# 一.能力

# 1.觉得自己有哪些优势 ？

首先作为应届生，我能全身心的投入到公司工作；

我觉得我在学习方面的优势就是：自学能力强，有毅力。我对一些比较新潮的技术我都非常愿意去学它，来提升自己并且为公司创造价值；

在工作方面的优势就是，做事积极性高，有上进心。能按时或者说提前完成上级所布置的任务，能积极地配合同事完成工作；

在生活方面，我为人随和，容易相处，适应能力强，能尽快的融入公司，薪资方面也没事什么要求，我能在这个城市生存下去吧，哈哈，谢谢！

**注意突出自己的差异点,与别人有什么不同！！！**

# **你最擅长什么?**

在技术方面我最擅长的就是举一反三。比如说碰到一个很平常的技术，我会想会不会还有更简单，更炫的技术来解决，哈哈，其实我是一个追求完美的人。

# 二.

# 1.项目中遇到了哪些困难，如何解决的？

做项目遇到困难当然是不可避免的，有些项目可能会遇到自己没有学到的知识，那这时候我就会花最短的时间去学会这些新技术，然后把它应用在项目中；有时候组员做项目进度慢，我也会进行协调；还有项目中遇到的报错就都是小意思了，自己用debug进行调试，实在是不会的我会问问来时同学来解决。谢谢。

# 2.项目中用到了哪些技术？

现在做的项目 我会用到三大框架SSH,SSM的技术，前端也会用的一些前端的框架 ，比如说EasyUI ，bootstrap;

# 3.对公司有什么要了解的？还有什么要问的？

我想问一下我在贵公司的实习期有多长，我多久可以转正？还有我实习期间我大概做些什么，有没有项目给我练习，如果我有不会的我可以问谁？

# 需求文档有哪些，敏捷开发的流程？



# 对mvc的了解？及执行过程。

mvc 是一种设计模式。struts框架，springMVC都基于这种框架；

Mvc中的M 表示三层中的mode层，表示业务数据和业务处理，相当于JavaBean。一个模型能为多个视图提供数据。这提高了应用程序的重用性；

V表示三层中的View层，用来显示数据，并接受用户的输入。视图不进行任何业务逻辑处理，

C 表示controller，用来业务逻辑的处理。处理后会返回一个视图；

过程如下:

1. 用户向服务器发送请求，请求被Spring 前端控制Servelt DispatcherServlet捕获；

2.  DispatcherServlet对请求URL进行解析，得到请求资源标识符（URI）。然后根据该URI，调用HandlerMapping获得该Handler，最后以HandlerExecutionChain对象的形式返回给DispatcherServlet；

 3.  DispatcherServlet 根据获得的Handler，选择一个合适的HandlerAdapter

4.  提取Request中的模型数据，填充Handler入参，**开始执行Handler（Controller)**。

5.  Handler执行完成后，向DispatcherServlet 返回一个ModelAndView对象；

6.  DispatcherServlet 根据返回的ModelAndView，选择一个适合的ViewResolver解析完之后将具体的view返回给DispatcherServlet ；

7. DispatcherServlet对拿到的视图view进行渲染（将Model的数据填充到View中，得到一个可展示的视图页面）

8. 将渲染结果返回给客户端。

# 6.对未来三年的职业规划是什么？

第一年，我会熟悉公司的业务流程，尽快的融入公司，这个期间我可能会遇到很多没有学过甚至没有学过的技术，我都会以最快的速度去学习的。

第二年，我会看我在公司有没有晋升的机会，如果有的话我会努力争取，晋升个组长什么的，就是不知道以我现在的学历，能不能有这个机会。

第三年，如果我的工作发展不错，相对稳定的话，我会在公司长期发展。

# 对薪资有什么要求？

作为应届毕业生我对薪资没有太大的要求，我想我能学到更多的知识来提升自己，为公司创造价值。但我来到这个城市，我想我的薪资应该能支撑我一个月的花销，当然我是非常节约的。

# 8.讲一下你做过的项目（举例说明）。

# 9.进入公司如何展开你的工作?

我会尽快的熟悉公司的业务流程，搞好同事之间的关系，尽快融入公司。然后在实习期间尽快提升自己的技术，为公司创造价值。

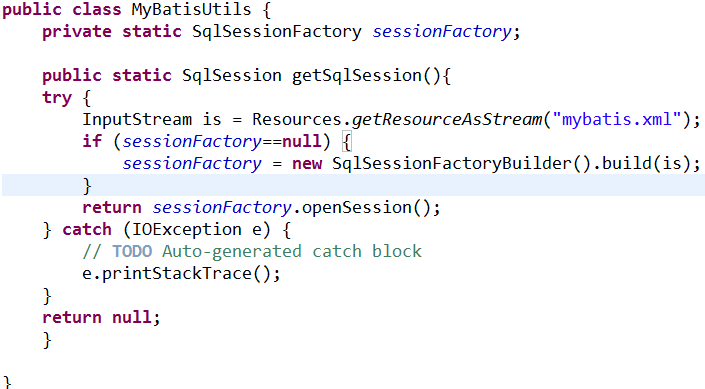
# 10.你对加班有什么看法？

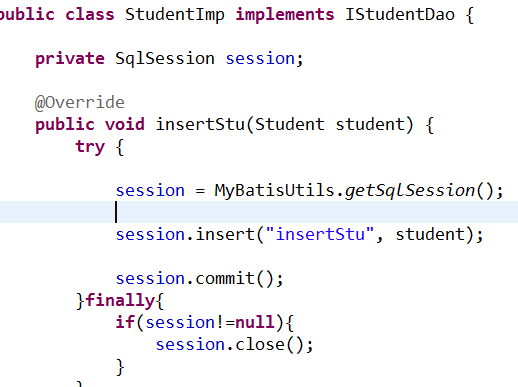
我愿意加班啊，出差也不介意的，我觉得只要不是加班到凌晨那种加班我都能接受。毕竟加班能提升自己的技术的嘛。

# 说说对Mybatis的理解？

MyBatis是ibatis的升级版，是一种orm工具，简化了数据库方面的操作，另外MyBatis消除了几乎所有的JDBC代码和参数的手工配置以及结果集的检索，myBatis使用简单的xml或者注解用于配置和原始映射，将接口和java的对象，映射成数据库中的记录；

**原理详解：**

**MyBatis应用程序根据XML配置文件创建SqlSessionFactory，SqlSessionFactory在根据配置，配置来源于两个地方，一处是配置文件，一处是Java代码的注解，获取一个SqlSession。SqlSession包含了执行sql所需要的所有方法，可以通过SqlSession实例直接运行映射的sql语句，完成对数据的增删改查和事务提交等，用完之后关闭SqlSession。**



# 11.1 说说**对struts2的理解**？

1、struts 是一个按MVC模式设计的Web层框架，其实它就是一个大大的servlet，这个servlet名为ActionServlet

**struts的优缺点：  
优点：**1.实现MVC模式，结构清晰，使开发者只关注业务逻辑的实现。  
2.有丰富的tag可以使用，struts的标签库，JSTL，EL，可以提高开发效率。  
3.通过一个配置文件，可以把握整个系统各部分之间的联系。使系统的脉络更加清晰。  
4.提供Exception处理机制。  
5.数据库连接池管理。  
6.支持I18N  
**缺点：**  
1.对事件支持不够. 在struts中，实际是一个表单Form对应一个Action类(或DispatchAction)，换一句话说：在Struts中实际是一个表单只能 对应一个事件，struts这种事件方式称为application event，application event和component event相比是一种粗粒度的事件;

**一．Struts2怎么处理一个用户请求的？**  
     1，首先用户发出一个请求，通过filterDispatcher拦截器拦截下来。   
     2， filterDispatcher会询问ActionMapper来决定这个请求是否需要调用某个Action。   
     3， 这个ActionMapper封装了请求的信息。如果ActionMapper决定需要调用某个Action，FilterDispatcher会按信息(action的名字,调用的方法名)创建一个Action的代理对象。实际上对Action的调用就是通过这个代理对象来实现的。   
     4，把请求交给这个代理，如果用户提交的请求没有指定方法，那么Struts2则默认excute方法。这个代理会创建一个调度器的实例。   
     这个调度器里面定义了一个invoke方法，这个方法就实现了拦截器的交互和执行Action里的excute方法。   
     5， 拦截器是以value栈的形式调用。   
      Struts2默认的有18个拦截器。第一个拦截器是一个异常拦截器，这样的话，我第一个拦截器把异常捕获到，那么以后就不用捕获其他拦截器出现的异常了。   
    6， 当Action执行完以后，调度器就会负责根据struts.xml中的配置找到对应的返回结果。   
     7， 然后调度器就返回所有的拦截器，通过response把结果集返回给客户。

# SVN和GIT是什么，区别是什么?

Svn 和git 都属于代码版本控制的工具

最大的区别在于：

Git是分布式的，而Svn不是分布的；

1.Git下载下来后，在OffLine状态下可以看到所有的Log,SVN不可以；

2.刚开始用时很狗血的一点，SVN必须先Update才能Commit,忘记了合并时就会出现一些错误，git还是比较少的出现这种情况。

3. Svn中的分支是一个完整的目录。且这个目录拥有完整的实际文件。如果工作成员想要开啟新的分支，那将会影响“全世界”！每个人都会拥有和你一样的分支。

而 git每个工作成员可以任意在自己的本地版本库开啟无限个分支。我可以开一个分支，做我喜欢的事。完全不需担心妨碍其他工作成员。只要我不合并及提交到主要版本库，没有一个工作成员会被影响。等到我不需要这个分支时， 我只要把它从我的本地版本库删除即可。无痛无痒。

# Spring 的注解是什么（关键字）?

@Component:标准一个普通的spring Bean类。

@Controller:标注一个控制器组件类。

@Service:标注一个业务逻辑组件类。

@Repository:标注一个DAO组件类。

//与@Component功能相同意义不同的注解还有三个

//1. Controller 注解在处理器上 （SpringMVC）

//2. Repository 注解在Dao上

//3. Service 注解在service上

# 数据库为什么要用索引？

索引会增加速度，DB在执行一条Sql语句的时候，默认的方式是根据搜索条件进行全表扫描，遇到匹配条件的就加入搜索结果集合。如果我们对某一字段增加索引，查询时就会先去索引列表中一次定位到特定值的行数，大大减少遍历匹配的行数，所以能明显增加查询的速度；索引比并不是越多越好，因为索引是需要存储空间的，而且有个致命缺点是对于update/insert/delete的每次执行，字段的索引都必须重新计算更新。

# 数据库（建表，查询，排序，事务逻辑和存储过程）

# json和xml的区别?

XML是标准通用标记语言 (SGML) 的子集，非常适合 Web 传输。

XML使用DTD(document type definition)文档类型定义来组织数据;格式统一，具有跨平台和跨语言的特性；

JSON(**[JavaScript](http://lib.csdn.net/base/javascript" \o "JavaScript知识库" \t "http://blog.csdn.net/wangpanbaoding/article/details/_blank)** Object Notation)一种轻量级的数据交换格式，JSON采用兼容性很高的、完全独立于语言文本格式，可在不同平台之间进行数据交换。

# aop原理是什么?

面向切面编程，把散落在程序中的公共部分提取出来，做成切面类，这样的好处在于，代码的可重用，一旦涉及到该功能的需求发生变化，只要修改该代码就行，否则，你要到处修改，如果只要修改1、2处那还可以接受，万一有1000处呢。

AOP底层的东西就是**JDK动态代理和CGLIB代理**，说白了就是增强类的功能。

最常用的AOP应用在数据库连接以及事务处理上。

# 什么是ioc？

IOC控制反转，也叫依赖注入（DI） 意思是把对象之间的依赖关系通过spring配置来实现，目的是为了解耦。降低类与类之间的耦合度，spring的ioc是，我们只需要再A类之间声明一个B类的对象即可，并不需要像原来一样去new（） 一个B类对象，A，B对象都由spring这个容器来创建，然后通过注入的方式把B类对象注入给A类对象，

另外 spring的注入常见的分为三种

1. set注入
2. 构造注入
3. 自动装配

# 18.Spring的优点？

## spring 的优点？ 1.降低了组件之间的耦合性 ，实现了软件各层之间的解耦  2.可以使用容易提供的众多服务，如事务管理，消息服务等  3.容器提供单例模式支持  4.容器提供了AOP技术，利用它很容易实现如权限拦截，运行期监控等功能  5.容器提供了众多的辅助类，能加快应用的开发  6.spring对于主流的应用框架提供了集成支持，如hibernate，JPA，Struts等  7.spring属于低侵入式设计，代码的污染极低  8.独立于各种应用服务器  9.spring的DI机制降低了业务对象替换的复杂性  10.Spring的高度开放性，并不强制应用完全依赖于Spring，开发者可以自由选择spring的部分或全部

## **19.面向切面编程在spring中主要表现为两个方面** 1.面向切面编程提供声明式事务管理  2.spring支持用户自定义的切面

# 线程死锁的原因

所谓死锁，是指多个进程循环等待它方占有的资源而无限期地僵持下去的局面。很显然，如果没有外力的作用，那麽死锁涉及到的各个进程都将永远处于封锁状态。从上面的例子可以看出，计算机系统产生死锁的根本原因就是资源有限且操作不当。

1. 产生死锁的必要条件

〈1〉互斥条件。即某个资源在一段时间内只能由一个进程占有，不能同时被两个或两个以上的进程占有。这种独占资源如CD-ROM驱动器，打印机等等，必须在占有该资源的进程主动释放它之后，其它进程才能占有该资源。这是由资源本身的属性所决定的。如独木桥就是一种独占资源，两方的人不能同时过桥。

    〈2〉不可抢占条件。进程所获得的资源在未使用完毕之前，资源申请者不能强行地从资源占有者手中夺取资源，而只能由该资源的占有者进程自行释放。如过独木桥的人不能强迫对方后退，也不能非法地将对方推下桥，必须是桥上的人自己过桥后空出桥面（即主动释放占有资源），对方的人才能过桥。

    〈3〉占有且申请条件。进程至少已经占有一个资源，但又申请新的资源；由于该资源已被另外进程占有，此时该进程阻塞；但是，它在等待新资源之时，仍继续占用已占有的资源。还以过独木桥为例，甲乙两人在桥上相遇。甲走过一段桥面（即占有了一些资源），还需要走其余的桥面（申请新的资源），但那部分桥面被乙占有（乙走过一段桥面）。甲过不去，前进不能，又不后退；乙也处于同样的状况。

    〈4〉循环等待条件。存在一个进程等待序列{P1，P2，...，Pn}，其中P1等待P2所占有的某一资源，P2等待P3所占有的某一源，......，而Pn等待P1所占有的的某一资源，形成一个进程循环等待环。就像前面的过独木桥问题，甲等待乙占有的桥面，而乙又等待甲占有的桥面，从而彼此循环等待。

  上面我们提到的这四个条件在死锁时会同时发生。也就是说，只要有一个必要条件不满足，则死锁就可以排除。

# Struts VS Spring两种MVC框架比较

**一、Struts**

Struts是Apache软件基金下Jakarta项目的一部分。Struts框架的主要架构设计和开发者是Craig R.McClanahan。Struts是目前Java Web MVC框架中不争的王者。经过长达五年的发展，Struts已经逐渐成长为一个稳定、成熟的框架，并且占有了MVC框架中最大的市场份额。但是Struts某些技术特性上已经落后于新兴的MVC框架。面对Spring MVC、Webwork2 这些设计更精密，扩展性更强的框架，Struts受到了前所未有的挑战。但站在产品开发的角度而言，Struts仍然是最稳妥的选择。

Struts有一组相互协作的类（组件）、Serlvet以及jsp tag lib组成。基于struts构架的web应用程序基本上符合JSP Model2的设计标准，可以说是MVC设计模式的一种变化类型。根据上面对framework的描述，我们很容易理解为什么说Struts是一个web framwork，而不仅仅是一些标记库的组合。但 Struts 也包含了丰富的标记库和独立于该框架工作的实用程序类。Struts有其自己的控制器（Controller），同时整合了其他的一些技术去实现模型层（Model）和视图层（View）。在模型层，Struts可以很容易的与数据访问技术相结合，包括EJB,JDBC和Object Relation Bridge。在视图层，Struts能够与JSP, Velocity Templates,XSL等等这些表示层组件想结合。

**Struts的体系结构**

struts framework是MVC 模式的体现，下面我们就从分别从模型、视图、控制来看看struts的体系结构（Architecture）。

**从视图角度（View）**

主要由JSP建立，struts自身包含了一组可扩展的自定义标签库（TagLib），可以简化创建用户界面的过程。目前包括：Bean Tags，HTML Tags，Logic Tags，Nested Tags，Template Tags这几个Taglib。有关它们的详细资料请参考struts用户手册。

**从模型角度（Model）**

模型主要是表示一个系统的状态（有时候，改变系统状态的业务逻辑操作也划分到模型中）。在Struts中，系统的状态主要有ActionForm Bean体现，一般情况下，这些状态是**非持久性**的。如果需要将这些状态转化为持久性数据存储，Struts本身也提供了Utitle包，可以方便的与数据库操作。

**从控制器角度（Controller）**

在Struts framework中，Controller主要是**ActionServlet**，但是对于业务逻辑的操作则主要由Action、ActionMapping、ActionForward这几个组件协调完成（也许这几个组件，应该划分到模型中的业务逻辑一块）。其中，Action扮演了真正的业务逻辑的实现者，而**ActionMapping和ActionForward则指定了不同业务逻辑或流程的运行方向。**

对于Struts 如何控制、处理客户请求，让我们通过对struts的四个核心组件介绍来具体说明。这几个组件就是：ActionServlet。Action Classes，Action Mapping（此处包括ActionForward），ActionFrom Bean。

**二、Spring**

Spring实际上是《Expert One-on-One J2EE Design and Development》一书中所阐述的设计思想的具体实现。在One-on-One一书中，Rod Johnson 倡导J2EE 实用主义的设计思想，并随书提供了一个初步的开发框架实现（interface21 开发包）。而Spring 正是这一思想的更全面和具体的体现。Rod Johnson在interface21 开发包的基础之上，进行了进一步的改造和扩充，使其发展为一个更加开放、清晰、全面、高效的开发框架。

Spring是一个开源框架，由Rod Johnson创建并且在他的著作《J2EE设计开发编程指南》里进行了描述。它是为了解决企业应用开发的复杂性而创建的。Spring使使用基本的JavaBeans来完成以前只可能由EJB完成的事情变得可能了。然而，Spring的用途不仅限于服务器端的开发。从简单性、可测试性和松耦合的角度而言，任何Java应用都可以从Spring中受益。

简单来说，**Spring是一个轻量的控制反转和面向切面的容器框架**。当然，这个描述有点过于简单。但它的确概括出了Spring是做什么的。为了更好地理解Spring，让我们分析一下这个描述：

**1、轻量**  
从大小与开销两方面而言Spring都是轻量的。完整的Spring框架可以在一个大小只有1MB多的JAR文件里发布。并且Spring所需的处理开销也是微不足道的。此外，Spring是**非侵入式的**：典型地，Spring应用中的对象不依赖于轻量，从**大小与开销**两方面而言Spring都是轻量的。完整的Spring框架可以在一个大小只有1MB多的JAR文件里发布。并且Spring所需的处理开销也是微不足道的。此外，Spring是非侵入式的：典型地，Spring应用中的对象不依赖于Spring的特定类。

**2、控制反转**Spring通过一种称作控制反转（IoC）的技术促进了松耦合。当应用了IoC，对象**被动地传递它们的依赖而不是自己创建或者查找依赖对象**。你可以认为IoC与JNDI相反??不是对象从容器中查找依赖，而是容器在对象初始化时不等被请求就将依赖传递给它。

**3、面向切面**Spring包含对**面向切面编程**的丰富支持，允许通过分离应用的业务逻辑与系统服务（例如审计与事物管理）进行内聚性的开发。应用对象只做它们应该做的，完成业务逻辑，仅此而已。它们并不负责（甚至是意识）其它的系统关注点，例如日志或事物支持。

**4、容器**Spring包含和管理应用对象的配置和生命周期，在这个意义上它是一种容器。你可以配置你的每个bean如何被创建?基于一个配置原形为你的bean创建一个单独的实例或者每次需要时都生成一个新的实例以及它们是如何相互关联的。然而，Spring不应该被混同于传统的重量的EJB容器，它们经常是庞大与笨重的，难以使用。

框架：Spring是由简单的组件配置和组合复杂的应用成为可能。在Spring中，应用对象被声明式地组合，典型地是在一个XML文件里。Spring也提供了很多基础功能（事务管理、持久性框架集成等等），将应用逻辑的开发留给了你。

所有Spring的这些特征使你能够编写更干净、更可管理、并且更易于测试的代码。它们也为Spring中的各种子框架提供了基础。