1、谈谈对 spring 框架的了解 ，spring 有什么作用（IOC，AOP），spring 的核心是什么？

回答：Spring 是一个开源框架，它是为了解决企业应用开发的复杂性而创建的。框架的主要优势之一就是其分层架构，分层架构允许使用者选择使用哪一个组件，同时为 J2EE 应用程序开发提供集成的框架。Spring 使用基本的 JavaBean 来完成以前只可能由 EJB 完成的事情。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 是一个分层的 JavaSE/EE full-stack(一站式) 轻量级开源框架。

Spring 的核心是控制反转（IoC）和面向切面（AOP）。

Spring 的作用：

1）方便解耦，简化开发：通过 Spring 提供的 IOC 容器，我们可以将对象之间的依赖关系交由 Spring 进行控制，避免编码所造成的过度程序耦合。有了 Spring，用户不必再为单例模式类、属性文件解析等这些很多底层的需求编写代码，可以更专注于上层的应用。

2）AOP 编程的支持：通过 Spring 提供的 AOP 功能，方便进行面向切面的编程，许多不容易用传统 OOP 实现的功能可以通过 AOP 轻松应付。

3）声明式事务的支持：在 Spring 中，我们可以从单调烦闷的事务管理代码中解脱出来，通过声明式方式灵活地进行事务的管理，提供开发效率和质量。

4）方便程序的测试：可以用非容器依赖的编程方式进行几乎所有的

测试工作，在 Spring 里，测试不再是昂贵的操作，而是随手可做的事情。例如：Spring 对 Junit4 支持，可以通过注解方便的测试 Spring 程序。

5）方便集成各种优秀框架：Spring 不排斥各种优秀的开源框架，相反，Spring 可以降低各种框架的使用难度，Spring 提供了对各种优秀框架的直接支持。

6）降低 Java EE API 的使用难度：Spring 对很多难用的 Java EE API 提供了一个薄薄的封装层，通过 Spring 的简易封装，这些 Java EE API 的使用难度大为降低。

注：Spring 的源码设计精巧、结构清晰、匠心独运，处处体现着大师对 Java 设计模式灵活运用以及 Java 技术的高深造诣。Spring 框架源码无疑是 Java 技术的最佳实践范例。如果想在短时间内迅速提高自己的 Java 技术水平和应用开发水平，学习和研究 Spring 源码将会使你收到意想不到的效果。

2、SpringMVC 的常用注解，执行流程，都有哪几种解析器，必须要返回 modelAndView 么，SpringMVC 接收一个 json 数据时怎么处理的，用什么注解?

回答：SpringMVC 常用注解：

1）@Controller 用于标记在一个类上，使用它标记的类就是一个 SpringMVC Controller 对象。

2）@RequestMapping 是一个用来处理请求地址映射的注解，可用于

类或方法上。用于类上，表示类中的所有响应请求的方法都是以该地址作为父路径。

3）@Resource 和@Autowired 两者都是做 bean 的注入时使用，其实@Resource 并不是 Spring 的注解，它的包是 java.annotation.Resource，需要导入，但是 Spring 支持该注解的注入。@Autowired 为 Spring 提供的注解，需要导入包 org.springframework.beans.factory.annotation.Autowired；只按照 byType 注入。

4）@PathVariable 用于将请求 URL 中的模板变量映射到功能处理方法的参数上，即取出 uri 模板中的变量作为参数。

5）@Cookie 用来获取 Cookie 中的值。

6）@RequestParam 用于将请求参数区数据映射到功能处理方法的参数上。

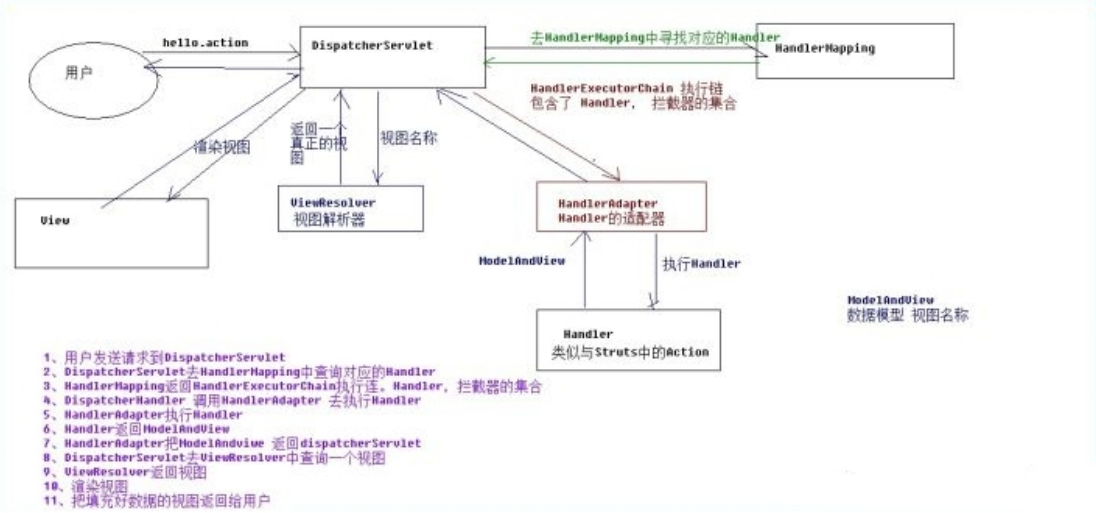
7）@SessionAttrbutes 即将值放到 session 作用域中，写在 class 上面；除了可以通过属性名指定需要放到会话中的属性外（value 属性值），还可以通过模型属性的对象类型指定哪些模型属性需要放到会话中（types 属性值）。

8)@ModelAttribute 代表的是: 该 Controller 的所有方法在调用前，先执行此@ModelAttribute 方法，可用于注解和方法参数中，可以把这个@ModelAttribute 特性，应用在 BaseController 当中，所有的 Controller 继承 BaseController,即可实现在调用 Controller 时，先执行@ModelAttribute 方法。

9）@ResponseBody 该注解用于将 Controller 的方法返回的对象，通过适当的 HttpMessageConverter 转换为指定格式后，写入到 Response 对象的 body 数据区；返回数据不是 html 标签的页面，而是其他某种格式的数据时（如 son、xml 等）使用。

10）@RequestBody 该用于读取 Request 请求的 body 部分数据，使用系统默认配置的 HttpMessageConverter 进行解析，然后把相应的数据绑定到要返回的对象上；再把 HttpMessageConverter 返回的对象数据绑定到 controller 中方法的参数上。

SpringMVC 的执行流程：



常用的视图解析器：（AbstractCachingViewResolver、UrlBasedViewResolver、XmlViewResolver、BeanNameViewResolver、ResourceBundleViewResolver、FreeMarkerViewResolver、VolocityViewResolver）

必须要返回 modelAndView 么，SpringMVC 接收一个 json 数据时怎么处理的,用什么注解?

不是，使用@ResponseBody 把后台 pojo 转换 json 对象，返回到页面;

@RequestBody 接受前台 json，把 json 数据自动封装 pojo。

3、Spring 依赖注入的几种方式？

回答：Spring 常用的依赖注入方式是：Setter 方法注入、构造器注入、Filed 注入(用于注解方式)；

1）Setter 方法注入：首先要配置被注入的 bean，在该 bean 对应的类中，应该有要注入的对象属性或者基本数据类型的属性。

2）构造器注入：在 PersonBiz 类中注入 PersonDAO 和一个 String 类型的数据；在该类中，不用为 PersonDAO 属性和 String 数据类型的属性设置 setter 方法，但是需要生成该类的构造方法；在配置文件中配置该类的 bean，并配置构造器，在配置构造器中用到了<constructor-arg>节点。

3）Filed 注入(用于注解方式)：在 spring 中，注入依赖对象可以采用手工装配或自动装配，在实际应用开发中建议使用手工装配，因为自动装配会产生许多未知情况，开发人员无法预见最终的装配结果。

4、Spring 设置为单例 ，那么线程安全问题怎么解决？

回答：首先了解一下在什么情况下，单例的 Bean 对象存在线程安全问题，当 Bean 对象对应的类存在可变的成员变量并且其中存在改变这个变量的线程时，多线程操作该 Bean 对象时会出现线程安全；产生的原因是当多线程中存在线程改变了 bean 对象的可变成员变量时，其他线程无法访问该 bean 对象的初始状态，从而造成数据错乱；

解决办法：1)在 Bean 对象中尽量避免定义可变的成员变量；2)在 Bean 对象中定义一个 ThreadLocal 成员变量，将需要的可变成员变量保存在 ThreadLocal 中。

5、 Struts2 和 SpringMVC 的区别 ？

回答：1）Struts2 是类级别上的拦截，一个 Action 对应一个 request 上下文，SpringMVC 是方法级别的拦截，一个方法对应一个 request 上下文；而且 Struts2 过滤后是去 Struts2 配置文件中 Action，而 SpringMVC 过滤后是去 Controller 中找对应于@RequestMapping 注解的 uri 绑定的方法，从这里看 Struts2 使用起来更麻烦，因为你要每个类的请求你都要配置对应的拦截器。

2）因为拦截器的原因，导致 Struts2 的 action 比较乱，因为它要定义属性来获取请求中参数的数据，而属性在一个类的方法间是共享的（方法间不能独享 request、response 数据），所以会有点乱；SpringMVC 方法之间基本独立，独享 request、response 之间的数据。请求数据通过参数获取，处理结果通过 model Map 交回给框架（方法间不共享变量）。

3）SpringMVC 集成了 Ajax，使用非常方便，只需要一个注解 @ResponseBody 就可以实现，然后直接返回响应文本即可，而 Struts2 拦截器集成了 Ajax，在 Action 中处理时一般必须安装插件或者自己写代码集成进去，使用起来也相对不方便。

6、Struts2 可以是单例的吗 为什么？

回答：Struts2 的 Action 是多例模式的，也就是每次请求产生一个 Action 的对象，但是通过 spring 可以控制成单例，控制成单例的话，可以减少内存的消耗，因为可以保存 action 不被销毁，但不保证这些数据在多线程的环境下不被相互影响。

7、什么是有状态对象，什么是无状态对象？

回答：

1）有状态对象，多线程环境下不安全，那么适合用 Prototype 原型模式。Prototype：每次对 bean 的请求都会创建一个新的 bean 实例；

2）无状态对象(Stateless Bean)，就是没有实例变量的对象，不能保存数据，是不变类，是线程安全的。

8、Spring 的常用注解？

回答：@Controller：标记于一个类上面；用来注解这个 bean（类）是 MVC 模型中的一个控制层，使分发处理器识别到该类，该类会被 spring 的 auto-scan 扫到纳入管理。

@RequestMapping：标记于一个被@Controller 标注的类上；标记于被@Controller 标注类里面的方法上面；表示该被标注类下面所有方法的父类标注。

@Service：用于标注业务层组件上；标注与业务层组件上表示定义一个 bean，自动根据所标注的组件名称实例化一个首字母为小写的

bean。

@Resource：标注于字段上或者 setter 方法上，@Resource 默认按 ByName 进行自动装配；用来自动装配 Bean，激活一个命名资源的依赖注入，@Resource 属性 name 可以定义被自动装配 Bean 的名称。

@Autowired：与@Resource 的用法和作用基本一致；@Resource 属于 J2EE，@Autowired 属于 Spring；@Autowired 是根据类型（ByType）进行自动装配的；@Autowired 没有 name 属性，如果要按名称进行装配，需要配合@Qualifier 使用。

@Repository：是用户标注数据访问层组件（DAO 层）；实现 DAO 访问，将类识别为 Bean，同时它将所标注的类中抛出的数据访问异常封装为 Spring 的数据访问异常类型。

@Component：泛指组件，当组件不好归类的时候，我们可以使用这个注解进行标注；和前面@Service、@Repository、@Controller 一样，只是它们比@Component 更细化。

9、报表用的什么生成图表？

回答：1）JFreeChart ：http://www.jfree.org/jfreechart/；

2）ECharts：http://git.oschina.net/bree/ECharts；

3） jCharts：http://jcharts.sourceforge.net/；

4）DynamicReports：http://www.dynamicreports.org/；

5）JChartLib：http://sourceforge.net/projects/jchartlib/；

6）SWTChart：http://www.swtchart.org/；

以上 JAVA 常用的生成图表工具。

10、Spring 是如何管理事务的？

回答：Spring 的事务机制包括声明式事务和编程式事务；编程式事务管理（Spring 推荐使用 TransactionTemplate，实际开发中使用声明式事务较多）；声明式事务管理（将我们从复杂的事务处理中解脱出来，获取连接，关闭连接，事务提交、回滚、异常处理等这些操作都不用我们处理了，Spring 都会帮我们处理；使用了 AOP 面向切面编程实现的，本质就是在目标方法执行前后进行拦截。在目标方法执行前加入或创建一个事务，在执行方法执行后，根据实际情况选择提交或是回滚事务。）

Spring 事务管理主要包括 3 个接口，Spring 的事务主要由它们三个共同完成的。

1）PlatformTransactionManager：事务管理器（主要用于平台相关事务的管理），主要有三个方法（commit 事务提交；rollback 事务回滚；getTransaction 获取事务状态）；

2）TransactionDefinition：事务定义信息（用来定义事务相关的属性，给事务管理器 PlatformTransactionManager 使用），主要有四个方法（getIsolationLevel 获取隔离级别、getPropagationBehavior 获取传播行为、getTimeout 获取超时时间、isReadOnly 是否只读）；

3）TransactionStatus：事务具体运行状态（事务管理过程中，每个时间点事务的状态信息）。

声明式事务：优点（不需要在业务逻辑代码中编写事务相关代码，只需要在配置文件配置或使用注解（@Transaction），这种方式没有入侵性）；

缺点（声明式事务的最细粒度作用于方法上，如果像代码块也有事务需求，只能变通下，将代码块变为方法）。

11、简单说说你知道的 spring 的底层？

回答：1）Spring 对 Bean 进行实例化（相当于程序中的 new Xx()）

2）Spring 将值和 Bean 的引用注入进 Bean 对应的属性中

3）如果 Bean 实现了 BeanNameAware 接口，Spring 将 Bean 的 ID 传递给 setBeanName()方法（实现 BeanNameAware 清主要是为了通过 Bean 的引用来获得 Bean 的 ID，一般业务中是很少有用到 Bean 的 ID 的）

4）如果 Bean 实现了 BeanFactoryAware 接口，Spring 将调用 setBeanDactory(BeanFactory bf)方法并把 BeanFactory 容器实例作为参数传入。（实现 BeanFactoryAware 主要目的是为了获取 Spring 容器，如 Bean 通过 Spring 容器发布事件等）

5）如果 Bean 实现了 ApplicationContextAwaer 接口，Spring 容器将调用 setApplicationContext(ApplicationContext ctx)方法，把 y 应用上下文作为参数传入.(作用与 BeanFactory 类似都是为了获取 Spring 容器，不同的是 Spring 容器在调用 setApplicationContext 方法时会把它自己作为 setApplicationContext 的参数传入，而

Spring 容器在调用 setBeanDactory 前需要程序员自己指定（注入）setBeanDactory 里的参数 BeanFactory ）

6）如果 Bean 实现了 BeanPostProcess 接口，Spring 将调用它们的 postProcessBeforeInitialization（预初始化)方法(作用是在 Bean 实例创建成功后对进行增强处理，如对 Bean 进行修改，增加某个功能）

7）如果 Bean 实现了 InitializingBean 接口，Spring 将调用它们的 afterPropertiesSet 方法，作用与在配置文件中对 Bean 使用 init-method 声明初始化的作用一样，都是在 Bean 的全部属性设置成功后执行的初始化方法。

8）如果 Bean 实现了 BeanPostProcess 接口，Spring 将调用它们的 postProcessAfterInitialization（后初始化）方法（作用与 6 的一样，只不过 6 是在 Bean 初始化前执行的，而这个是在 Bean 初始化后执行的，时机不同)

9）经过以上的工作后，Bean 将一直驻留在应用上下文中给应用使用，直到应用上下文被销毁

10）如果 Bean 实现了 DispostbleBean 接口，Spring 将调用它的 destory 方法，作用与在配置文件中对 Bean 使用 destory-method 属性的作用一样，都是在 Bean 实例销毁前执行的方法。

12、说说 solr 的底层 ？

回答：1） 索引过程：（1、有一系列被索引文件；2、被索引文件经

过语法分析和语言处理形成一系列词(Term)；3、经过索引创建形成词典和反向索引表；4、通过索引存储将索引写入硬盘。）

2）搜索过程：（1、用户输入查询语句；2、 对查询语句经过语法分析和语言分析得到一系列词(Term)；3、通过语法分析得到一个查询树；4、通过索引存储将索引读入到内存；5、利用查询树搜索索引，从而得到每个词(Term)的文档链表，对文档链表进行交，差，并得到结果文档；6、将搜索到的结果文档对查询的相关性进行排序；7、返回查询结果给用户。）

13、Solr 如何搭建，简单介绍一下，你用的什么版本？

回答：solr-4.10.3 搭建：1）下载 solr-4.10.3 版本，JDK 需要 1.7 及以上版本；2）解压 solr-4.10.3；3）创建 solr 工程；4）部署到 tomcat 容器；5）新建 solrCore，在此目录下新建 core2 文件夹；6）查询数据表数据；7）java 通过 solr 查询数据库表。

14、Mybatis 和 hibernate 的区别？

回答：

mybatis：

1）入门简单，即学即用，提供数据库查询的自动对象绑定功能，而且延续了很好的 SQL 使用经验，对于没有那么高的对象模型要求的项目来说，相当完美；

2）可以进行更为细致的 SQL 优化，可以减少查询字段；

3）缺点就是框架还是比较简陋，功能尚有缺失，虽然简化了数据绑定代码，但是整个底层数据库查询实际还是要自己写的，工作量也比较大，而且不太容易适应快速数据库修改；

4）二级缓存机制不佳。

hibernate：

1）功能强大，数据库无关性好，O/R 映射能力强，如果你对 Hibernate 相当精通，而且对 Hibernate 进行了适当的封装，那么你的项目整个持久层代码会相当简单，需要写的代码很少，开发速度很快，非常爽；

2）有更好的二级缓存机制，可以使用第三方缓存；

3）缺点就是学习门槛不低，要精通门槛更高，而且怎么设计 O/R 映射，在性能和对象模型之间如何取得平衡，以及怎样用好 Hibernate 方面需要你的经验和能力都很强才行。

15、对于 hibernate3,4,5 有什么了解，其中的特性是什么？

回答：

hibernate4 中的新特性基于 hibernate3 中的改变：

1）数据库方言的设置（在 3.3 版本中连接 MySQL 数据库只需要指明 MySQLDialect 即可。在 4.1 版本中可以指出 MySQL5Dialect）

2）buildSessionFactory（4.1 版本中 buildSessionFactory()已经被 buildSessionFactory(ServiceRegistry ServiceRegistry)取代）

3）annotation（4.1 版本中推荐使用 annotation 配置，所以在引进 jar 包时把 requested 里面的包全部引进来就已经包含了 annotation

必须包了）

4）事务，hibernateTemplate（hibernate4 已经完全可以实现事务了与 spring3.1 中的 hibernatedao，hibernateTemplate 等有冲突，所以 spring3.1 里已经不提供 hibernatedaosupport，hibernateTemplate）

5）自动建表（hibernate4.1 已经可以自动建表，所以开发时只需要自己开发类然后配置好久 OK）

hibernate5 的新特性：1）明确了关联关系映射；2）明确 Hibernate 检索优化；3）明确 Hibernate 缓存机制；4）明确 Hibernate 对事务并发控制的管理；5）明确 Hibernate 注解开发。

16、SpringMVC 的底层是基于什么实现的？

回答: Spring MVC 是基于 servlet 功能实现的，通过实现 Servlet 接口的 DispatcherServlet 来封装其核心功能实现，通过将请求分派给处理程序，同时带有可配置的处理程序映射、视图解析、本地语言、主题解析以及上传下载文件支持。

17、请罗列出您所理解的微服务架构应具有的关键组件及关键指标？

回答：1）负载平衡：Linkerd 提供了多种负载均衡算法，它们使用实时性能指标来分配负载并减少整个应用程序的尾部延迟。

2）熔断：Linkerd 包含自动熔断，将停止将流量发送到被认为不健康的实例，从而使他们有机会恢复并避免连锁反应故障。

3）服务发现：Linkerd 与各种服务发现后端集成，通过删除特定的(ad-hoc)服务发现实现来帮助您降低代码的复杂性。

4）动态请求路由：Linkerd 启用动态请求路由和重新路由，允许您使用最少量的配置来设置分段服务（staging service），金丝雀（canaries），蓝绿部署（blue-green deploy），跨 DC 故障切换和黑暗流量（dark traffic）。

5）重试次数和截止日期：Linkerd 可以在某些故障时自动重试请求，并且可以在指定的时间段之后让请求超时。

6）TLS：Linkerd 可以配置为使用 TLS 发送和接收请求，您可以使用它来加密跨主机边界的通信，而不用修改现有的应用程序代码。

7）HTTP 代理集成：Linkerd 可以作为 HTTP 代理，几乎所有现代 HTTP 客户端都广泛支持，使其易于集成到现有应用程序中。

8）透明代理：您可以在主机上使用 iptables 规则，设置通过 Linkerd 的透明代理。

9）gRPC：Linkerd 支持 HTTP/2 和 TLS，允许它路由 gRPC 请求，支持高级 RPC 机制，如双向流，流程控制和结构化数据负载。

10）分布式跟踪：Linkerd 支持分布式跟踪和度量仪器，可以提供跨越所有服务的统一的可观察性。

11）仪器仪表：Linkerd 支持分布式跟踪和度量仪器，可以提供跨越所有服务的统一的可观察性。

18、hibernate 的二级缓存有什么用?

回答：因为应用程序访问数据库，读写数据的代价非常高，而利用持久层的缓存可以减少应用程序与数据库之间的交互，即把访问过的数据保存到缓存中，应用程序再次访问已经访问过的数据，这些数据就可以从缓存中获取，而不必再从数据库中获取。同时如果数据库中的数据被修改或者删除，那么是该数据所对应的缓存数据，也会被同步修改或删除，进而保持缓存数据的一致性。

19、介绍一下 mybatis？

回答：1）入门简单，即学即用，提供数据库查询的自动对象绑定功能，而且延续了很好的 SQL 使用经验，对于没有那么高的对象模型要求的项目来说，相当完美；

2）二级缓存机制不佳。

3）可以进行更为细致的 SQL 优化，可以减少查询字段；

4）缺点就是框架还是比较简陋，功能尚有缺失，虽然简化了数据绑定代码，但是整个底层数据库查询实际还是要自己写的，工作量也比较大，而且不太容易适应快速数据库修改；

20、 Shiro 的原理？

回答：shiro 是 apache 的一个开源框架，是一个权限管理的框架，实现用户认证，用户授权。spring 中有 spring security（原名 Acegi），是一个权限框架，它和 spring 依赖过于紧密，没有 shiro 使用简单。shiro 不依赖与 spring，shiro 不仅可以实现 web 应用的

权限管理，还可以实现 c/s 系统，分布式系统权限管理，shiro 属于轻量级框架，越来越多的企业项目使用 shiro。使用 shiro 实现系统的权限管理，有效提高开发效率，从而降低开发成本。

shiro 架构有 3 个主要概念：Subject、SecurityManager、Realms。

1）Subject，正如我们在教程中所说，Subject 其实代表的就是当前正在执行操作的用户，只不过因为“User”一般指代人，但是一个“Subject”可以是人，也可以是任何的第三方系统，服务账号等任何其他正在和当前系统交互的第三方软件系统。

所有的 Subject 实例都被绑定到一个 SecurityManager，如果你和一个 Subject 交互，所有的交互动作都会被转换成 Subject 与 SecurityManager 的交互。

2）SecurityManager。SecurityManager 是 Shiro 的核心，他主要用于协调 Shiro 内部各种安全组件，不过我们一般不用太关心 SecurityManager，对于应用程序开发者来说，主要还是使用 Subject 的 API 来处理各种安全验证逻辑。

3）Realm，这是用于连接 Shiro 和客户系统的用户数据的桥梁。一旦 Shiro 真正需要访问各种安全相关的数据（比如使用用户账户来做用户身份验证以及权限验证)时，他总是通过调用系统配置的各种 Realm 来读取数据。

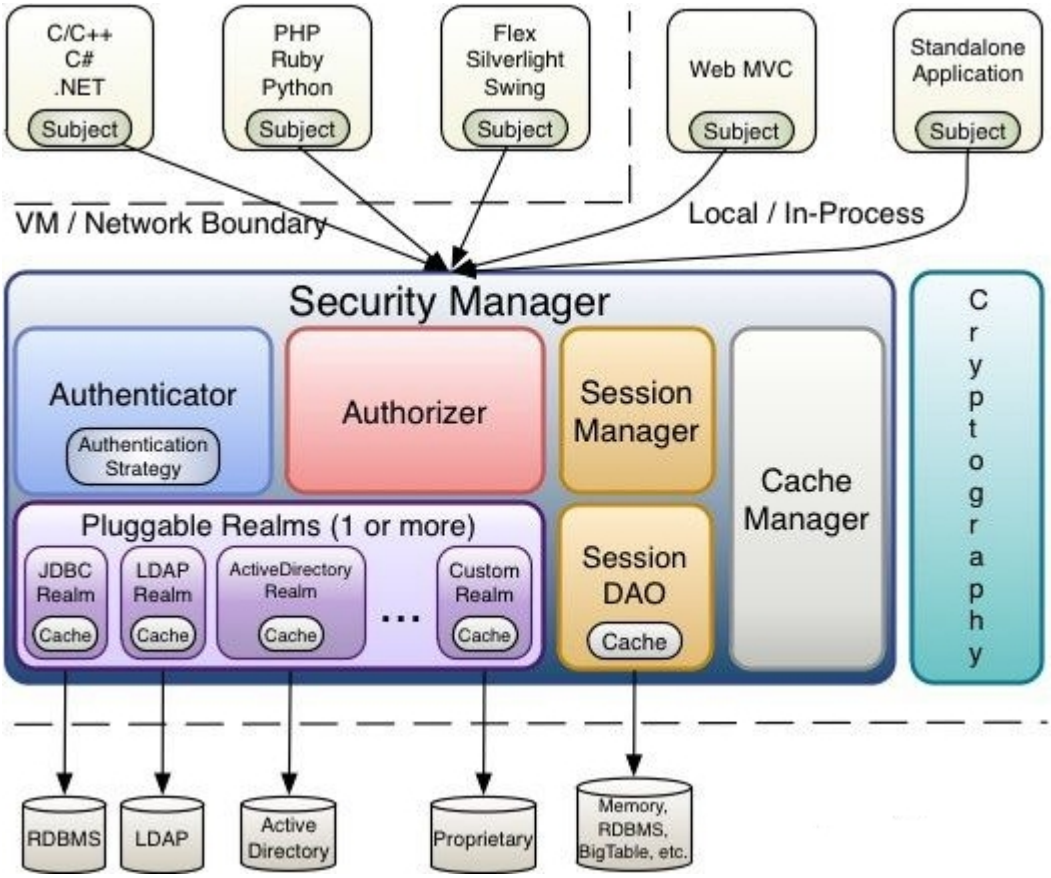
正因为如此，Realm 往往被看做是安全领域的 DAO，他封装了数据源连接相关的细节，将数据以 Shiro 需要的格式提供给 Shiro。当我们配置 Shiro 的时候，我们至少需要配置一个 Realm 来提供用户验证和

权限控制方面的数据。我们可能会给 SecurityManager 配置多个 Realm，但是不管怎样，我们至少需要配置一个。

Shiro 提供了几种开箱即用的 Realm 来访问安全数据源，比如 LDAP、关系数据库、基于 ini 的安全配置文件等等，如果默认提供的这几种 Realm 无法满足你的需求，那么你也可以编写自己的定制化的 Realm 插件。

和其他内部组件一样，SecurityManager 决定了在 Shiro 中如何使用 Realm 来读取身份和权限方面的数据，然后组装成 Subject 实例。

shiro 详细架构：



21、Webservice 是什么，怎么用 ？

回答：从表面上看，WebService 就是一个应用程序，它向外界暴露

出一个能够通过 Web 进行调用的 API。这就是说，你能够用编程的方法通过 Web 调用来实现某个功能的应用程序。从深层次上看，Web Service 是一种新的 Web 应用程序分支，它们是自包含、自描述、模块化的应用，可以在网络(通常为 Web)中被描述、发布、查找以及通过 Web 来调用; Web Service 便是基于网络的、分布式的模块化组件，它执行特定的任务，遵守具体的技术规范，这些规范使得 Web Service 能与其他兼容的组件进行互操作。它可以使用标准的互联网协议，像超文本传输协议 HTTP 和 XML，将功能体现在互联网和企业内部网上。WebService 平台是一套标准，它定义了应用程序如何在 Web 上实现互操作性。你可以用你喜欢的任何语言，在你喜欢的任何平台上写 Web Service；WebService 为 Internet 上的组件服务•通过网络提供，以 URL 定位方法调用•以 Internet 技术为基础•未来的分散式应用程序。

第一步：创建一个 Java 项目

第二步：创建一个类，加入 Webservice 注解

@WebService 是 jdk 提供的一个注解，它位于 javax.jws.*这个包中。

第三步：创建一个方法

第四步：在 main 方法中调用 jdk 提供的发布服务的方法

通过 EndPoint(端点服务)发布一个 webService。

Endpoint 也是 jdk 提供的一个专门用于发布服务的类，它的 publish 方法接收两个参数，一个是本地的服务地址，二是提供服务的类。它位于 javax.xml.ws.*包中。

static Endpoint.publish(String address, Object implementor)

在给定地址处针对指定的实现者对象创建并发布端点。

stop 方法用于停止服务。

EndPoint 发布完成服务以后，将会独立的线程运行。所以，publish 之后的代码，可以正常执行

22、solr 存数据是不是要创建索引？

回答：要创建，存数据之前需要创建标签并设置属性，需要创建的索引必须在 Schema 里面有。