### 1、什么是Mybatis？

1、Mybatis 是一个半 ORM（ 对象关系映射）框架，它内部封装了 JDBC，开发时只需要关注 SQL 语句本身， 不需要花费精力去处理加载驱动、创建连接、创建

statement 等繁杂的过程。程序员直接编写原生态 sql，可以严格控制 sql 执行性能， 灵活度高。

2、MyBatis 可以使用 XML 或注解来配置和映射原生信息， 将 POJO 映射成数据库中的记录， 避免了几乎所有的 JDBC 代码和手动设置参数以及获取结果集。

3、通过 xml 文件或注解的方式将要执行的各种 statement 配置起来， 并通过java 对象和 statement 中 sql 的动态参数进行映射生成最终执行的 sql 语句，最后由 mybatis 框架执行 sql 并将结果映射为 java 对象并返回。（ 从执行 sql 到返回 result 的过程）。

### 2、Mybaits 的优点：

1、基于 SQL 语句编程，相当灵活，不会对应用程序或者数据库的现有设计造成任何影响，SQL 写在 XML 里，解除 sql 与程序代码的耦合，便于统一管理；提供 XML 标签， 支持编写动态 SQL 语句， 并可重用。

2、与 JDBC 相比，减少了 50%以上的代码量，消除了 JDBC 大量冗余的代码，不需要手动开关连接；

3、很好的与各种数据库兼容（ 因为 MyBatis 使用 JDBC 来连接数据库，所以只要JDBC 支持的数据库 MyBatis 都支持）。

4、能够与 Spring 很好的集成；

5、提供映射标签， 支持对象与数据库的 ORM 字段关系映射； 提供对象关系映射标签， 支持对象关系组件维护。

### 3、MyBatis 框架的缺点：

1、SQL 语句的编写工作量较大， 尤其当字段多、关联表多时， 对开发人员编写SQL 语句的功底有一定要求。

2、SQL 语句依赖于数据库， 导致数据库移植性差， 不能随意更换数据库。

### 4、MyBatis 框架适用场合：

1、MyBatis 专注于 SQL 本身， 是一个足够灵活的 DAO 层解决方案。

2、对性能的要求很高，或者需求变化较多的项目，如互联网项目， MyBatis 将是不错的选择。

### 5、MyBatis 与Hibernate 有哪些不同？

1、Mybatis 和 hibernate 不同，它不完全是一个 ORM 框架，因为 MyBatis 需要程序员自己编写 Sql 语句。

2、Mybatis 直接编写原生态 sql， 可以严格控制 sql 执行性能， 灵活度高， 非常适合对关系数据模型要求不高的软件开发， 因为这类软件需求变化频繁， 一但需求变化要求迅速输出成果。但是灵活的前提是 mybatis 无法做到数据库无关性， 如果需要实现支持多种数据库的软件，则需要自定义多套 sql 映射文件，工作量大。

3、Hibernate 对象/关系映射能力强， 数据库无关性好， 对于关系模型要求高的软件， 如果用 hibernate 开发可以节省很多代码， 提高效率。

### 6、#{}和${}的区别是什么？

#{}是预编译处理， ${}是字符串替换。

Mybatis 在处理#{}时，会将 sql 中的#{}替换为?号，调用 PreparedStatement 的set 方法来赋值；

Mybatis 在处理${}时， 就是把${}替换成变量的值。使用#{}可以有效的防止 SQL 注入， 提高系统安全性。

### 7、当实体类中的属性名和表中的字段名不一样 ，怎么办 ？

第 1 种： 通过在查询的 sql 语句中定义字段名的别名， 让字段名的别名和实体类的属性名一致。

<select id=”selectorder” parametertype=”int” resultetype=” me.gacl.domain.order”>

select order\_id id, order\_no orderno ,order\_price price form orders where order\_id=#{id};

</select>

第 2 种： 通过<resultMap>来映射字段名和实体类属性名的一一对应的关系。

<select id="getOrder" parameterType="int" resultMap="orderresultmap">

select \* from orders where order\_id=#{id}

</select>

<resultMap type=”me.gacl.domain.order” id=”orderresultmap”>

<!–用 id 属性来映射主键字段–>

<id property=”id” column=”order\_id”>

<!–用 result 属性来映射非主键字段，property 为实体类属性名，column 为数据表中的属性–>

<result property = “orderno” column =”order\_no”/>

<result property=”price” column=”order\_price” />

</reslutMap>

### 8、 模糊查询like 语句该怎么写?

第 1 种： 在 Java 代码中添加 sql 通配符。

string wildcardname = “%smi%”;

list<name> names = mapper.selectlike(wildcardname);

<select id=”selectlike”>

select \* from foo where bar like #{value}

</select>

第 2 种： 在 sql 语句中拼接通配符， 会引起 sql 注入

string wildcardname = “smi”;

list<name> names = mapper.selectlike(wildcardname);

<select id=”selectlike”>

select \* from foo where bar like "%"#{value}"%"

</select>

### 9、通常一个Xml 映射文件，都会写一个Dao 接口与之对应，

请问，这个Dao 接口的工作原理是什么？Dao 接口里的方法， 参数不同时，方法能重载吗？

Dao 接口即 Mapper 接口。接口的全限名，就是映射文件中的 namespace 的值； 接口的方法名， 就是映射文件中 Mapper 的 Statement 的 id 值； 接口方法内的参数， 就是传递给 sql 的参数。

Mapper 接口是没有实现类的，当调用接口方法时，接口全限名+方法名拼接字符串作为 key 值， 可唯一定位一个 MapperStatement。在 Mybatis 中， 每一个

<select>、<insert>、<update>、<delete>标签， 都会被解析为一个

MapperStatement 对象。

举例： com.mybatis3.mappers.StudentDao.findStudentById， 可以唯一 找 到 namespace 为 com.mybatis3.mappers.StudentDao 下 面 id 为findStudentById 的 MapperStatement 。

Mapper 接口里的方法，是不能重载的，因为是使用 全限名+方法名 的保存和寻找策略。Mapper 接口的工作原理是 JDK 动态代理， Mybatis 运行时会使用 JDK 动态代理为 Mapper 接口生成代理对象 proxy， 代理对象会拦截接口方法， 转而执行 MapperStatement 所代表的 sql， 然后将 sql 执行结果返回。

### 10、Mybatis 是如何进行分页的？分页插件的原理是什么？

Mybatis 使用 RowBounds 对象进行分页， 它是针对 ResultSet 结果集执行的内存分页，而非物理分页。可以在 sql 内直接书写带有物理分页的参数来完成物理分页功能， 也可以使用分页插件来完成物理分页。

分页插件的基本原理是使用 Mybatis 提供的插件接口， 实现自定义插件， 在插件的拦截方法内拦截待执行的 sql，然后重写 sql，根据 dialect 方言，添加对应的物理分页语句和物理分页参数。

### 11、Mybatis 是如何将sql 执行结果封装为目标对象并返回的？ 都有哪些映射形式？

第一种是使用<resultMap>标签， 逐一定义数据库列名和对象属性名之间的映射关系。

第二种是使用 sql 列的别名功能， 将列的别名书写为对象属性名。

有了列名与属性名的映射关系后， Mybatis 通过反射创建对象， 同时使用反射给对象的属性逐一赋值并返回， 那些找不到映射关系的属性， 是无法完成赋值的。

### 12、如何执行批量插入?

首先,创建一个简单的 insert 语句:

<insert id=”insertname”>

insert into names (name) values (#{value})

</insert>

然后在 java 代码中像下面这样执行批处理插入:

list < string > names = new arraylist(); names.add(“fred”); names.add(“barney”); names.add(“betty”);

names.add(“wilma”);

// 注意这里 executortype.batch sqlsession sqlsession =

sqlsessionfactory.opensession(executortype.batch); try {

namemapper mapper = sqlsession.getmapper(namemapper.class); for (string name: names) {

mapper.insertname(name);

}

sqlsession.commit();

} catch (Exception e) { e.printStackTrace(); sqlSession.rollback();

throw e;

}

finally {

sqlsession.close();

}

### 13、如何获取自动生成的(主)键值?

insert 方法总是返回一个 int 值 ， 这个值代表的是插入的行数。

如果采用自增长策略，自动生成的键值在 insert 方法执行完后可以被设置到传入的参数对象中。

示例：

<insert id=”insertname” usegeneratedkeys=”true” keyproperty=” id”>

insert into names (name) values (#{name})

</insert>

name name = new name(); name.setname(“fred”);

int rows = mapper.insertname(name);

// 完成后,id 已经被设置到对象中system.out.println(“rows inserted = ” + rows);

system.out.println(“generated key value = ” + name.getid());

### 14、在 mapper 中如何传递多个参数?

1、第一种： DAO 层的函数

public UserselectUser(String name,String area);

对应的 xml,#{0}代表接收的是 dao 层中的第一个参数，#{1}代表 dao 层中第二参数，更多参数一致往后加即可。

<select id="selectUser"resultMap="BaseResultMap"> select \* fromuser\_user\_t whereuser\_name = #{0}

anduser\_area=#{1}

</select>

2、第二种： 使用 [@param](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D) 注解:

public interface usermapper {

user selectuser(@param(“username”) string username,@param(“hashedpassword”) string hashedpassword);

}

然后,就可以在 xml 像下面这样使用(推荐封装为一个 map,作为单个参数传递给mapper):

<select id=”selectuser” resulttype=”user”> select id, username, hashedpassword from some\_table

where username = #{username}

and hashedpassword = #{hashedpassword}

</select>

3、第三种： 多个参数封装成 map

try {

//映射文件的命名空间.SQL 片段的 ID，就可以调用对应的映射文件中的

SQL

//由于我们的参数超过了两个，而方法中只有一个 Object 参数收集，因此我们使用 Map 集合来装载我们的参数

Map < String, Object > map = new HashMap(); map.put("start", start);

map.put("end", end);

return sqlSession.selectList("StudentID.pagination", map);

} catch (Exception e) { e.printStackTrace(); sqlSession.rollback(); throw e;

} finally {

MybatisUtil.closeSqlSession();

}

### 15、Mybatis 动态sql 有什么用？执行原理？有哪些动态sql？

Mybatis 动态 sql 可以在 Xml 映射文件内，以标签的形式编写动态 sql，执行原理是根据表达式的值 完成逻辑判断并动态拼接 sql 的功能。

Mybatis 提供了 9 种动态 sql 标签：trim | where | set | foreach | if | choose

#### | when | otherwise | bind 。

16、Xml 映射文件中，除了常见的 select|insert|updae|delete 标签之外，还有哪些标签？

答： <resultMap>、<parameterMap>、<sql>、<include>、

<selectKey>， 加上动态 sql 的 9 个标签， 其中<sql>为 sql 片段标签， 通过

<include>标签引入 sql 片段，<selectKey>为不支持自增的主键生成策略标签。

17、Mybatis 的 Xml 映射文件中， 不同的 Xml 映射文件， id 是否可以重复？

不同的 Xml 映射文件， 如果配置了 namespace， 那么 id 可以重复； 如果没有配置 namespace， 那么 id 不能重复；

原因就是 namespace+id 是作为 Map<String, MapperStatement>的 key 使用的， 如果没有 namespace， 就剩下 id， 那么， id 重复会导致数据互相覆盖。有了 namespace，自然 id 就可以重复，namespace 不同，namespace+id 自然也就不同。

### 18、为什么说 Mybatis 是半自动 ORM 映射工具？它与全自动的区别在哪里？

Hibernate 属于全自动 ORM 映射工具， 使用 Hibernate 查询关联对象或者关联集合对象时， 可以根据对象关系模型直接获取， 所以它是全自动的。而 Mybatis 在查询关联对象或关联集合对象时，需要手动编写 sql 来完成，所以，称之为半自动 ORM 映射工具。

### 19、 一对一、一对多的关联查询 ？

<mapper namespace="com.lcb.mapping.userMapper">

<!--association 一对一关联查询 -->

#### <select id="getClass" parameterType="int" resultMap="ClassesResultMap">

select \* from class c,teacher t where c.teacher\_id=t.t\_id and c.c\_id=#{id}

</select>

<resultMap type="com.lcb.user.Classes" id="ClassesResultMap">

<!-- 实体类的字段名和数据表的字段名映射 -->

<id property="id" column="c\_id"/>

<result property="name" column="c\_name"/>

<association property="teacher" javaType="com.lcb.user.Teacher">

<id property="id" column="t\_id"/>

<result property="name" column="t\_name"/>

</association>

</resultMap>

<!--collection 一对多关联查询 -->

<select id="getClass2" parameterType="int" resultMap="ClassesResultMap2">

select \* from class c,teacher t,student s where c.teacher\_id=t.t\_id and c.c\_id=s.class\_id and c.c\_id=#{id}

</select>

<resultMap type="com.lcb.user.Classes" id="ClassesResultMap2">

<id property="id" column="c\_id"/>

<result property="name" column="c\_name"/>

<association property="teacher" javaType="com.lcb.user.Teacher">

<id property="id" column="t\_id"/>

<result property="name" column="t\_name"/>

</association>

<collection property="student" ofType="com.lcb.user.Student">

<id property="id" column="s\_id"/>

<result property="name" column="s\_name"/>

</collection>

</resultMap>

</mapper>

20、MyBatis 实现一对一有几种方式?具体怎么操作的？

有联合查询和嵌套查询,联合查询是几个表联合查询,只查询一次, 通过在resultMap 里面配置 association 节点配置一对一的类就可以完成；

嵌套查询是先查一个表，根据这个表里面的结果的 外键 id，去再另外一个表里面查询数据,也是通过 association 配置，但另外一个表的查询通过 select 属性配置。

### 21、MyBatis 实现一对多有几种方式,怎么操作的？

有联合查询和嵌套查询。联合查询是几个表联合查询,只查询一次,通过在

resultMap 里面的 collection 节点配置一对多的类就可以完成； 嵌套查询是先查一个表,根据这个表里面的 结果的外键 id,去再另外一个表里面查询数据,也是通过配置 collection,但另外一个表的查询通过 select 节点配置。

### 22、Mybatis 是否支持延迟加载？如果支持，它的实现原理是

什么？

答： Mybatis 仅支持 association 关联对象和 collection 关联集合对象的延迟加载， association 指的就是一对一， collection 指的就是一对多查询。在 Mybatis 配置文件中， 可以配置是否启用延迟加载 lazyLoadingEnabled=true|false。

它的原理是， 使用 CGLIB 创建目标对象的代理对象， 当调用目标方法时， 进入拦截器方法， 比如调用 a.getB().getName()， 拦截器 invoke()方法发现 a.getB()是null 值， 那么就会单独发送事先保存好的查询关联 B 对象的 sql， 把 B 查询上来， 然后调用 a.setB(b)，于是 a 的对象 b 属性就有值了，接着完成 a.getB().getName()方法的调用。这就是延迟加载的基本原理。

当然了， 不光是 Mybatis， 几乎所有的包括 Hibernate， 支持延迟加载的原理都是一样的。

### 23、Mybatis 的一级、二级缓存:

1. 一级缓存: 基于 PerpetualCache 的 HashMap 本地缓存， 其存储作用域为Session， 当 Session flush 或 close 之后， 该 Session 中的所有 Cache 就将清空， 默认打开一级缓存。
2. 二级缓存与一级缓存其机制相同，默认也是采用 PerpetualCache，HashMap 存储， 不同在于其存储作用域为 Mapper(Namespace)， 并且可自定义存储源， 如 Ehcache。默认不打开二级缓存， 要开启二级缓存， 使用二级缓存属性类需要实现 Serializable 序列化接口(可用来保存对象的状态),可在它的映射文件中配置

#### <cache/> ；

1. 对于缓存数据更新机制， 当某一个作用域(一级缓存 Session/二级缓存

Namespaces)的进行了 C/U/D 操作后，默认该作用域下所有 select 中的缓存将被 clear 。

### 24、什么是 MyBatis 的接口绑定？有哪些实现方式？

接口绑定，就是在 MyBatis 中任意定义接口,然后把接口里面的方法和 SQL 语句绑定, 我们直接调用接口方法就可以,这样比起原来了 SqlSession 提供的方法我们可以有更加灵活的选择和设置。

接口绑定有两种实现方式,一种是通过注解绑定， 就是在接口的方法上面加上@Select、@Update 等注解， 里面包含 Sql 语句来绑定； 另外一种就是通过 xml 里面写 SQL 来绑定, 在这种情况下,要指定 xml 映射文件里面的 namespace 必须为接口的全路径名。当 Sql 语句比较简单时候,用注解绑定, 当 SQL 语句比较复杂时候,用 xml 绑定,一般用 xml 绑定的比较多。

### 25、使用 MyBatis 的mapper 接口调用时有哪些要求？

1、Mapper 接口方法名和 mapper.xml 中定义的每个 sql 的 id 相同；

2、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；

3、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；

4、Mapper.xml 文件中的 namespace 即是 mapper 接口的类路径。

### 26、Mapper 编写有哪几种方式？

第一种： 接口实现类继承 SqlSessionDaoSupport： 使用此种方法需要编写mapper 接口， mapper 接口实现类、mapper.xml 文件。

1、在 sqlMapConfig.xml 中配置 mapper.xml 的位置

<mappers>

<mapper resource="mapper.xml 文件的地址" />

<mapper resource="mapper.xml 文件的地址" />

</mappers>

1、定义 mapper 接口

3、实现类集成 SqlSessionDaoSupport

mapper 方法中可以 this.getSqlSession()进行数据增删改查。4、spring 配置

<bean id=" " class="mapper 接口的实现">

<property name="sqlSessionFactory" ref="sqlSessionFactory"></property>

</bean>

#### 第二种： 使用 org.mybatis.spring.mapper.MapperFactoryBean：

1、在 sqlMapConfig.xml 中配置 mapper.xml 的位置， 如果 mapper.xml 和mappre 接口的名称相同且在同一个目录， 这里可以不用配置

<mappers>

<mapper resource="mapper.xml 文件的地址" />

<mapper resource="mapper.xml 文件的地址" />

</mappers>

2、定义 mapper 接口：

1、mapper.xml 中的 namespace 为 mapper 接口的地址

2、mapper 接口中的方法名和 mapper.xml 中的定义的 statement 的 id 保持一致

3、Spring 中定义

<bean id="" class="org.mybatis.spring.mapper.MapperFactoryBean">

<property name="mapperInterface" value="mapper 接口地址" />

<property name="sqlSessionFactory" ref="sqlSessionFactory" />

</bean>

第三种： 使用 mapper 扫描器： 1、mapper.xml 文件编写：

mapper.xml 中的 namespace 为 mapper 接口的地址；

mapper 接口中的方法名和 mapper.xml 中的定义的 statement 的 id 保持一致； 如果将 mapper.xml 和 mapper 接口的名称保持一致则不用在 sqlMapConfig.xml 中进行配置。

2、定义 mapper 接口：

注意 mapper.xml 的文件名和 mapper 的接口名称保持一致， 且放在同一个目录3、配置 mapper 扫描器：

<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">

<property name="basePackage" value="mapper 接口包地址"></property>

<property name="sqlSessionFactoryBeanName" value="sqlSessionFactory"/>

</bean>

4、使用扫描器后从 spring 容器中获取 mapper 的实现对象。

### 27、简述 Mybatis 的插件运行原理，以及如何编写一个插件。

答： Mybatis 仅可以编写针对 ParameterHandler、ResultSetHandler、

StatementHandler、Executor 这 4 种接口的插件， Mybatis 使用 JDK 的动态代理， 为需要拦截的接口生成代理对象以实现接口方法拦截功能， 每当执行这 4 种接口对象的方法时，就会进入拦截方法，具体就是 InvocationHandler 的 invoke() 方法， 当然， 只会拦截那些你指定需要拦截的方法。

编写插件： 实现 Mybatis 的 Interceptor 接口并复写 intercept()方法， 然后在给插件编写注解， 指定要拦截哪一个接口的哪些方法即可， 记住， 别忘了在配置文件中配置你编写的插件。

## 源码分析专题-Spring 面试题

### 1、什么是spring?

Spring 是个 java 企业级应用的开源开发框架。Spring 主要用来开发 Java 应用， 但是有些扩展是针对构建 J2EE 平台的 web 应用。Spring 框架目标是简化 Java 企业级应用开发， 并通过 POJO 为基础的编程模型促进良好的编程习惯。

### 2、使用Spring 框架的好处是什么？

#### 轻量：Spring 是轻量的，基本的版本大约 2MB。

* 控制反转：Spring 通过控制反转实现了松散耦合，对象们给出它们的依赖，而不是创建或查找依赖的对象们。
* 面向切面的编程(AOP)：Spring 支持面向切面的编程，并且把应用业务

逻辑和系统服务分开。

* 容器：Spring 包含并管理应用中对象的生命周期和配置。
* MVC 框架：Spring 的 WEB 框架是个精心设计的框架，是 Web 框架的一个很好的替代品。
* 事务管理：Spring 提供一个持续的事务管理接口，可以扩展到上至本地事务下至全局事务（JTA）。
* 异常处理：Spring 提供方便的API 把具体技术相关的异常（比如由JDBC， Hibernate or JDO 抛出的）转化为一致的 unchecked 异常。

### 3、Spring 由哪些模块组成?

以下是 Spring 框架的基本模块：

#### Core module

* Bean module
* Context module
* Expression Language module
* JDBC module
* ORM module
* OXM module
* Java Messaging Service(JMS) module
* Transaction module
* Web module
* Web-Servlet module
* Web-Struts module
* Web-Portlet module

### 4、核心容器（应用上下文) 模块。

这是基本的 Spring 模块， 提供 spring 框架的基础功能， BeanFactory 是 任何以 spring 为基础的应用的核心。Spring 框架建立在此模块之上，它使 Spring 成为一个容器。

### 5、BeanFactory – BeanFactory 实现举例。

Bean 工厂是工厂模式的一个实现，提供了控制反转功能，用来把应用的配置和依赖从正真的应用代码中分离。

最常用的 BeanFactory 实现是 XmlBeanFactory 类。

### 6、XMLBeanFactory

最常用的就是 org.springframework.beans.factory.xml.XmlBeanFactory ， 它根据 XML 文件中的定义加载 beans。该容器从 XML 文件读取配置元数据并用它去创建一个完全配置的系统或应用。

### 7、解释AOP 模块

AOP 模块用于发给我们的 Spring 应用做面向切面的开发， 很多支持由 AOP 联盟提供， 这样就确保了 Spring 和其他 AOP 框架的共通性。这个模块将元数据编程引入 Spring。

### 8、解释JDBC 抽象和DAO 模块。

通过使用 JDBC 抽象和 DAO 模块，保证数据库代码的简洁，并能避免数据库资源错误关闭导致的问题， 它在各种不同的数据库的错误信息之上， 提供了一个统一的异常访问层。它还利用 Spring 的 AOP 模块给 Spring 应用中的对象提供事务管理服务。

### 9、解释对象/关系映射集成模块。

Spring 通过提供 ORM 模块，支持我们在直接 JDBC 之上使用一个对象/关系映射映射(ORM)工具，Spring 支持集成主流的 ORM 框架，如 Hiberate,JDO 和 iBATIS SQL Maps。Spring 的事务管理同样支持以上所有 ORM 框架及 JDBC。

### 10、解释 WEB 模块。

Spring 的 WEB 模块是构建在 application context 模块基础之上， 提供一个适合 web 应用的上下文。这个模块也包括支持多种面向 web 的任务，如透明地处理多个文件上传请求和程序级请求参数的绑定到你的业务对象。它也有对 Jakarta Struts 的支持。

### 12、Spring 配置文件

Spring 配置文件是个 XML 文件，这个文件包含了类信息，描述了如何配置它们， 以及如何相互调用。

### 13、什么是 Spring IOC 容器？

Spring IOC 负责创建对象，管理对象（ 通过依赖注入（ DI），装配对象，配置对象， 并且管理这些对象的整个生命周期。

### 14、IOC 的优点是什么？

IOC 或 依赖注入把应用的代码量降到最低。它使应用容易测试，单元测试不再需要单例和 JNDI 查找机制。最小的代价和最小的侵入性使松散耦合得以实现。IOC 容器支持加载服务时的饿汉式初始化和懒加载。

### 15、ApplicationContext 通常的实现是什么?

#### FileSystemXmlApplicationContext ：此容器从一个 XML 文件中加载 beans 的定义，XML Bean 配置文件的全路径名必须提供给它的构造函数。

* ClassPathXmlApplicationContext：此容器也从一个 XML 文件中加载beans 的定义，这里，你需要正确设置 classpath 因为这个容器将在classpath 里找 bean 配置。
* WebXmlApplicationContext：此容器加载一个 XML 文件，此文件定

义了一个 WEB 应用的所有 bean。

### 16、Bean 工厂和 Application contexts 有什么区别？

Application contexts 提供一种方法处理文本消息， 一个通常的做法是加载文件资源（ 比如镜像），它们可以向注册为监听器的 bean 发布事件。另外，在容器或容器内的对象上执行的那些不得不由 bean 工厂以程序化方式处理的操作，可以在Application contexts 中以声明的方式处理。Application contexts 实现了

MessageSource 接口， 该接口的实现以可插拔的方式提供获取本地化消息的方法。

### 17、一个 Spring 的应用看起来象什么？

#### 一个定义了一些功能的接口。

* 这实现包括属性，它的 Setter ， getter 方法和函数等。
* Spring AOP。
* Spring 的 XML 配置文件。
* 使用以上功能的客户端程序。

### 依赖注入

### 18、什么是 Spring 的依赖注入？

依赖注入， 是 IOC 的一个方面， 是个通常的概念， 它有多种解释。这概念是说你不用创建对象， 而只需要描述它如何被创建。你不在代码里直接组装你的组件和服务， 但是要在配置文件里描述哪些组件需要哪些服务， 之后一个容器（ IOC 容器） 负责把他们组装起来。

### 19、有哪些不同类型的 IOC（依赖注入）方式？

#### 构造器依赖注入：构造器依赖注入通过容器触发一个类的构造器来实现的，该类有一系列参数，每个参数代表一个对其他类的依赖。

* Setter 方法注入：Setter 方法注入是容器通过调用无参构造器或无参static 工厂 方法实例化 bean 之后，调用该 bean 的 setter 方法，即实现了基

于 setter 的依赖注入。

### 20、哪种依赖注入方式你建议使用，构造器注入，还是 Setter 方法注入？

你两种依赖方式都可以使用， 构造器注入和 Setter 方法注入。最好的解决方案是用构造器参数实现强制依赖， setter 方法实现可选依赖。

### Spring Beans

### 21.什么是 Spring beans?

Spring beans 是那些形成 Spring 应用的主干的 java 对象。它们被 Spring IOC 容器初始化， 装配， 和管理。这些 beans 通过容器中配置的元数据创建。比如， 以 XML 文件中 的形式定义。

Spring 框架定义的 beans 都是单件 beans。在 bean tag 中有个属性”

singleton”， 如果它被赋为 TRUE， bean 就是单件， 否则就是一个 prototype bean。默认是 TRUE， 所以所有在 Spring 框架中的 beans 缺省都是单件。

### 22、一个 Spring Bean 定义 包含什么？

一个 Spring Bean 的定义包含容器必知的所有配置元数据， 包括如何创建一个bean， 它的生命周期详情及它的依赖。

### 23、如何给 Spring 容器提供配置元数据?

这里有三种重要的方法给 Spring 容器提供配置元数据。XML 配置文件。

基于注解的配置。基于 java 的配置。

### 24、你怎样定义类的作用域?

当定义一个 在 Spring 里， 我们还能给这个 bean 声明一个作用域。它可以通过bean 定义中的 scope 属性来定义。如，当 Spring 要在需要的时候每次生产一个新的 bean 实例，bean 的 scope 属性被指定为 prototype。另一方面，一个 bean

每次使用的时候必须返回同一个实例， 这个 bean 的 scope 属性 必须设为singleton。

### 25、解释 Spring 支持的几种bean 的作用域。

Spring 框架支持以下五种 bean 的作用域：

#### singleton : bean 在每个 Spring ioc 容器中只有一个实例。

* prototype：一个 bean 的定义可以有多个实例。
* request：每次 http 请求都会创建一个 bean，该作用域仅在基于 web 的 Spring ApplicationContext 情形下有效。
* session：在一个 HTTP Session 中，一个 bean 定义对应一个实例。该

作用域仅在基于 web 的 Spring ApplicationContext 情形下有效。

* global-session：在一个全局的 HTTP Session 中，一个 bean 定义对应一个实例。该作用域仅在基于 web 的 Spring ApplicationContext 情形下有效。

缺省的 Spring bean 的作用域是 Singleton.

### 26、Spring 框架中的单例 bean 是线程安全的吗?

不， Spring 框架中的单例 bean 不是线程安全的。

### 27、解释 Spring 框架中bean 的生命周期。

#### Spring 容器 从 XML 文件中读取 bean 的定义，并实例化 bean。

* Spring 根据 bean 的定义填充所有的属性。
* 如果bean 实现了BeanNameAware 接口，Spring 传递bean 的ID 到 setBeanName 方法。
* 如果 Bean 实现了 BeanFactoryAware 接口， Spring 传递beanfactory 给 setBeanFactory 方 法 。
* 如果有任何与 bean 相关联的 BeanPostProcessors，Spring 会在postProcesserBeforeInitialization()方法内调用它们。
* 如果 bean 实现 IntializingBean 了，调用它的 afterPropertySet 方法， 如果 bean 声明了初始化方法，调用此初始化方法。
* 如果有 BeanPostProcessors 和 bean 关联，这些 bean 的postProcessAfterInitialization() 方法将被调用。
* 如果 bean 实现了 DisposableBean，它将调用 destroy()方法。

### 28、哪些是重要的 bean 生命周期方法？你能重载它们吗？

有两个重要的 bean 生命周期方法， 第一个是 setup ， 它是在容器加载 bean 的时候被调用。第二个方法是 teardown 它是在容器卸载类的时候被调用。

The bean 标签有两个重要的属性（ init-method 和 destroy-method）。用它们你可以自己定制初始化和注销方法。它们也有相应的注解（ @PostConstruct 和@PreDestroy）。

### 29、什么是 Spring 的内部bean？

当一个 bean 仅被用作另一个 bean 的属性时， 它能被声明为一个内部 bean， 为了定义 inner bean， 在 Spring 的 基于 XML 的 配置元数据中， 可以在 或 元素内使用 元素， 内部 bean 通常是匿名的， 它们的 Scope 一般是 prototype。

### 30、在 Spring 中如何注入一个java 集合？

Spring 提供以下几种集合的配置元素：

#### 类型用于注入一列值，允许有相同的值。

* 类型用于注入一组值，不允许有相同的值。
* 类型用于注入一组键值对，键和值都可以为任意类型。
* 类型用于注入一组键值对，键和值都只能为 String 类型。

### 31、什么是 bean 装配?

装配，或 bean 装配是指在 Spring 容器中把 bean 组装到一起，前提是容器需要知道 bean 的依赖关系， 如何通过依赖注入来把它们装配到一起。

### 32、什么是 bean 的自动装配？

Spring 容器能够自动装配相互合作的 bean， 这意味着容器不需要和配置， 能通过 Bean 工厂自动处理 bean 之间的协作。

### 33、解释不同方式的自动装配 。

有五种自动装配的方式，可以用来指导 Spring 容器用自动装配方式来进行依赖注入。

#### no：默认的方式是不进行自动装配，通过显式设置 ref 属性来进行装配。

* byName：通过参数名 自动装配，Spring 容器在配置文件中发现 bean 的 autowire 属性被设置成 byname，之后容器试图匹配、装配和该 bean 的属性具有相同名字的 bean。
* byType:：通过参数类型自动装配，Spring 容器在配置文件中发现 bean

的 autowire 属性被设置成 byType，之后容器试图匹配、装配和该 bean 的属性具有相同类型的 bean。如果有多个 bean 符合条件，则抛出错误。

* constructor：这个方式类似于 byType， 但是要提供给构造器参数，如

果没有确定的带参数的构造器参数类型，将会抛出异常。

* autodetect：首先尝试使用 constructor 来自动装配，如果无法工作， 则使用 byType 方式。

### 34.自动装配有哪些局限性 ?

自动装配的局限性是：

#### 重写：你仍需用 和 配置来定义依赖，意味着总要重写自动装配。

* 基本数据类型：你不能自动装配简单的属性，如基本数据类型，String 字符串，和类。
* 模糊特性：自动装配不如显式装配精确，如果有可能，建议使用显式装配。

### 35、你可以在 Spring 中注入一个null 和一个空字符串吗？

可以。

### Spring 注解

### 36、什么是基于 Java 的Spring 注解配置? 给一些注解的例子.

基于 Java 的配置， 允许你在少量的 Java 注解的帮助下， 进行你的大部分 Spring 配置而非通过 XML 文件。

以[@Configuration](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D) 注解为例， 它用来标记类可以当做一个 bean 的定义， 被Spring IOC 容器使用。另一个例子是@Bean 注解， 它表示此方法将要返回一个对象， 作为一个 bean 注册进 Spring 应用上下文。

### 37、什么是基于注解的容器配置?

相对于 XML 文件，注解型的配置依赖于通过字节码元数据装配组件，而非尖括号的声明。

开发者通过在相应的类， 方法或属性上使用注解的方式， 直接组件类中进行配置， 而不是使用 xml 表述 bean 的装配关系。

### 38、怎样开启注解装配？

注解装配在默认情况下是不开启的，为了使用注解装配，我们必须在 Spring 配置文件中配置 [context:annotation-config/](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D)元素。

### 39、[@Required](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D) 注解

这个注解表明 bean 的属性必须在配置的时候设置，通过一个 bean 定义的显式的属性值或通过自动装配， 若@Required 注解的 bean 属性未被设置， 容器将抛出BeanInitializationException。

### 40、[@Autowired](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D) 注解

[@Autowired](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D) 注解提供了更细粒度的控制， 包括在何处以及如何完成自动装配。它的用法和@Required 一样，修饰 setter 方法、构造器、属性或者具有任意名称和/或多个参数的 PN 方法。

### 41、[@Qualifier](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D) 注解

当有多个相同类型的 bean 却只有一个需要自动装配时， 将[@Qualifier](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D) 注解和[@Autowire](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D) 注解结合使用以消除这种混淆， 指定需要装配的确切的 bean。

### Spring 数据访问

### 42.在 Spring 框架中如何更有效地使用JDBC?

使用 SpringJDBC 框架， 资源管理和错误处理的代价都会被减轻。所以开发者只需写 statements 和 queries 从数据存取数据，JDBC 也可以在 Spring 框架提供的模板类的帮助下更有效地被使用， 这个模板叫 JdbcTemplate （ 例子见这里here）

### 43、JdbcTemplate

JdbcTemplate 类提供了很多便利的方法解决诸如把数据库数据转变成基本数据类型或对象， 执行写好的或可调用的数据库操作语句， 提供自定义的数据错误处理。

### 44、Spring 对 DAO 的支持

Spring 对数据访问对象（ DAO） 的支持旨在简化它和数据访问技术如 JDBC， Hibernate or JDO 结合使用。这使我们可以方便切换持久层。编码时也不用担心会捕获每种技术特有的异常。

### 45、使用 Spring 通过什么方式访问Hibernate?

在 Spring 中有两种方式访问 Hibernate：

#### 控制反转 Hibernate Template 和 Callback。

* 继承 HibernateDAOSupport 提供一个 AOP 拦截器。

### 46、Spring 支持的 ORM

Spring 支持以下 ORM：

#### Hibernate

* iBatis
* JPA (Java Persistence API)
* TopLink
* JDO (Java Data Objects)
* OJB

47.如何通过HibernateDaoSupport 将Spring 和Hibernate

结合起来？

用 Spring 的 SessionFactory 调用 LocalSessionFactory。集成过程分三步：

#### 配置 the Hibernate SessionFactory。

* 继承 HibernateDaoSupport 实现一个 DAO。
* 在 AOP 支持的事务中装配。

### 48、Spring 支持的事务管理类型

Spring 支持两种类型的事务管理：

#### 编程式事务管理：这意味你通过编程的方式管理事务，给你带来极大的灵活性，但是难维护。

* 声明式事务管理：这意味着你可以将业务代码和事务管理分离，你只需用注解和 XML 配置来管理事务。

### 49、Spring 框架的事务管理有哪些优点？

* 它为不同的事务 API 如 JTA，JDBC，Hibernate，JPA 和 JDO，提供一个不变的编程模式。
* 它为编程式事务管理提供了一套简单的 API 而不是一些复杂的事务 API 如
* 它支持声明式事务管理。
* 它和 Spring 各种数据访问抽象层很好得集成。

### 50、你更倾向用那种事务管理类型？

大多数 Spring 框架的用户选择声明式事务管理， 因为它对应用代码的影响最小， 因此更符合一个无侵入的轻量级容器的思想。声明式事务管理要优于编程式事务管理， 虽然比编程式事务管理（ 这种方式允许你通过代码控制事务） 少了一点灵活性。

### Spring 面向切面编程（AOP） 51、解释 AOP

面向切面的编程， 或 AOP， 是一种编程技术， 允许程序模块化横向切割关注点，

或横切典型的责任划分， 如日志和事务管理。

### 52、Aspect 切面

AOP 核心就是切面， 它将多个类的通用行为封装成可重用的模块， 该模块含有一组 API 提供横切功能。比如，一个日志模块可以被称作日志的 AOP 切面。根据需求的不同， 一个应用程序可以有若干切面。在 Spring AOP 中， 切面通过带有@Aspect 注解的类实现。

### 52、在 Spring AOP 中，关注点和横切关注的区别是什么？

关注点是应用中一个模块的行为， 一个关注点可能会被定义成一个我们想实现的一个功能。

横切关注点是一个关注点， 此关注点是整个应用都会使用的功能， 并影响整个应用， 比如日志， 安全和数据传输， 几乎应用的每个模块都需要的功能。因此这些都属于横切关注点。

### 54、连接点

连接点代表一个应用程序的某个位置， 在这个位置我们可以插入一个 AOP 切面， 它实际上是个应用程序执行 Spring AOP 的位置。

### 55、通知

通知是个在方法执行前或执行后要做的动作， 实际上是程序执行时要通过SpringAOP 框架触发的代码段。

Spring 切面可以应用五种类型的通知：

#### before：前置通知，在一个方法执行前被调用。

* after: 在方法执行之后调用的通知，无论方法执行是否成功。
* after-returning: 仅当方法成功完成后执行的通知。
* after-throwing: 在方法抛出异常退出时执行的通知。
* around: 在方法执行之前和之后调用的通知。

### 56、切点

切入点是一个或一组连接点， 通知将在这些位置执行。可以通过表达式或匹配的方式指明切入点。

### 57、什么是引入?

引入允许我们在已存在的类中增加新的方法和属性。

### 58、什么是目标对象?

被一个或者多个切面所通知的对象。它通常是一个代理对象。也指被通知

（ advised） 对象。

### 59、什么是代理?

代理是通知目标对象后创建的对象。从客户端的角度看， 代理对象和目标对象是一样的。

### 60、有几种不同类型的自动代理？

BeanNameAutoProxyCreator

DefaultAdvisorAutoProxyCreator Metadata autoproxying

### 61、什么是织入。什么是织入应用的不同点？

织入是将切面和到其他应用类型或对象连接或创建一个被通知对象的过程。 织入可以在编译时， 加载时， 或运行时完成。

### 62、解释基于 XML Schema 方式的切面实现。

在这种情况下， 切面由常规类以及基于 XML 的配置实现。

### 63、解释基于注解的切面实现

在这种情况下(基于@AspectJ 的实现)，涉及到的切面声明的风格与带有 java5 标注的普通 java 类一致。

### Spring 的MVC

### 64、什么是 Spring 的MVC 框架？

Spring 配备构建 Web 应用的全功能 MVC 框架。Spring 可以很便捷地和其他MVC 框架集成，如 Struts，Spring 的 MVC 框架用控制反转把业务对象和控制逻辑清晰地隔离。它也允许以声明的方式把请求参数和业务对象绑定。

### 65、DispatcherServlet

Spring 的 MVC 框架是围绕 DispatcherServlet 来设计的，它用来处理所有的 HTTP 请求和响应。

### 66、WebApplicationContext

WebApplicationContext 继承了 ApplicationContext 并增加了一些 WEB 应用必备的特有功能， 它不同于一般的 ApplicationContext ， 因为它能处理主题， 并找到被关联的 servlet。

### 67、什么是 Spring MVC 框架的控制器？

控制器提供一个访问应用程序的行为， 此行为通常通过服务接口实现。控制器解析用户输入并将其转换为一个由视图呈现给用户的模型。Spring 用一个非常抽象的方式实现了一个控制层， 允许用户创建多种用途的控制器。

### 68、[@Controller](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D) 注解

该注解表明该类扮演控制器的角色， Spring 不需要你继承任何其他控制器基类或引用 Servlet API。

### 69、[@RequestMapping](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D) 注解

该注解是用来映射一个 URL 到一个类或一个特定的方处理法上。