**Struts部分**

1. 说说你对Struts2的理解及Struts2 的运行原理

Struts2是一款优秀的MVC框架，框架本身大致可以分为3个部分：核心控制器FilterDispatcher、业务控制器Action和用户实现的企业业务逻辑组件。核心控制器FilterDispatcher是Struts 2框架的基础，包含了框架内部的控制流程和处理机制。业务控制器Action和业务逻辑组件是需要用户来自己实现的。用户在开发Action和业务逻辑组件的同时，还需要编写相关的配置文件，供核心控制器FilterDispatcher来使用。

Struts2的执行流程：

1 、客户端初始化一个指向Servlet容器的请求；

2、 这个请求经过一系列的过滤器（Filter）（这些过滤器中有一个叫做ActionContextCleanUp的可选过滤器，这个过滤器对于Struts2和其他框架的集成很有帮助，例如：SiteMesh Plugin）

3 、接着FilterDispatcher被调用，FilterDispatcher询问ActionMapper来决定这个请是否需要调用某个Action

4、如果ActionMapper决定需要调用某个Action，FilterDispatcher把请求的处理交给ActionProxy

5、ActionProxy通过Configuration Manager询问框架的配置文件，找到需要调用的Action类

6、ActionProxy创建一个ActionInvocation的实例。

7、ActionInvocation实例使用命名模式来调用， 在调用Action的过程前后，涉及到相关拦截器（Intercepter）的调用。

8、一旦Action执行完毕，ActionInvocation负责根据struts.xml中的配置找到对应的返回结果

1. Struts2 如何定位action中的方法

1 感叹号定位方法（动态方法）。  
2 在xml配置文件中通过配置多个action，使用action的method指定方法。  
3 使用通配符(\*)匹配方法。

1. 模型驱动与属性驱动是什么 模型驱动使用时注意什么问题

答：模型驱动与属性驱动都是用来封装数据的。  
1.模型驱动：在实现类中实现ModelDriven<T>接口使用泛型把属性类封装起来，重写getModel()方法，然后在实现类里创建一个属性类的实例，  
通过这个实例拿到封装进来的值，拿返回值的时候使用工具进行值拷贝。  
2.属性驱动：在实现类里定义属性，生成get与set方法，通过属性来拿值。  
注意：模型驱动使用时注意的是在使用前先把属性类实例化，否则会出现空指针错误，拿返回对象的值需要用拷贝内存因为地址发生改变。  
模型驱动不可以使用局部类型转换器。

4、Strust2如何访问Servlet API

答：1：通过ActionContext可以获得request，application，session三个作用域(struts2实际上是通过适配器来把servlet的api转换为一个map，  
并把这些map放在ActionContext里面)。  
2：通过ActionContext的子类ServletActionContext去获取原滋原味的API。  
3：可以通过实现ServletRequestAware接口，重写里面的setServletRequest方法可以获得request，实际上是通过set的依赖注入。

5、什么是拦截器 说说Struts2用拦截器来干什么 说出6个拦截器来

答：在访问类的某个方法或者属性之前执行，拦截的是Action的请求，进行拦截然后在方法的执行前或者之后加入某些操作，如：国际化，权限，效验等与源代码无关的操作。国际化，表单重复提交，类型转换，自动装配，数据封装,异常处理，模型驱动，请求参数，处理类型转换错误，日志拦截器。

6、如何实现自定义拦截器？需要注意什么？

答：可以继承MethodFilterInterceptor重写doIntercept方法指定某个方法进行拦截，或者继承AbstractInterceptor，重写intercept方法,在xml配置文件中定义自定义拦截器，然后注入到拦截器栈中，再把拦截器引用到action中。在把拦截器注入拦截器栈时配置<param>标签，使用name属性配置excludeMethods表示不拦截指定方法。在配置文件中配置includeMethods指定拦截某个方法。可配置到某个action单独使用，也可以配置到包下面的所有action使用。

注意要把默认的拦截器栈重新引用，否则会被覆盖。

7、ActionContext是用来干什么的

答：ActionContext是Action的上下文。通过ActionContext这个类来可以获得:request，application，session，Locale、ValueStack,ConversionErrors  
1 可以获得servlet的api：request，application，session。  
2 进行国际化：获得语言环境 Locale。  
3 收集错误信息：ConversionErrors。  
4 有个值栈来装action实例与创建的模型驱动的实例:ValueStack。  
5 ActionContext是线程安全的-用本地线程实现的。

8、拦截器和过滤器的区别

1、拦截器是基于java反射机制的，而过滤器是基于函数回调的。  
2、过滤器依赖于servlet容器，而拦截器不依赖于servlet容器。  
3、拦截器只能对Action请求起作用，而过滤器则可以对几乎所有请求起作用。  
4、拦截器可以访问Action上下文、值栈里的对象，而过滤器不能。  
5、在Action的生命周期中，拦截器可以多次调用，而过滤器只能在容器初始化时被调用一次。

9、struts2是如何启动的？

struts2框架是通过Filter启动的，即StrutsPrepareAndExecuteFilter，此过滤器为struts2的核心过滤器；StrutsPrepareAndExecuteFilter的init()方法中将会读取类路径下默认的配置文件struts.xml完成初始化操作。struts2读取到struts.xml的内容后，是将内容封装进javabean对象然后存放在内存中，以后用户的每次请求处理将使用内存中的数据，而不是每次请求都读取struts.xml文件。

10、ActionContext、ServletContext、pageContext的区别？

1）ActionContext是当前的Action的上下文环境，通过ActionContext可以获取到request、session、ServletContext等与Action有关的对象的引用；

2）ServletContext是域对象，一个web应用中只有一个ServletContext，生命周期伴随整个web应用；

3）pageContext是JSP中的最重要的一个内置对象，可以通过pageContext获取其他域对象的应用，同时它是一个域对象，作用范围只针对当前页面，当前页面结束时，pageContext销毁，  生命周期是JSP四个域对象中最小的。

11、拦截器的生命周期与工作过程？

1）每个拦截器都是实现了Interceptor接口的 Java 类；

2）init(): 该方法将在拦截器被创建后立即被调用, 它在拦截器的生命周期内只被调用一次. 可以在该方法中对相关资源进行必要的初始化；

3）intercept(ActionInvocation invocation): 每拦截一个动作请求, 该方法就会被调用一次；

4）destroy: 该方法将在拦截器被销毁之前被调用, 它在拦截器的生命周期内也只被调用一次；

5）struts2中有内置了18个拦截器。

12、struts的工作原理

1、初始化，读取struts-config.xml、web.xml等配置文件（所有配置文件的初始化）

2、发送HTTP请求,客户端发送以.do结尾的请求

3、填充FormBean（实例化、复位、填充数据、校验、保存）

4、将请求转发到Action（调用Action的execute（）方法）

5、处理业务（可以调用后台类，返回ActionForward对象）

6、返回目标响应对象（从Action返回到ActionServlet）

7、转换Http请求到目标响应对象（查找响应，根据返回的Forward keyword）

8、Http响应，返回到Jsp页面

**Hibernate部分**

1、Hibernate工作原理？

1.读取并解析配置文件

2.读取并解析映射信息，创建SessionFactory

3.打开Session

4.创建事务Transation

5.持久化操作

6.提交事务

7.关闭Session

8.关闭SesstionFactory

2、为什么要使用Hibernate？

1. 对JDBC访问数据库的代码做了封装，大大简化了数据访问层繁琐的重复性代码。

2. Hibernate是一个基于JDBC的主流持久化框架，是一个优秀的ORM实现。他很大程度的简化DAO层的编码工作

3. hibernate使用Java反射机制，而不是字节码增强程序来实现透明性。

4. hibernate的性能非常好，因为它是个轻量级框架。映射的灵活性很出色。它支持各种关系数据库，从一对一到多对多的各种复杂关系。

3、Hibernate是如何延迟加载?

Hibernate3提供了属性的延迟加载功能 当Hibernate在查询数据的时候，数据并没有存在与内存中，当程序真正对数据的操作时，对象才存在与内存中，就实现了延迟加载，他节省了服务器的内存开销，从而提高了服务器的性能。

 4、Hibernate中怎样实现类之间的关系?(如：一对多、多对多的关系)

类与类之间的关系主要体现在表与表之间的关系进行操作，它们都是对对象进行操作，我们程序中把所有的表与类都映射在一起，它们通过配置文件中的many-to-one、one-to-many、many-to-many

5、load()和get()的区别

hibernate对于load方法认为该数据在数据库中一定存在，可以放心的使用代理来延迟加载，load默认支持延迟加载，在用到对象中的其他属性数 据时才查询数据库，但是万一数据库中不存在该记录，只能抛异常ObjectNotFoundEcception；所说的load方法抛异常是指在使用该对 象的数据时，数据库中不存在该数据时抛异常，而不是在创建这个对象时。由于session中的缓存对于hibernate来说是个相当廉价的资源，所以在 load时会先查一下session缓存看看该id对应的对象是否存在，不存在则创建代理（load时候之查询一级缓存，不存在则创建代理）。get() 现在一级缓存找，没有就去二级缓存找，没有就去数据库找，没有就返回null ；而对于get方法，hibernate一定要获取到真实的数据，否则返回null。

7、谈谈hibernate的延迟加载和openSessionInView

延迟加载要在session范围内，用到的时候再加载；opensessioninview是在web层写了一个filter来打开和关闭session，这样就表示在一次request过程中session一直开着，保证了延迟加载在session中的这个前提。

8.Hibernate中的update()和saveOrUpdate()的区别.

saveOrUpdate()做下面的事:

如果对象已经在本session中持久化了，不做任何事

如果另一个与本session关联的对象拥有相同的持久化标识(identifier)，抛出一个异常

如果对象没有持久化标识(identifier)属性，对其调用save()

如果对象的持久标识(identifier)表明其是一个新实例化的对象，对其调用save()

如果对象是附带版本信息的（通过 <version>或 <timestamp>） 并且版本属性的值表明其是一个新实例化的对象，save()它。 否则update() 这个对象

9、说一下Hibernate对象的状态

**Spring部分**

1、spring工作原理：

1.spring mvc请所有的请求都提交给DispatcherServlet,它会委托应用系统的其他模块负责负责对请求进行真正的处理工作。  
2.DispatcherServlet查询一个或多个HandlerMapping,找到处理请求的Controller.   
3.DispatcherServlet请请求提交到目标Controller   
4.Controller进行业务逻辑处理后，会返回一个ModelAndView   
5.Dispathcher查询一个或多个ViewResolver视图解析器,找到ModelAndView对象指定的视图对象   
6.视图对象负责渲染返回给客户端。

2、为什么要用spring:

AOP 让开发人员可以创建非行为性的关注点，称为横切关注点，并将它们插入到应用程序代码中。使用 AOP 后，公共服务 （比 如日志、持久性、事务等）就可以分解成方面并应用到域对象上，同时不会增加域对象的对象模型的复杂性。   
  
IOC 允许创建一个可以构造对象的应用环境，然后向这些对象传递它们的协作对象。正如单词 倒置 所表明的，IOC 就像反 过来的 JNDI。没有使用一堆抽象工厂、服务定位器、单元素（singleton）和直接构造（straight construction），每一个对象都是用 其协作对象构造的。因此是由容器管理协作对象（collaborator）。   
  
Spring即使一个AOP框架，也是一IOC容器。 Spring 最好的地方是它有助于您替换对象。有了 Spring，只要用 JavaBean 属性和配置文件加入依赖性（协作对象）。然后可以很容易地在需要时替换具有类似接口的协作对象。

3、请你谈谈SSH整合：

SSH：  
Struts（表示层）+Spring（业务层）+Hibernate（持久层）  
Struts：  
Struts是一个表示层框架，主要作用是界面展示，接收请求，分发请求。  
在MVC框架中，Struts属于VC层次，负责界面表现，负责MVC关系的分发。（View：沿用JSP，HTTP，Form，Tag，Resourse ；Controller：ActionServlet，struts-config.xml，Action）  
Hibernate：  
Hibernate是一个持久层框架，它只负责与关系数据库的操作。  
Spring：  
Spring是一个业务层框架，是一个整合的框架，能够很好地黏合表示层与持久层。

4、Spring里面如何配置数据库驱动？

使用”org.springframework.jdbc.datasource.DriverManagerDataSource”数据源来配置数据库驱动。示例如下：  
<bean id=”dataSource”>  
<property name=”driverClassName”>  
<value>org.hsqldb.jdbcDriver</value>  
</property>  
<property name=”url”>  
<value>jdbc:hsqldb:db/appfuse</value>  
</property>  
<property name=”username”><value>sa</value></property>  
<property name=”password”><value></value></property>  
</bean>

5、Spring里面applicationContext.xml文件能不能改成其他文件名？

ContextLoaderListener是一个ServletContextListener, 它在你的web应用启动的时候初始化。缺省情况下， 它会在WEB-INF/applicationContext.xml文件找Spring的配置。 你可以通过定义一个<context-param>元素名字为”contextConfigLocation”来改变Spring配置文件的位置。示例如下：

<listener>  
<listener-class>org.springframework.web.context.ContextLoaderListener

<context-param>  
<param-name>contextConfigLocation</param-name>  
<param-value>/WEB-INF/xyz.xml</param-value>  
</context-param>

</listener-class>  
</listener>

6、如何在web应用里面配置spring?

在web.xml中加入如下同容,在启动web服务器时加载/WEB-INF/applicationContext.xml中的内容。  
<servlet>  
<servlet-name>context</servlet-name>  
<servlet-class>  
org.springframework.web.context.ContextLoaderServlet  
</servlet-class>  
<load-on-startup>1</load-on-startup>  
</servlet>  
通过如下类得到ApplicationContext实例  
WebApplicationContextUtils.getWebApplicationContext

7、Spring里面如何定义hibernate mapping？

添加hibernate mapping 文件到web/WEB-INF目录下的applicationContext.xml文件里面。示例如下：  
<property name=”mappingResources”>  
<list>  
<value>org/appfuse/model/User.hbm.xml</value>  
</list>  
</property>

8、解释一下Dependency injection(DI,依赖注入)和IOC(Inversion of control,控制反转)?

依赖注入DI是一个程序设计模式和架构模型， 一些时候也称作控制反转，尽管在技术上来讲，依赖注入是一个IOC的特殊实现，依赖注入是指一个对象应用另外一个对象来提供一个特殊的能力，例如：把一个数据库连接已参数的形式传到一个对象的结构方法里面而不是在那个对象内部自行创建一个连接。控制反转和依赖注入的基本思想就是把类的依赖从类内部转化到外部以减少依赖  
应用控制反转，对象在被创建的时候，由一个调控系统内所有对象的外界实体，将其所依赖的对象的引用，传递给它。也可以说，依赖被注入到对象中。所以，控制反转是，关于一个对象如何获取他所依赖的对象的引用，这个责任的反转。

9、spring中的BeanFactory与ApplicationContext的作用有哪些？

1. BeanFactory负责读取bean配置文档，管理bean的加载，实例化，维护bean之间的依赖关系，负责bean的声明周期。  
2. ApplicationContext除了提供上述BeanFactory所能提供的功能之外，还提供了更完整的框架功能：

a. 国际化支持  
b. 资源访问：Resource rs = ctx. getResource(”classpath:config.properties”), “file:c:/config.properties”  
c. 事件传递：通过实现ApplicationContextAware接口  
d. 常用的获取ApplicationContext的方法：  
FileSystemXmlApplicationContext：从文件系统或者url指定的xml配置文件创建，参数为配置文件名或文件名数组  
ClassPathXmlApplicationContext：从classpath的xml配置文件创建，可以从jar包中读取配置文件  
WebApplicationContextUtils：从web应用的根目录读取配置文件，需要先在web.xml中配置，可以配置监听器或者servlet来实现  
<listener>  
<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  
</listener>  
<servlet>  
<servlet-name>context</servlet-name>  
<servlet-class>org.springframework.web.context.ContextLoaderServlet</servlet-class>  
<load-on-startup>1</load-on-startup>  
</servlet>  
这两种方式都默认配置文件为web-inf/applicationContext.xml，也可使用context-param指定配置文件  
<context-param>  
<param-name>contextConfigLocation</param-name>  
<param-value>/WEB-INF/myApplicationContext.xml</param-value>  
</context-param>

10、如何配置spring+struts?

在struts-config.xml加入一个插件，通过它加载applicationContext.xml在struts-config.xml修改action-mapping标记，具体action交给了DelegateActionProxy 通过DelegateActionProxy进入一spring的环境。 在spring的配置文件加入<bean name=”/login” class=”" singleton=”false”/>

11、spring中的核心类有那些，各有什么作用?

BeanFactory：产生一个新的实例，可以实现单例模式  
BeanWrapper：提供统一的get及set方法  
ApplicationContext:提供框架的实现，包括BeanFactory的所有功能

12、什么是aop，aop的作用是什么?

面向切面编程（AOP）提供另外一种角度来思考程序结构，通过这种方式弥补了面向对象编程（OOP）的不足除了类（classes）以外，AOP提供了切面。切面对关注点进行模块化，例如横切多个类型和对象的事务管理  
Spring的一个关键的组件就是AOP框架，可以自由选择是否使用AOP  
提供声明式企业服务，特别是为了替代EJB声明式服务。最重要的服务是声明性事务管理，这个服务建立在Spring的抽象事物管理之上  
允许用户实现自定义切面，用AOP来完善OOP的使用  
可以把Spring AOP看作是对Spring的一种增强

13、请介绍一下Spring框架中Bean的生命周期

一、Bean的定义  
Spring通常通过配置文件定义Bean。如：

<?xml version=”1.0″ encoding=”UTF-8″?>  
<beans xmlns=”http://www.springframework.org/schema/beans”  
xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”  
xsi:schemaLocation=”http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.0.xsd”>  
<bean id=”HelloWorld” class=”com.pqf.beans.HelloWorld”>  
<property name=”msg”>  
<value>HelloWorld</value>  
</property>  
</bean>  
</beans>

这个配置文件就定义了一个标识为 HelloWorld 的Bean。在一个配置文档中可以定义多个Bean。

二、Bean的初始化  
有两种方式初始化Bean。  
1、在配置文档中通过指定init-method 属性来完成  
在Bean的类中实现一个初始化Bean属性的方法，如init()，如：  
public class HelloWorld{  
public String msg=null;  
public Date date=null;

public void init() {  
msg=”HelloWorld”;  
date=new Date();  
}  
……  
}  
然后，在配置文件中设置init-mothod属性：  
<bean id=”HelloWorld” class=”com.pqf.beans.HelloWorld” init-mothod=”init” >  
</bean>

2、实现 org.springframwork.beans.factory.InitializingBean接口  
Bean实现InitializingBean接口，并且增加 afterPropertiesSet() 方法：

public class HelloWorld implement InitializingBean {  
public String msg=null;  
public Date date=null;

public void afterPropertiesSet() {  
msg=”向全世界问好！”;  
date=new Date();  
}  
……  
}

那么，当这个Bean的所有属性被Spring的BeanFactory设置完后，会自动调用afterPropertiesSet()方法对Bean进行初始化，于是，配置文件就不用指定 init-method属性了。

三、Bean的调用  
有三种方式可以得到Bean并进行调用：  
1、使用BeanWrapper  
HelloWorld hw=new HelloWorld();  
BeanWrapper bw=new BeanWrapperImpl(hw);  
bw.setPropertyvalue(”msg”,”HelloWorld”);  
system.out.println(bw.getPropertyCalue(”msg”));

2、使用BeanFactory  
InputStream is=new FileInputStream(”config.xml”);  
XmlBeanFactory factory=new XmlBeanFactory(is);  
HelloWorld hw=(HelloWorld) factory.getBean(”HelloWorld”);  
system.out.println(hw.getMsg());

3、使用ApplicationConttext  
ApplicationContext actx=new FleSystemXmlApplicationContext(”config.xml”);  
HelloWorld hw=(HelloWorld) actx.getBean(”HelloWorld”);  
System.out.println(hw.getMsg());

四、Bean的销毁  
1、使用配置文件中的 destory-method 属性  
与初始化属性 init-methods类似，在Bean的类中实现一个撤销Bean的方法，然后在配置文件中通过 destory-method指定，那么当bean销毁时，Spring将自动调用指定的销毁方法。

2、实现 org.springframwork.bean.factory.DisposebleBean接口  
如果实现了DisposebleBean接口，那么Spring将自动调用bean中的Destory方法进行销毁，所以，Bean中必须提供Destory方法。

14、AOP里面重要的几个名词概念解释：

* 切面（Aspect）： 一个关注点的模块化，这个关注点可能会横切多个对象。事务管理是J2EE应用中一个关于横切关注点的很好的例子。 在Spring AOP中，切面可以使用通用类（基于模式的风格） 或者在普通类中以 @Aspect 注解（@AspectJ风格）来实现。
* 连接点（Joinpoint）： 在程序执行过程中某个特定的点，比如某方法调用的时候或者处理异常的时候。 在Spring AOP中，一个连接点 总是 代表一个方法的执行。 通过声明一个org.aspectj.lang.JoinPoint类型的参数可以使通知（Advice）的主体部分获得连接点信息。
* 通知（Advice）： 在切面的某个特定的连接点（Joinpoint）上执行的动作。通知有各种类型，其中包括“around”、“before”和“after”等通知。 通知的类型将在后面部分进行讨论。许多AOP框架，包括Spring，都是以拦截器做通知模型， 并维护一个以连接点为中心的拦截器链。
* 切入点（Pointcut）： 匹配连接点（Joinpoint）的断言。通知和一个切入点表达式关联，并在满足这个切入点的连接点上运行（例如，当执行某个特定名称的方法时）。 切入点表达式如何和连接点匹配是AOP的核心：Spring缺省使用AspectJ切入点语法。

15、说说AOP里面通知的几种类型：

* 前置通知（Before advice）： 在某连接点（join point）之前执行的通知，但这个通知不能阻止连接点前的执行（除非它抛出一个异常）。
* 返回后通知（After returning advice）： 在某连接点（join point）正常完成后执行的通知：例如，一个方法没有抛出任何异常，正常返回。
* 抛出异常后通知（After throwing advice）： 在方法抛出异常退出时执行的通知。
* 后通知（After (finally) advice）： 当某连接点退出的时候执行的通知（不论是正常返回还是异常退出）。
* 环绕通知（Around Advice）： 包围一个连接点（join point）的通知，如方法调用。这是最强大的一种通知类型。 环绕通知可以在方法调用前后完成自定义的行为。它也会选择是否继续执行连接点或直接返回它们自己的返回值或抛出异常来结束执行。

**MyBatis部分**

1、#{}和${}的区别是什么？

答：${}是Properties文件中的变量占位符，它可以用于标签属性值和sql内部，属于静态文本替换，比如${driver}会被静态替换为com.mysql.jdbc.Driver。#{}是sql的参数占位符，Mybatis会将sql中的#{}替换为?号，在sql执行前会使用PreparedStatement的参数设置方法，按序给sql的?号占位符设置参数值，比如ps.setInt(0, parameterValue)，#{item.name}的取值方式为使用反射从参数对象中获取item对象的name属性值，相当于param.getItem().getName()。

2、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：还有很多其他的标签，<resultMap>、<parameterMap>、<sql>、<include>、<selectKey>，加上动态sql的9个标签，trim|where|set|foreach|if|choose|when|otherwise|bind等，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。

3、最佳实践中，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？

答：Dao接口，就是人们常说的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MappedStatement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MappedStatement，举例：com.mybatis3.mappers.StudentDao.findStudentById，可以唯一找到namespace为com.mybatis3.mappers.StudentDao下面id = findStudentById的MappedStatement。在Mybatis中，每一个<select>、<insert>、<update>、<delete>标签，都会被解析为一个MappedStatement对象。

Dao接口里的方法，是不能重载的，因为是全限名+方法名的保存和寻找策略。

Dao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Dao接口生成代理proxy对象，代理对象proxy会拦截接口方法，转而执行MappedStatement所代表的sql，然后将sql执行结果返回。

4、Mybatis是如何进行分页的？分页插件的原理是什么？

答：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。

举例：select \* from student，拦截sql后重写为：select t.\* from （select \* from student）t limit 0，10

5、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。

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。

6、Mybatis执行批量插入，能返回数据库主键列表吗？

答：能，JDBC都能，Mybatis当然也能。

7、Mybatis动态sql是做什么的？都有哪些动态sql？能简述一下动态sql的执行原理不？

答：Mybatis动态sql可以让我们在Xml映射文件内，以标签的形式编写动态sql，完成逻辑判断和动态拼接sql的功能，Mybatis提供了9种动态sql标签trim|where|set|foreach|if|choose|when|otherwise|bind。

其执行原理为，使用OGNL从sql参数对象中计算表达式的值，根据表达式的值动态拼接sql，以此来完成动态sql的功能。

8、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？

答：第一种是使用<resultMap>标签，逐一定义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，将列别名书写为对象属性名，比如T\_NAME AS NAME，对象属性名一般是name，小写，但是列名不区分大小写，Mybatis会忽略列名大小写，智能找到与之对应对象属性名，你甚至可以写成T\_NAME AS NaMe，Mybatis一样可以正常工作。

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。

9、Mybatis能执行一对一、一对多的关联查询吗？都有哪些实现方式，以及它们之间的区别。

答：能，Mybatis不仅可以执行一对一、一对多的关联查询，还可以执行多对一，多对多的关联查询，多对一查询，其实就是一对一查询，只需要把selectOne()修改为selectList()即可；多对多查询，其实就是一对多查询，只需要把selectOne()修改为selectList()即可。

关联对象查询，有两种实现方式，一种是单独发送一个sql去查询关联对象，赋给主对象，然后返回主对象。另一种是使用嵌套查询，嵌套查询的含义为使用join查询，一部分列是A对象的属性值，另外一部分列是关联对象B的属性值，好处是只发一个sql查询，就可以把主对象和其关联对象查出来。

那么问题来了，join查询出来100条记录，如何确定主对象是5个，而不是100个？其去重复的原理是<resultMap>标签内的<id>子标签，指定了唯一确定一条记录的id列，Mybatis根据<id>列值来完成100条记录的去重复功能，<id>可以有多个，代表了联合主键的语意。

同样主对象的关联对象，也是根据这个原理去重复的，尽管一般情况下，只有主对象会有重复记录，关联对象一般不会重复。

举例：下面join查询出来6条记录，一、二列是Teacher对象列，第三列为Student对象列，Mybatis去重复处理后，结果为1个老师6个学生，而不是6个老师6个学生。

       t\_id    t\_name           s\_id

|          1 | teacher      |      38 |  
|          1 | teacher      |      39 |  
|          1 | teacher      |      40 |  
|          1 | teacher      |      41 |  
|          1 | teacher      |      42 |  
|          1 | teacher      |      43 |

10、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。

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

答：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；毕竟namespace不是必须的，只是最佳实践而已。

原因就是namespace+id是作为Map<String, MappedStatement>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。

12、Mybatis中如何执行批处理？

答：使用BatchExecutor完成批处理。

13、Mybatis都有哪些Executor执行器？它们之间的区别是什么？

答：Mybatis有三种基本的Executor执行器，**SimpleExecutor、ReuseExecutor、BatchExecutor。**

**SimpleExecutor：**每执行一次update或select，就开启一个Statement对象，用完立刻关闭Statement对象。

**ReuseExecutor：**执行update或select，以sql作为key查找Statement对象，存在就使用，不存在就创建，用完后，不关闭Statement对象，而是放置于Map<String, Statement>内，供下一次使用。简言之，就是重复使用Statement对象。

**BatchExecutor：**执行update（没有select，JDBC批处理不支持select），将所有sql都添加到批处理中（addBatch()），等待统一执行（executeBatch()），它缓存了多个Statement对象，每个Statement对象都是addBatch()完毕后，等待逐一执行executeBatch()批处理。与JDBC批处理相同。

作用范围：Executor的这些特点，都严格限制在SqlSession生命周期范围内。

14、Mybatis中如何指定使用哪一种Executor执行器？

答：在Mybatis配置文件中，可以指定默认的ExecutorType执行器类型，也可以手动给DefaultSqlSessionFactory的创建SqlSession的方法传递ExecutorType类型参数。

15、Mybatis是否可以映射Enum枚举类？

答：Mybatis可以映射枚举类，不单可以映射枚举类，Mybatis可以映射任何对象到表的一列上。映射方式为自定义一个TypeHandler，实现TypeHandler的setParameter()和getResult()接口方法。TypeHandler有两个作用，一是完成从javaType至jdbcType的转换，二是完成jdbcType至javaType的转换，体现为setParameter()和getResult()两个方法，分别代表设置sql问号占位符参数和获取列查询结果。

16、Mybatis映射文件中，如果A标签通过include引用了B标签的内容，请问，B标签能否定义在A标签的后面，还是说必须定义在A标签的前面？

答：虽然Mybatis解析Xml映射文件是按照顺序解析的，但是，被引用的B标签依然可以定义在任何地方，Mybatis都可以正确识别。

原理是，Mybatis解析A标签，发现A标签引用了B标签，但是B标签尚未解析到，尚不存在，此时，Mybatis会将A标签标记为未解析状态，然后继续解析余下的标签，包含B标签，待所有标签解析完毕，Mybatis会重新解析那些被标记为未解析的标签，此时再解析A标签时，B标签已经存在，A标签也就可以正常解析完成了。

17、简述Mybatis的Xml映射文件和Mybatis内部数据结构之间的映射关系？

答：Mybatis将所有Xml配置信息都封装到All-In-One重量级对象Configuration内部。在Xml映射文件中，<parameterMap>标签会被解析为ParameterMap对象，其每个子元素会被解析为ParameterMapping对象。<resultMap>标签会被解析为ResultMap对象，其每个子元素会被解析为ResultMapping对象。每一个<select>、<insert>、<update>、<delete>标签均会被解析为MappedStatement对象，标签内的sql会被解析为BoundSql对象。

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？

答：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、Mybatis基础: #{...} 和 ${...} 的区别

MyBatis将 #{…} 解释为JDBC prepared statement 的一个参数标记。而将 ${…} 解释为字符串替换。理解这两者的区别是很有用的, 因为在某些SQL语句中并不能使用参数标记(parameter markers)。

比如,我们不能在表名(table name)的位置使用参数标记。

假设有下面的代码:

Map<String, Object> parms = new HashMap<String, Object>();

parms.put("table", "foo"); // 表名

parms.put("criteria", 37); // 查询过滤条件

List<Object> rows = mapper.generalSelect(parms);

<select id="generalSelect" parameterType="map">

select \* from ${table} where col1 = #{criteria}

</select>

MyBatis生成的SQL语句(prepared statement)如下所示:

select \* from foo where col1 = ?

重要提示: 请注意,使用$ {…} (字符串替换)时可能会有SQL注入攻击的风险。另外,字符串替换在处理复杂类型也可能常常发生问题,如日期类型。由于这些因素,我们建议您尽可能地使用 #{…} 这种方式。要使用LIKE语句该怎么写?

20、有两种使用LIKE的方法。(推荐使用)第一种方法是,在[Java](http://www.2cto.com/kf/ware/Java/)代码中添加SQL通配符。

示例一:

String wildcardName = "%Smi%";

List<Name> names = mapper.selectLike(wildcardName);

<select id="selectLike">

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

</select>

第二种方式是在SQL语句中拼接通配符。这种方法相对来说安全性要低一些,因为可能会被SQL注入攻击。

示例二:

String wildcardName = "Smi";

List<Name> names = mapper.selectLike(wildcardName);

<select id="selectLike">

 select \* from foo where bar like '%' || '${value}' || '%'

</select>

重要提示: 请注意两种方式中 $ 和 # 的使用!

21、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<String>();

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();

} finally {

 sqlSession.close();

}

22、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。而自动生成的键值在 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());

23、在mapper中如何传递多个参数?

Java的反射机制并不能让框架获取到参数的名字(方法签名中只有参数类型,可以说是为了优化,也可以说设计就是如此,总之名字无意义), 所以MyBatis默认的命名为: param1,param2……

如果想给他们指定名称,可以使用 @param 注解:

import org.apache.ibatis.annotations.Param;

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>

**Spring MVC部分**

一、SpringMVC工作原理：

1.spring mvc请所有的请求都提交给DispatcherServlet,它会委托应用系统的其他模块负责负责对请求进行真正的处理工作。

2.DispatcherServlet查询一个或多个HandlerMapping,找到处理请求的Controller.  3.DispatcherServlet请请求提交到目标Controller

4.Controller进行业务逻辑处理后，会返回一个ModelAndView

5.Dispathcher查询一个或多个ViewResolver视图解析器,找到ModelAndView对象指定的视图对象

6.视图对象负责渲染返回给客户端。