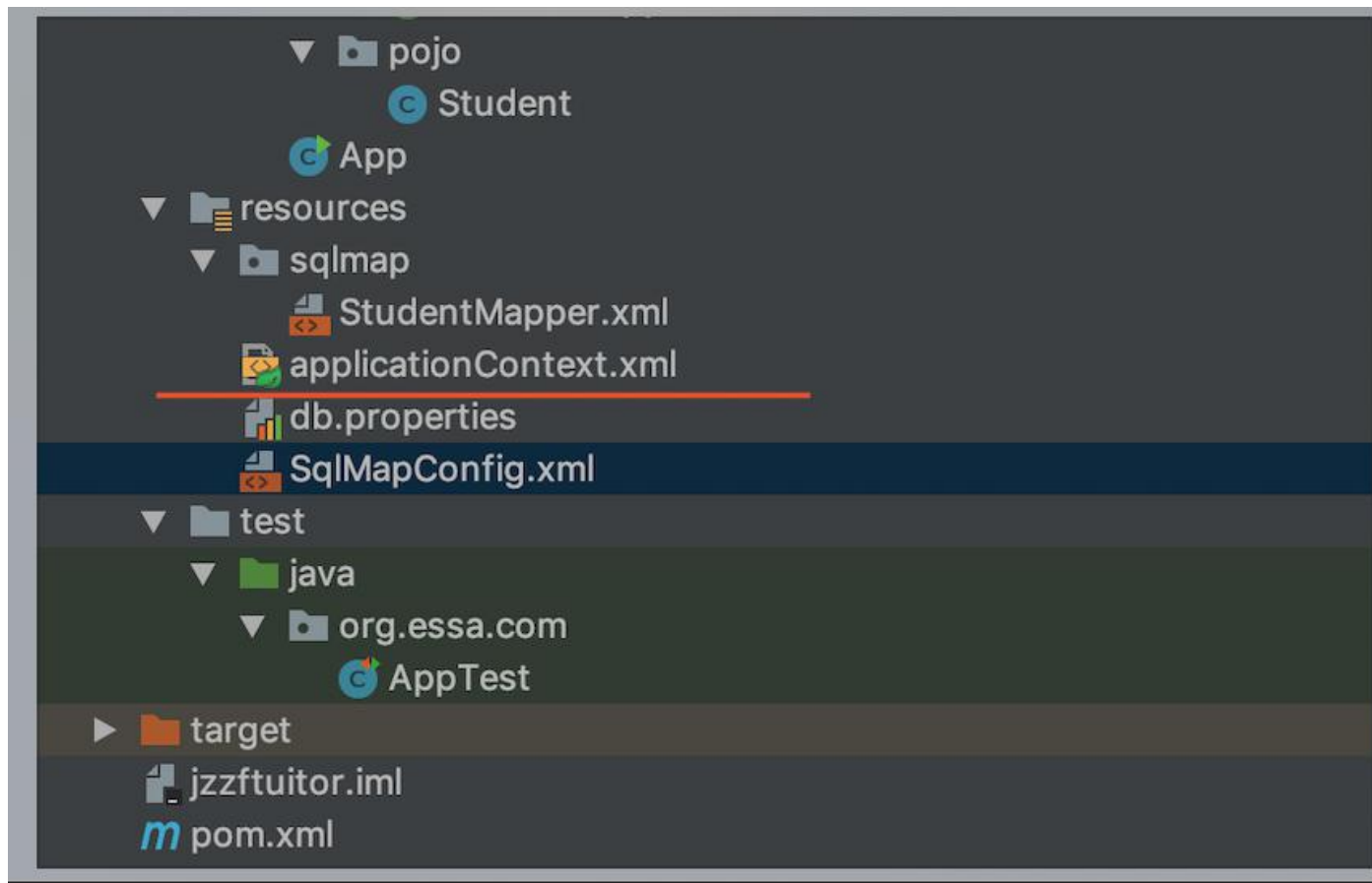
```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
  <!--mybatis核心配置文件-->

  <!--给类取别名，简化输入，方便映射配置文件中使用-->
  <typeAliases>
    <typeAlias type="com.essa.pojo.Student" alias="student"/>
  </typeAliases>

  <!--加载mapper映射配置文件-->
  <mappers>
    <mapper resource="sqlmap/StudentMapper.xml"/>
  </mappers>
</configuration>
```

指定要加载的mapper
文件列表

- applicaitonContext.xml

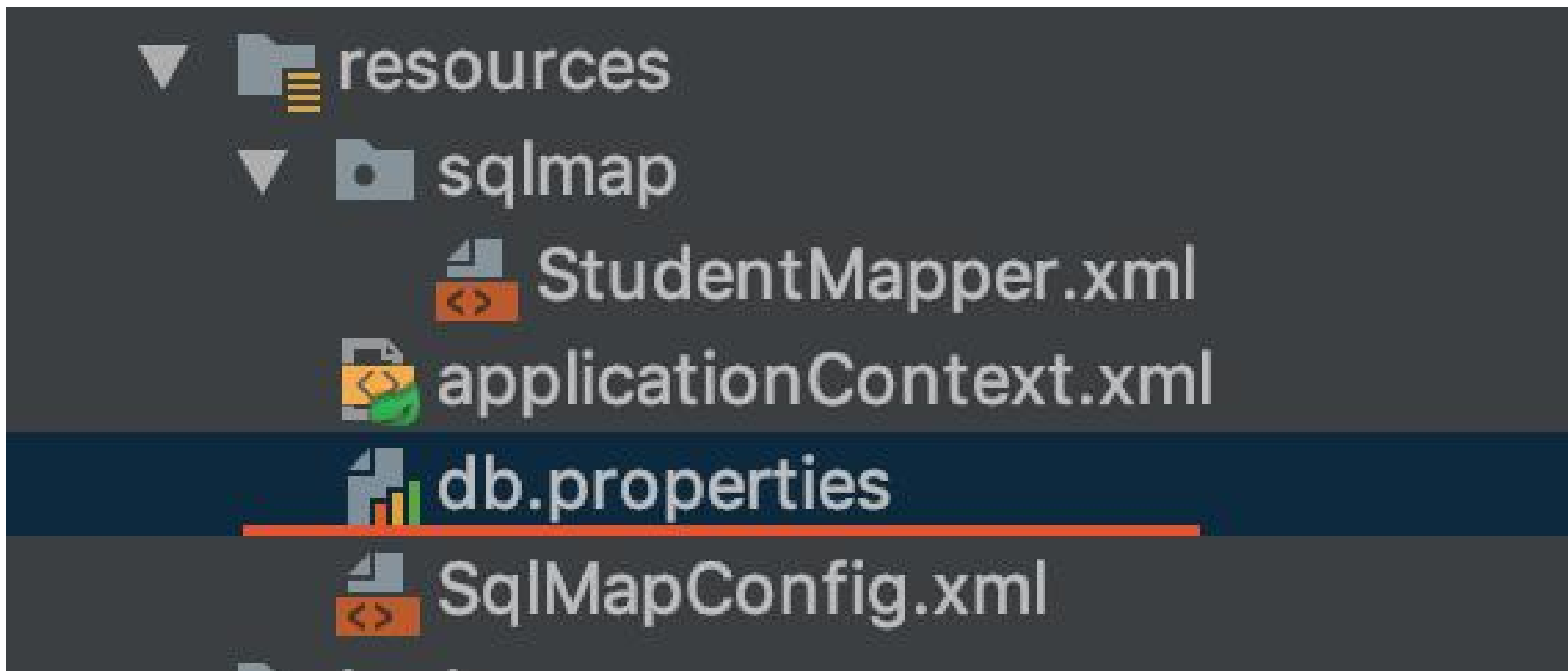


Spring applicationContext.xml中托管mybatis核心类

```
<!-- 配置数据源 -->
<bean id="dataSource" class="${dataSource}" destroy-method="close">
    <property name="driverClassName" value="${db.driver}" />
    <property name="url" value="${db.url}" />
    <property name="username" value="${db.username}" />
    <property name="password" value="${db.password}" />
    <property name="maxActive" value="10" />
    <property name="maxIdle" value="5" />
</bean>

<!-- 配置sqlSessionFactory, SqlSessionFactoryBean是用来产生sqlSessionFactory的 -->
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
    <!-- 加载mybatis的全局配置文件-->
    <property name="configLocation" value="SqlMapConfig.xml" />
    <!-- 加载数据源, 使用上面配置好的数据源 -->
    <property name="dataSource" ref="dataSource" />
</bean>

<!-- MapperFactoryBean: 根据mapper接口生成的代理对象 -->
<bean id="studentMapper" class="org.mybatis.spring.mapper.MapperFactoryBean">
    <property name="mapperInterface" value="com.essa.mapper.StudentMapper"/>
    <property name="sqlSessionFactory" ref="sqlSessionFactory" />
</bean>
```

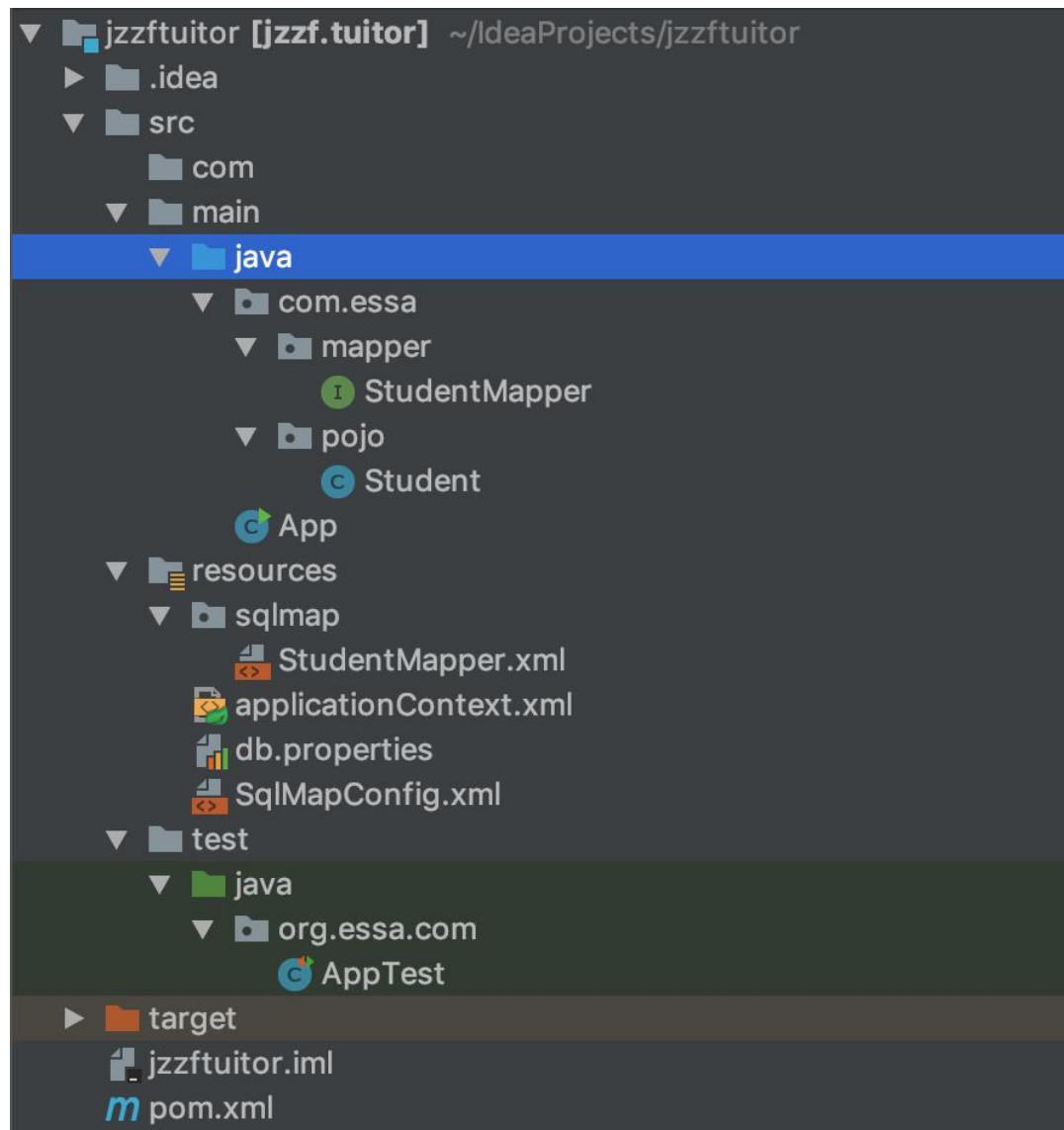


```
dataSource=org.apache.commons.dbcp.BasicDataSource  
db.driver=com.mysql.cj.jdbc.Driver  
#数据库连接字符串(改成自己的连接)  
db.url=jdbc:mysql://localhost:3306/studb  
#数据库用户名(这里改成自己的用户名)  
db.username=cup  
#数据库密码(这里改成自己的密码)  
db.password=cup
```

```
public class AppTest
{
    /**
     * Rigorous Test :-)
     */
    @Test
    public void test01() {
        //获取applicationContext文件并加载
        ApplicationContext ac = new ClassPathXmlApplicationContext( configLocation: "applicationContext.xml");
        //获取StudentDao的bean
        StudentMapper sd = (StudentMapper) ac.getBean( s: "studentMapper");
        Student s = sd.findStudentById("1");
        System.out.println("学生姓名: " + s.getName());
        System.out.println("学生id: " + s.getId());
    }
}
```

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_201.jdk/Contents/Home/bin/java ...  
四月 03, 2020 2:33:19 上午 org.springframework.context.support.ClassPathXmlApplicationContext prepareRefresh  
信息: Refreshing org.springframework.context.support.ClassPathXmlApplicationContext@6d86b085: startup date [Fri Apr  
四月 03, 2020 2:33:19 上午 org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions  
信息: Loading XML bean definitions from class path resource [applicationContext.xml]  
四月 03, 2020 2:33:20 上午 org.springframework.context.support.PropertySourcesPlaceholderConfigurer loadProperties  
信息: Loading properties file from class path resource [db.properties]  
学生姓名: Jim  
学生id: 1  
  
Process finished with exit code 0
```

hinal 智能研发 LinkE



- 本次在前两讲的基础上进行工程实操演练
- 通过0-1的过程从原理上理解orm框架的场景和原理
- 主要解决面试问题中数据应用相关的原理性问题和工程性问题
- 面对这样的问题,只有亲自做一遍,才能真正应对面试,背答案是很容易被识破的.