1.

java ee的三种开发模式：

--- jsp, 最小型的项目，直接用jsp来编写，即，html和java代码写在一起

----jsp+javabean：中型项目，可以实现html代码和java代码的分离

----jsp + servlet +javabean大型项目，也是最好的方式，

也就是有些功能使用jsp来开发，有些功能用servlet来开发，jsp部分的后台逻辑调用javabean来完成，而servlet部分的业务逻辑就在servlet里边直接完成了

具体什么是servlet可看该文档下面部分对servlet的解析和例子，什么地方用servlet什么地方用jsp这个就看设计和开发者了.

2.

---拦截器：interceptors>

拦截器，在AOP（Aspect-Oriented Programming）中用于在某个方法或字段被访问之前，进行拦截然后在之前或之后加入某些操作。拦截是AOP的一种实现策略。

谈到拦截器，还有一个词大家应该知道——拦截器链（Interceptor Chain，在Struts 2中称为拦截器栈 Interceptor Stack）。

拦截器链就是将拦截器按一定的顺序联结成一条链。在访问被拦截的方法或字段时，拦截器链中的拦截器就会按其之前定义的顺序被调用。

大部分时候，拦截器方法都是通过代理的方式来调用的。Struts 2的拦截器实现相对简单。

当请求到达Struts 2的ServletDispatcher时，Struts 2会查找配置文件，并根据其配置实例化相对的拦截器对象，然后串成一个列表（list），

最后一个一个地调用列表中的拦截器。

---过滤器：filter

它是一个程序，它先于与之相关的servlet或JSP页面运行在服务器上。过滤器可附加到一个或多个servlet或JSP页面上，并且可以检查进入这些资源的请求信息。

在Servlet作为过滤器使用时，它可以对客户的请求进行处理。处理完成后，它会交给下一个过滤器处理，这样，客户的请求在过滤链里逐个处理，直到请求发送到目标为止。

例如，某网站里有提交“修改的注册信息”的网页，当用户填写完修改信息并提交后，服务器在进行处理时需要做两项工作：判断客户端的会话是否有效；

对提交的数据进行统一编码。这两项工作可以在由两个过滤器组成的过滤链里进行处理。当过滤器处理成功后，把提交的数据发送到最终目标；

如果过滤器处理不成功，将把视图派发到指定的错误页面。

---拦截器与过滤器的区别：

它俩基本没啥使用的先后或者冲突，拦截器说白了就是一种AOP技术，可以放在程序的任何地方，就是对某个固定的执行点，进行嵌入，让其执行完动作A之后，都执行

我们编写的拦截器，比如，请求结束之后，我们都要求记录日志，这个时候我们可以使用拦截器了。

而过滤器，则是用于对用户的请求进行预处理，例如，请求来之前，我们都需要判断用户的ip,如果ip不合格，则调整到错误页面。

例如，某些页面，我们需要判断用户是否登录了，这些地方，我们都可以做成统一的过滤器，

需要说明的是，拦截器和过滤器都需要配置，基本上对java来说，一切都是配置啊。

----其实我们thinkphp的框架里边，用了很多面向切面的变成技术，里边叫行为扩展，就是在很多标签位里边，定义自己的行为，这些行为就是拦截器。

3.

struts2里边，请求在执行了一序列的过滤器或者拦截器之后，达到了核心控制器FilterDispatch,由核心控制器来读取配置文件上struts.xml来根据里边的配置

调用相应的Action来处理用户的请求，当Action处理完成之后，又返回给FilterFispatch，由它再查找配置文件struts.xml,查找对应的页面跳转。

目前最新版：2.5.X

4.

Hibernate是一个开放源代码的对象关系映射框架，它对JDBC进行了非常轻量级的对象封装，它将POJO与数据库表建立映射关系，是一个全自动的orm框架，hibernate可以自动生成SQL语句，自动执行，

使得Java程序员可以随心所欲的使用对象编程思维来操纵数据库。 Hibernate可以应用在任何使用JDBC的场合，

既可以在Java的客户端程序使用，也可以在Servlet/JSP的Web应用中使用。

Hibernate的API一共有6个，分别为:Session、SessionFactory、Transaction、Query、Criteria和Configuration。

通过这些接口，可以对持久化对象进行存取、事务控制

Session

Session接口负责执行被持久化对象的CRUD操作(CRUD的任务是完成与数据库的交流，包含了很多常见的SQL语句)。

但需要注意的是Session对象是非线程安全的。同时，Hibernate的session不同于JSP应用中的HttpSession。

这里当使用session这个术语时，其实指的是Hibernate中的session，而以后会将HttpSession对象称为用户session。

SessionFactory

SessionFactory接口负责初始化Hibernate。它充当数据存储源的代理，并负责创建Session对象。这里用到了工厂模式。

需要注意的是SessionFactory并不是轻量级的，因为一般情况下，一个项目通常只需要一个SessionFactory就够，

当需要操作多个数据库时，可以为每个数据库指定一个SessionFactory。

Transaction

Transaction 接口是一个可选的API，可以选择不使用这个接口，取而代之的是Hibernate 的设计者自己写的底层事务处理代码。

Transaction 接口是对实际事务实现的一个抽象，这些实现包括JDBC的事务、JTA 中的UserTransaction、甚至可以是CORBA 事务。

之所以这样设计是能让开发者能够使用一个统一事务的操作界面，使得自己的项目可以在不同的环境和容器之间方便地移植。

Query

Query接口让你方便地对数据库及持久对象进行查询，它可以有两种表达方式：HQL语言或本地数据库的SQL语句。

Query经常被用来绑定查询参数、限制查询记录数量，并最终执行查询操作。

Criteria

Criteria接口与Query接口非常类似，允许创建并执行面向对象的标准化查询。值得注意的是Criteria接口也是轻量级的，

它不能在Session之外使用。

Configuration

Configuration 类的作用是对Hibernate 进行配置，以及对它进行启动。在Hibernate 的启动过程中，

Configuration 类的实例首先定位映射文档的位置，读取这些配置，然后创建一个SessionFactory对象。

虽然Configuration 类在整个Hibernate 项目中只扮演着一个很小的角色，但它是启动hibernate 时所遇到的第一个对象。

目前最新版5.X了

5。

控制反转（Inversion of Control，英文缩写为IoC）是一个重要的面向对象编程的法则来削减计算机程序的耦合问题，

也是轻量级的Spring框架的核心。

控制反转一般分为两种类型，依赖注入（Dependency Injection，简称DI）和依赖查找（Dependency Lookup）。依赖注入应用比较广泛。

说白了就是这样了，将对象也作为参数也作为配置项写到xml文件中，传入到相关的接口中，这对于面向单元测试非常有好处，我们在php的API框架已经使用过了，

就是某个方法要使用的对象，全部采用参数传入的方式来实现，而不是在自己的方法体里边去new这个对象。这样来减少耦合。

其基本概念是：不创建对象，但是描述创建它们的方式。在代码中不直接与对象和服务连接，但在配置文件中描述哪一个组件需要哪一项服务。容器（在 Spring 框架中是 IOC 容器）负责将这些联系在一起。

在典型的 IOC 场景中，容器创建了所有对象，并设置必要的属性将它们连接在一起，决定什么时间调用方法。

IoC最大的好处是什么？

因为把对象生成放在了XML里定义，所以当我们需要换一个实现子类将会变成很简单（一般这样的对象都是实现于某种接口的），只要修改XML就可以了，

这样我们甚至可以实现对象的热插拔（有点像USB接口和SCSI硬盘了）。

IoC最大的缺点是什么？

（1）生成一个对象的步骤变复杂了（事实上操作上还是挺简单的），对于不习惯这种方式的人，会觉得有些别扭和不直观。

（2）对象生成因为是使用反射编程，在效率上有些损耗。但相对于IoC提高的维护性和灵活性来说，这点损耗是微不足道的，

除非某对象的生成对效率要求特别高。

（3）缺少IDE重构操作的支持，如果在Eclipse要对类改名，那么你还需要去XML文件里手工去改了，

这似乎是所有XML方式的缺陷所在。

6.

Spring是一个用户降低模块耦合度的框架，非常轻量

目前最新版4.X

登录站点，下载Spring的最新稳定版本。最新版本为spring-framework-4.0.建议下载spring-framework-spring-framework-4.0.0.M2-dist这个压缩包不仅包含Spring的开发包，

而且包含Spring编译和运行所依赖的第三方类库。

解压缩下载到的压缩包。

将spring.jar复制到项目的CLASSPATH路径下，对于Web应用，将spring.jar文件复制到WEB-INF/lib路径下，该应用即可以利用Spring框架了。

通常Spring的框架还依赖于其他一些jar文件，因此还须将lib下对应的包复制到WEB-INF/lib路径下，具体要复制哪些jar文件，取决于应用所需要使用的项目。

通常需要复制cglib，dom4j，jakarta-commons，log4j等文件夹下的jar文件。

为了编译java文件，可以找到Spring的基础类，将Spring.jar文件的路径添加到环境变量CLASSPATH中。当然，也可以使用ANT工具，但无须添加环境变量。

如果使用Eclipse或者NetBeans等IDE时，也不需要设置环境变。

传统的web架构的view 表现层使用struts作为表现层。但是如果试用下spring自带的MVC，会发现spring 在一般场合完全可以取代struts。

从某些角度来说，spring的mvc设计的更加合理，有兴趣的话不妨尝试下单个的spring的MVC。

例子：

我们在包Com.zhaosoft.action中新建一个LoginActionSupport.java，该类不继承于struts的Action，而是继承于Spring的ActionSupport，

这样，我们就是在使用spring框架了。

Spring 并不是Java所特有的，它也应用于.NET平台。而且得到了很好的发展！

Spring DOTNET 是一个关注于.NET企业应用开发的应用程序框架。

7.

jsp页面的常用指令：

page ,用来定义jsp文件中的全局属性，一般放在页面顶端，其包括很多属性：常用的有language,importerrorPage,contentType,pageEncoding

例如：<%@page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

include:用于在当前jsp文件中加载其他文件代码，例如：<%@ include file="main.jsp"%>

taglib: 用于指定页面使用的标签库以及其自定义标签的前缀，例如：<%@tagliburi="tagLibraryURI" prefix="tagPrefix" %>

8.

jsp常用动作：

.jsp:include：在页面被请求的时候引入一个文件。

　　 <jsp:include>标签表示包含一个静态的或者动态的文件。

语法：

<jsp:include page="path" flush="true" />或

<jsp:include page="path" flush="true">

<jsp:param name="paramName" value="paramValue" />

</jsp:include>

注：

1、page="path" 为相对路径，或其表达式。

2、flush="true" 必须设置flush为true，因为它默认值是false。

3、<jsp:param>子句能传递一个或多个参数给动态文件，也可在一个页面中使用多个<jsp:param>来传递多个参数给动态文件。

.jsp:useBean：寻找或者实例化一个JavaBean。

<jsp:useBean>标签表示用来在JSP页面中创建一个BEAN实例并指定它的名字以及作用范围,可以进行复用

语法：

<jsp:useBean id="name" scope="page | request | session | application" typeSpec />

其中typeSpec有以下几种可能的情况：

class="className" | class="className" type="typeName" | beanName="beanName" type="typeName" | type="typeName" |

注：

必须使用class或type，而不能同时使用class和beanName。beanName表示Bean的名字，其形式为“a.b.c”。

　　.jsp:setProperty：设置JavaBean的属性。

<jsp:setProperty>标签表示用来设置Bean中的属性值。

语法：

<jsp:setProperty name="beanName" prop\_expr />

其中prop\_expr有以下几种可能：

property="\*" | property="propertyName" | property="propertyName" param="parameterName" | property="propertyName" value="propertyValue"

注：

使用jsp:setProperty来为一个Bean的属性赋值；可以使用两种方式来实现。

1、在jsp:useBean后使用jsp:setProperty：

<jsp:useBean id="myUser" … />

…

<jsp:setProperty name="user" property="user" … />

在这种方式中，jsp:setProperty将被执行。

2、jsp:setProperty出现在jsp:useBean标签内：

<jsp:useBean id="myUser" … >

…

<jsp:setProperty name="user" property="user" … />

</jsp:useBean>

在这种方式中，jsp:setProperty只会在新的对象被实例化时才将被执行。

\* 在<jsp:setProperty>中的name值应当和<jsp:useBean>中的id值相同。

　　.jsp:getProperty：输出某个JavaBean的属性。

<jsp:getProperty>标签表示获取BEAN属性的值并将之转化为一个字符串，然后将其插入到输出的页面中。

语法：

<jsp:getProperty name="name" property="propertyName" />

注：

1、使用<jsp:getProperty>之前，必须用<jsp:useBean>来创建它。

2、不能使用<jsp:getProperty>来检索一个已经被索引了的属性。

3、能够和JavaBeans组件一起使用<jsp:getProperty>，但不能与Enterprise Java Bean一起使用。

　　.jsp:forward：把请求转到一个新的页面。

<jsp:forward>标签表示重定向一个静态html/jsp的文件，或者是一个程序段。

语法：

<jsp:forward page="path"} />

或

<jsp:forward page="path"} >

<jsp:param name="paramName" value="paramValue" />……

</jsp:forward>

注：

1、page="path" 为一表达式，或一个字符串。

2、<jsp:param> name 指定参数名，value指定参数值。参数被发送到一个动态文件，参数可以是一个或多个值，而这个文件却必须是动态文件。

要传递多个参数，则可在一个JSP文件中使用多个<jsp:param>将多个参数发送到一个动态文件

.jsp:plugin：根据浏览器类型为Java插件生成OBJECT或EMBED标记。

这个先不详细介绍了

9.

jsp的九大内置对象，都不需要创建这些对象，直接使用就可以了

1、request对象javax.servlet.http.HttpServletRequest

request对象代表了客户端的请求信息，主要用于接受通过HTTP协议传送到服务器的数据。（包括头信息、系统信息、请求方式以及请求参数等）。request对象的作用域为一次请求。

当Request对象获取客户提交的汉字字符时，会出现乱码问题，必须进行特殊处理。首先，将获取的字符串用ISO-8859－1进行编码，并将编码存发到一个字节数组中，然后再将这个数组转化为字符串对象如下

Request常用的方法：getParameter(String strTextName) 获取表单提交的信息.

getProtocol() 获取客户使用的协议。

String strProtocol=request.getProtocol();

getServletPath() 获取客户提交信息的页面。String strServlet=request.getServletPath();

getMethod() 获取客户提交信息的方式 String strMethod=request.getMethod();

getHeader() 获取HTTP头文件中的accept,accept-encoding和Host的值,String strHeader=request.getHeader();

getRermoteAddr() 获取客户的IP地址。String strIP=request.getRemoteAddr();

getRemoteHost() 获取客户机的名称。String clientName=request.getRemoteHost();

getServerName() 获取服务器名称。 String serverName=request.getServerName();

getServerPort() 获取服务器的端口号。intserverPort=request.getServerPort();

getParameterNames() 获取客户端提交的所有参数的名字。

2、response对象javax.servlet.http.HttpServletResponse

response 代表的是对客户端的响应，主要是将JSP容器处理过的对象传回到客户端。response对象也具有作用域，它只在JSP页面内有效。

具有动态响应contentType属性,当一个用户访问一个JSP页面时，如果该页面用page指令设置页面的contentType属性是text/html，那么JSP引擎将按照这个属性值做出反应。

如果要动态改变这换个属性值来响应客户，就需要使用Response对象的setContentType(String s)方法来改变contentType的属性值。

response.setContentType(String s); 参数s可取text/html,application/x-msexcel,application/msword等。

在某些情况下，当响应客户时，需要将客户重新引导至另一个页面，可以使用Response的sendRedirect(URL)方法实现客户的重定向。

例如response.sendRedirect(index.jsp);

3、session对象javax.servlet.http.HttpSession

Session对象是一个JSP内置对象，它在第一个JSP页面被装载时自动创建，完成会话期管理。从一个客户打开浏览器并连接到服务器开始，到客户关闭浏览器离开这个服务器结束，被称为一个会话。

当一个客户访问一个服务器时，可能会在这个服务器的几个页面之间切换，服务器应当通过某种办法知道这是一个客户，就需要Session对象。

当一个客户首次访问服务器上的一个JSP页面时，JSP引擎产生一个Session对象，同时分配一个String类型的ID号，JSP引擎同时将这换个ID号发送到客户端，存放在Cookie中，

这样Session对象，直到客户关闭浏览器后，服务器端该客户的Session对象才取消，并且和客户的会话对应关系消失。当客户重新打开浏览器再连接到该服务器时，服务器为该客户再创建一个新的Session对象。

session 对象是由服务器自动创建的与用户请求相关的对象。服务器为每个用户都生成一个session对象，用于保存该用户的信息，跟踪用户的操作状态。

session对象内部使用Map类来保存数据，因此保存数据的格式为“Key/value”。 session对象的value可以使复杂的对象类型，而不仅仅局限于字符串类型。

public String getId()：获取Session对象编号。

public void setAttribute(String key,Objectobj)：将参数Object指定的对象obj添加到Session对象中，并为添加的对象指定一个索引关键字。

public Object getAttribute(String key)：获取Session对象中含有关键字的对象。

public Boolean isNew()：判断是否是一个新的客户。

4、application对象javax.servlet.ServletContext

application 对象可将信息保存在服务器中，直到服务器关闭，否则application对象中保存的信息会在整个应用中都有效。与session对象相比，application对象生命周期更长，类似于系统的“全局变量”。

服务器启动后就产生了这个Application对象，当客户再所访问的网站的各个页面之间浏览时，这个Application对象都是同一个，直到服务器关闭。但是与Session对象不同的时，

所有客户的Application对象都时同一个，即所有客户共享这个内置的Application对象。

setAttribute(String key,Objectobj)：将参数Object指定的对象obj添加到Application对象中，并为添加的对象指定一个索引关键字。

getAttribute(String key)：获取Application对象中含有关键字的对象。

6、pageContext对象javax.servlet.jsp.PageContext

pageContext对象的作用是取得任何范围的参数，通过它可以获取 JSP页面的out、request、reponse、session、application 等对象。

pageContext对象的创建和初始化都是由容器来完成的，在JSP页面中可以直接使用pageContext对象。

page 对象代表JSP本身，只有在JSP页面内才是合法的。 page隐含对象本质上包含当前 Servlet接口引用的变量，类似于Java编程中的 this 指针。

7、config对象javax.servlet.ServletConfig

config对象的主要作用是取得服务器的配置信息。通过pageConext对象的getServletConfig() 方法可以获取一个config对象。当一个Servlet 初始化时，

容器把某些信息通过config对象传递给这个 Servlet。开发者可以在web.xml 文件中为应用程序环境中的Servlet程序和JSP页面提供初始化参数。

8 cookie 对象

Cookie是Web服务器保存在用户硬盘上的一段文本。Cookie允许一个Web站点在用户电脑上保存信息并且随后再取回它。举例来说，一个Web站点可能会为每一个访问者产生一个唯一的ID，

然后以Cookie文件的形式保存在每个用户的机器上。

创建一个Cookie对象调用Cookie对象的构造函数就可以创建Cookie对象。Cookie对象的构造函数有两个字符串参数：Cookie名字和Cookie值。

例如：Cookie c = new Cookie(username",john"); 将Cookie对象传送到客户端。

JSP中，如果要将封装好的Cookie对象传送到客户端，可使用Response对象的addCookie()方法。

例如：response.addCookie(c)，读取保存到客户端的Cookie。

使用Request对象的getCookie()方法，执行时将所有客户端传来的Cookie对象以数组的形式排列，如果要取出符合需要的Cookie对象，就需要循环比较数组内每个对象的关键字。

设置Cookie对象的有效时间，用Cookie对象的setMaxAge()方法便可以设置Cookie对象的有效时间，

例如：Cookie c = newCookie(username","john");c.setMaxAge(3600);

Cookie对象的典型应用时用来统计网站的访问人数。由于代理服务器、缓存等的使用，唯一能帮助网站精确统计来访人数的方法就是为每个访问者建立一个唯一ID。使用Cookie，网站可以完成以下工作:

测定多少人访问过。测定访问者有多少是新用户（即第一次来访），多少是老用户。

测定一个用户多久访问一次网站。当一个用户第一次访问时，网站在数据库中建立一个新的ID，并把ID通过Cookie传送给用户。用户再次来访时，网站把该用户ID对应的计数器加1，得到用户的来访次数。

9、exception 对象java.lang.Throwable

exception 对象的作用是显示异常信息，只有在包含isErrorPage="true" 的页面中才可以被使用，在一般的JSP页面中使用该对象将无法编译JSP文件。

excepation对象和Java的所有对象一样，都具有系统提供的继承结构。

exception 对象几乎定义了所有异常情况。在Java程序中，可以使用try/catch关键字来处理异常情况；如果在JSP页面中出现没有捕获到的异常，就会生成 exception 对象，

并把 exception 对象传送到在page指令中设定的错误页面中，然后在错误页面中处理相应的 exception 对象。

10.

Javabean:是一种JAVA语言写成的可重用组件。为写成JavaBean，类必须是具体的和公共的，并且具有无参数的构造器。

Javabean是一种特殊的java类。

用户可以使用JavaBean将功能、处理、值、数据库访问和其他任何可以用java代码创造的对象进行打包，

并且其他的开发者可以通过内部的JSP页面、Servlet、其他JavaBean、applet程序或者应用来使用这些对象。

用户可以认为JavaBean提供了一种随时随地的复制和粘贴的功能，而不用关心任何改变。

JavaBean可分为两种：一种是有用户界面（UI，User Interface）的JavaBean；

还有一种是没有用户界面，主要负责处理事务（如数据运算，操纵数据库）的JavaBean。JSP通常访问的是后一种JavaBean。

一个JavaBean和一个JavaApplet相似，是一个非常简单的遵循某种严格协议的Java类。每个JavaBean的功能都可能不一样，但它们都必须支持以下特征。

一个bean没有必须继承的特定的基类或接口。可视化的bean必须继承的类是java.awt.Component，这样它们才能添加到可视化容器中去，

非可视化bean则不需要继承这个类。有许多bean，无论是在应用程序构造器工具中，还是在最后创建好的应用程序中，都具有很强的可视化特征，

但这并非每个bean必须的特征。

在使用Java编程时，并不是所有软件模块都需要转换成bean。

其定义必须遵循以下规范：

1．如果类的成员变量的名字是xxx，那么为了更改或获取成员变量的值，即更改或获取属性，在类中可以使用两个方法：

getXxx()，用来获取属性xxx。setXxx()，用来修改属性xxx.。

2．对于boolean类型的成员变量，即布尔逻辑类型的属性，允许使用"is"代替上面的"get"。

3．类中访问属性的方法都必须是public的，一般属性是private的。

4．类中如果有构造方法，那么这个构造方法也是public的并且是无参数的。

由此可见，JavaBean确实也是普通的Java对象，只不过它遵循了一些特别的约定而已。

一个JavaBean由3部分组成：

　　（1）属性（properties）

（2）方法（method）

（3）事件（event）

11.

实际在jsp中应用javabean，可以用两种方式：

一是直接写java脚本，一是使用我们上面介绍的jsp动作useBean.例如：<jsp:useBean id="login" class="com.crgjava.bean.Login" />

<jsp:setProperty name="login" property="userName" value="陈荣光" />

<p>用户名：<jsp:getProperty name="login" property="userName" /></p>

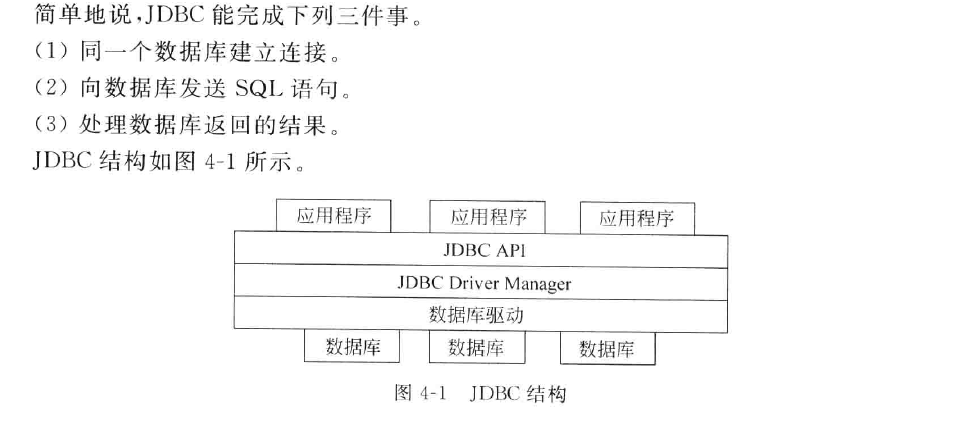
而在表单（form）提交时，如果表单参数的名字与javabean属性的名字相同，jsp引擎会自动将字符串转换为Javabean属性的类型，写法会变成，

<jsp:setProperty name="login" property="\*" />就可以了。

12.

Jdbc

即：Java database connectivity :java数据库连接，是一组程序应用接口（API），调用该接口来操作数据库，可以不用管我们连接的是什么类型的数据库（也就是不用为mysql或者oracle来单独编写代码。。。。。）



上图中，例如我们需要连接mysql数据库，那么我们去下载java对于mysql的驱动，引入程序即可（也就是我们上次下载的mysql-connector-java-5.1.40-bin.jar）

另外需要了解的知识是：

对于微软系的数据库或者源（例如sqserver ,access,excel等）还可以通过另一种方式，就是桥接的方式，sun公司和微软公司，联合编写了一个桥接驱动jdbc-odbc，利用该驱动，也可以连接微软系的数据源和数据库，这个需要安装和配置odbc（在控制面板-管理工具中可以找到odbc，并进行配置）了。

可以使用该语句：

来加载该桥接驱动。

实际上每个数据库厂商都为java提供了相依的jdbc驱动，所有我们最好使用jdbc的方式，不要使用桥接的方式来访问各种数据库。

例如，我们现在要用jdbc的方式来连接mysql数据库，

步骤：1.下载我们上面提到的mysql驱动，

然后将该驱动的jar包文件，放到项目的web-inf/lib下面

之后再jsp页面，第一行引入，<%@page import=”java.sql.\*”%>

后面使用以下写法，就可以操作数据库了：



Jdbc提供3种接口实现sql语句,分别是Statement,PreparedStatement和CallableStatement.

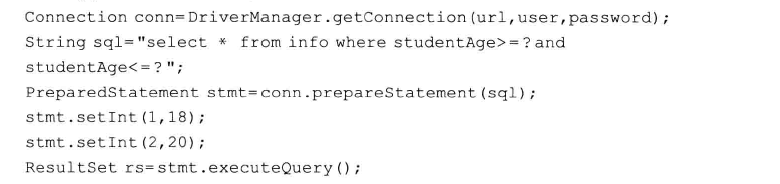
其中：

Statement:用于执行简单的sql语句。

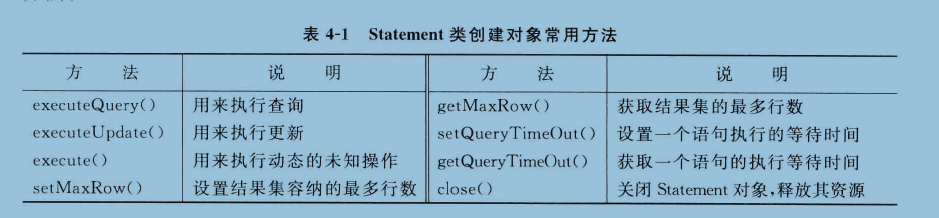
PreparedStatement：用于执行带有IN类型参数的预编译过的sql语句。其继承自：Statement

CallableStatement：用于执行一个数据库的存储过程。其继承自：PreparedStatement

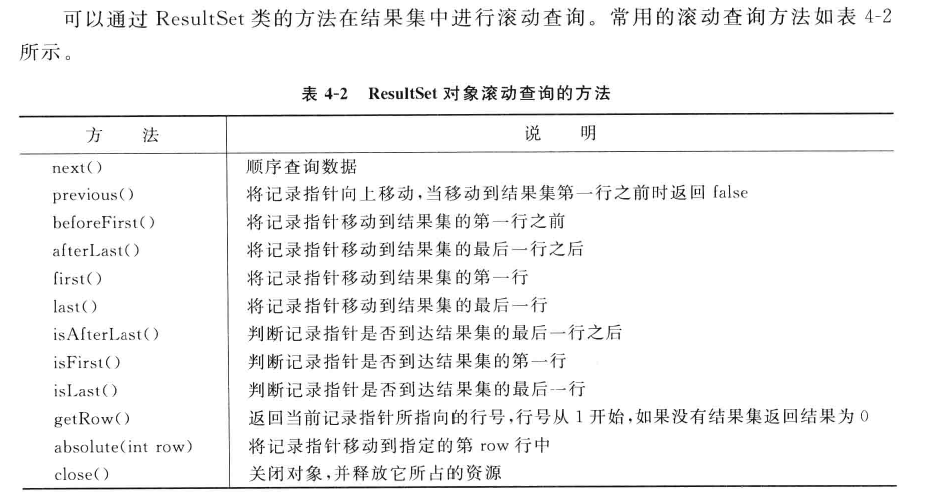
关于PreparedStatement的写法例子如下：



我们来看看Statement有什么方法；



我们再来看看返回的ResultSet类有什么方法：



13.

Servlet是一种在服务器端运行的java程序，而jsp是继servlet后，sun公司推出的新技术，它是以servlet为基础开发的。

而实际上在运行的时候，每一个jsp文件都会被jsp引擎翻译成为一个servlet文件（即.java文件），然后再编译成.class文件。

也就是说，jsp是在servlet的基础上又包装了一层。

看jsp文件和看jsp转换之后的servlet文件，我们可以发现里边的很大区别就是：

Jsp文件是在html里边写java代码，而servlet文件是在java里边输出html.

显然，sun公司为了前后端的分离，推出了jsp,这样，可以让jsp看起来更像是前端，毕竟里边大部分是html的布局等，只是嵌入了一些java代码。

总结下就是：我们完全可以不用jsp，而直接编写servlet文件，这样少一层转换，效率上也会快一些，但是那样就是前后端分离不是很彻底，如果有前端人员参与的话，项目很难进展。

所以，现在大部分都是用了编写jsp的方式，而不用在后端直接写servlet的方式。

（以上这些说明是想说明jsp的运行原理，由于servlet本身就是一个java类，开发者可以手动编写，还能实现其他很多功能，那些不是自动翻译成的sevlet就是另外一回事了，和这里讨论的不是一个概念。）

而在实际的项目中呢，我们会采用jsp+自己编写servlet(用于一些如果页面内容简单的页面)，结合完成项目。（例如，我们可以用jsp页面提交数据，提交到一个servlet中（这个时候就需要配置servlet了，配置之后，jsp页面form中的action就设置其提交到servlet了，），然后由servlet对从jsp页面提交过来的数据进行处理，然后输出out一个相应页面返回给客户端）

使用servlet的方式来开发是需要配置的，其配置文件就是在web-inf下面的web.xml文件。

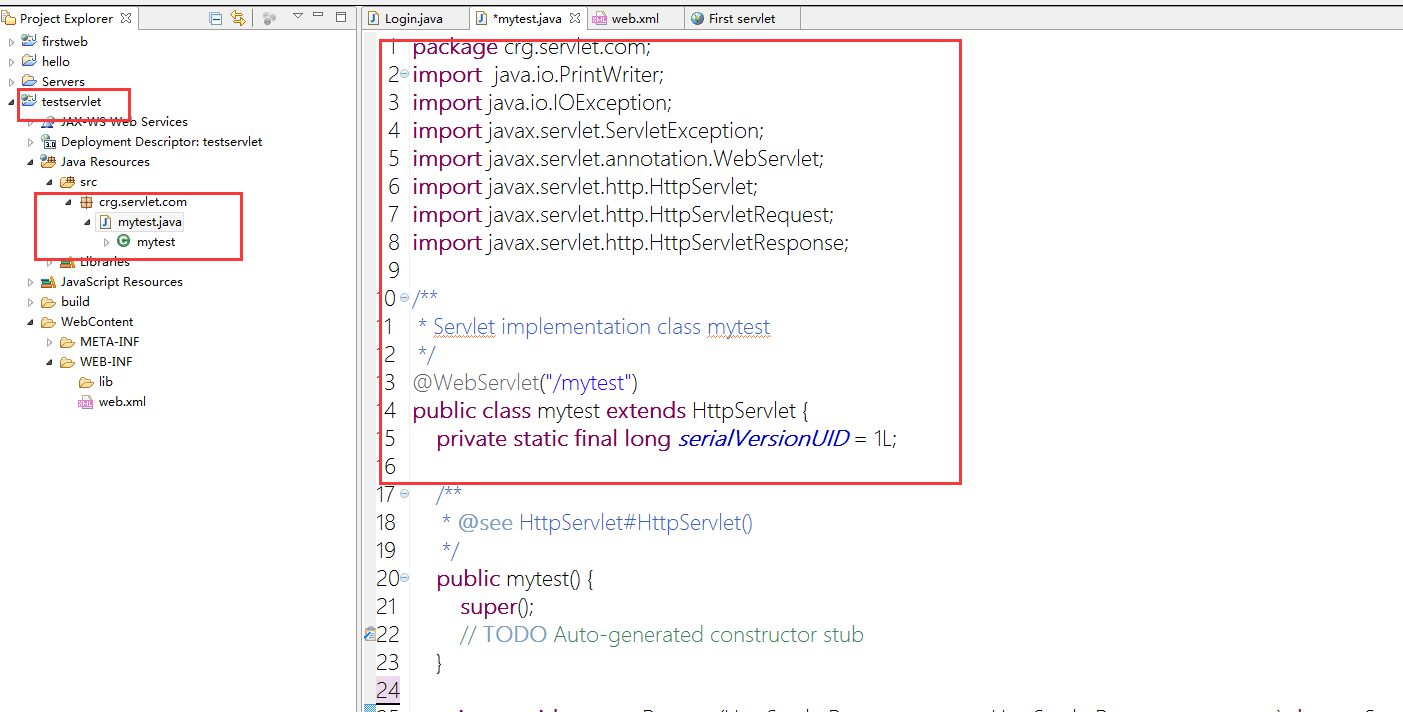
配置的时候看以下注释：



如何新建呢：

我们可以新建一个动态web项目

然后新建一个包，然后再包里边新建，选择新建servlet,然后下一步，之后呢会，动态新建一个类，该类已经动态生成一些代码了，最主要就是doget和dopost方法，之后呢，我们可以编写一个方法吧，具体如下：





然后我们选中mytest.java，右键，runas运行，可以看到浏览器上有结果输出了。

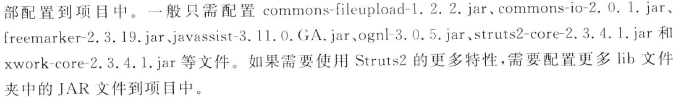
14.struts2

如何使用struts2框架呢？

我们可以先去下载框架,

<http://struts.apache.org/download.cgi>到这里去下载我们需要的版本。

解压之后，可以发现：apps文件夹里边是例子，doc里边是文档，src是源码，而lib里边就是该项目的核心类库以及它用到的第三方插件类库，发现这里边有非常多的jar包，但是实际项目一般项目只用到以下几个：但是需要注意，不同版本的struts其，需要的基本jar包是不同的，一定要查看文档，出了问题一定要查看tomcat日志，进行排查分析问题，看看少了什么jar包。



我们建立，动态web项目之后，将以上的包拷贝到项目的lib下，然后配置web.xml文件。

需要特别注意的是：不同版本的struts，他们在web.xml的配置上是不一样的，一定要查看文档。

其中web.xml增加以下配置过滤器：

<filter>

<filter-name>struts2</filter-name>

<filter-class>org.apache.struts2.dispatcher.filter.StrutsPrepareAndExecuteFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>struts2</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

配置过滤器是Struts2能跑起来最基本的前提，工作原理就是当tomcat容器启动时，首先加载web.xml，所以会加载其中定义的过滤器，因此StrutsPrepareAndExecuteFilter过滤器会过滤项目下所有的请求，然后后来利用过滤器捕获action请求，从而调用相应的方法进行处理，这就是Struts2的大致工作情况，StrutsPrepareAndExecuteFilter类出现可以让开发人员自定义过滤器，从而在请求到达处理之间对用户请求进行处理

配置好web.xml后需要建立Struts2的核心配置文件，位置就是项目下的src目录中，建立struts.xml文件，他负责配置所有的请求对应各种类或者方法，所以是核心配置文件，基本的结构如下：

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE struts PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN" "http://struts.apache.org/dtds/struts-2.3.dtd">

<struts>

<package name="zzf" extends="struts-default">

<action name="hellostruts2" class="crg.first.action.HelloStruts2Action">

<result>/result.jsp</result>

</action>

<action name="login" method="login" class="crg.first.action.HelloStruts2Action">

<result>/login.jsp</result>

</action>

</package>

</struts>

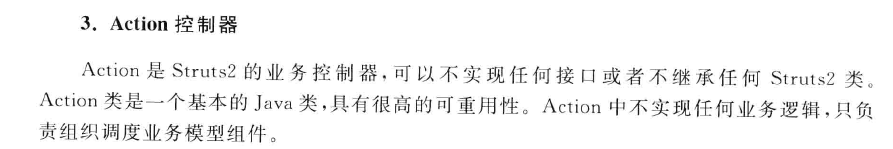
注意<struts>节点里边,

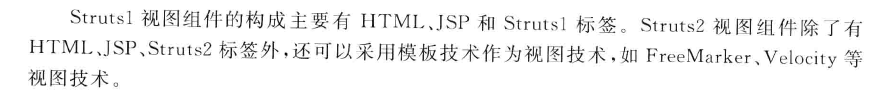
一个action就负责处理一个请求，packge我们使用的默认包，当有多个packge时，name一定要区分开，namespace就是访问的空间，　　第一个action对应的访问路径就是http://localhost:8080/study-struts/HelloStruts2/hellostruts2.action当然默认也可以不加action

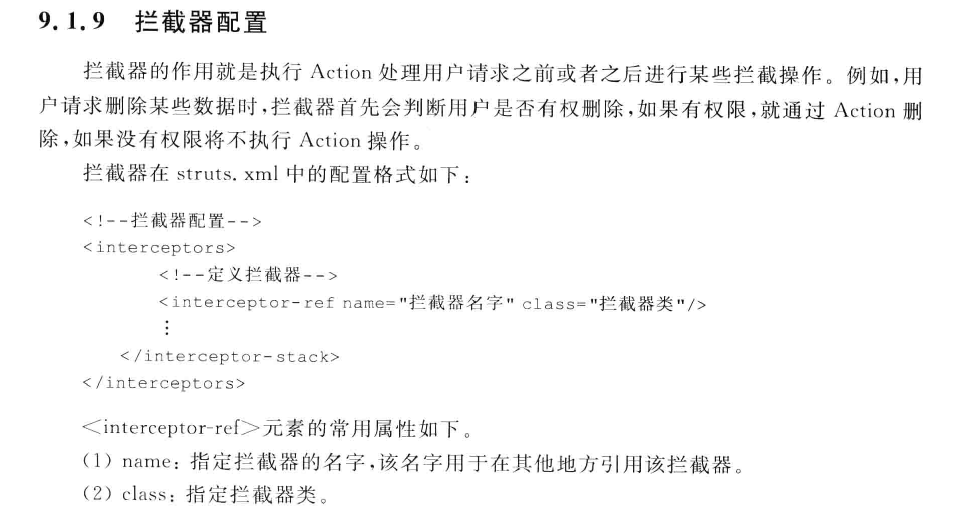
　　第二个action对应的就是http://localhost:8080/study-struts/HelloStruts2/login.action然后去建立一个Action类，就是HelloStruts2Action，类名命名必须是以Action结尾的驼峰命名，并且继承于ActionSupport类，

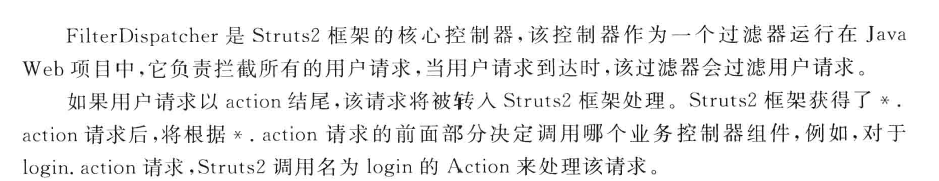
　如果前面配置文件没指定method属性，默认访问的方法就是execute方法，我们另外还自定义了一个login方法，SUCCESS常量返回了默认的<result>指定的视图，下面建立对应的jsp文件.在WebContent目录下面，建立result.jsp和login.jsp,里边随便写点什么html内容吧。

也就是说：



而对于前端的视图方面，有以下技术：提供了很多模板引擎可以使用：

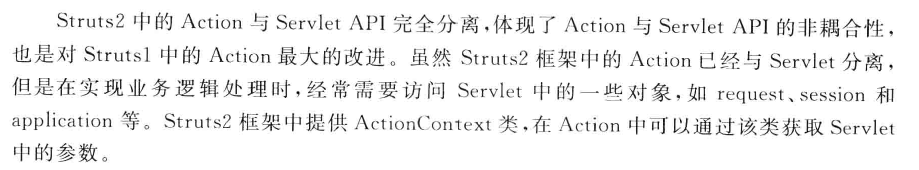




注意文中提到的FilterDispatcher是旧版本struts中的概念，新版本都用我们实际代码中的：StrutsPrepareAndExecuteFilter 替代了。

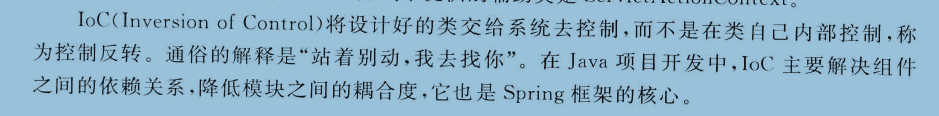


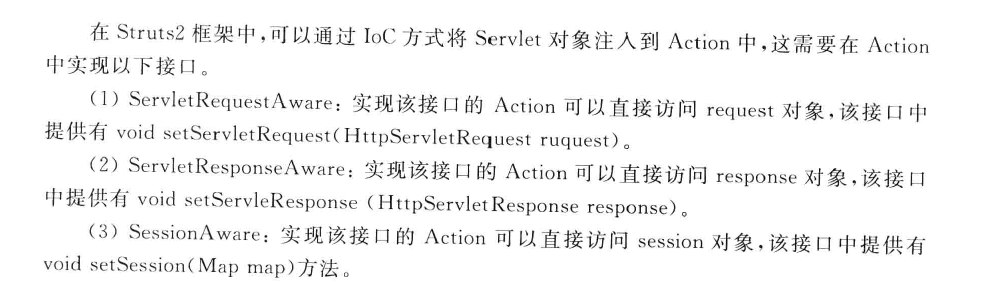




另外，还有一个也可以访问ServletActionContext,该类继承自ActionContext类，功能类似，这样都是直接的方式，也叫：非IOC方式。

还有一种使用IOC方式来访问Servlet辅助类的方式。







15.hibernate

可以到[www.hibernate.org](http://www.hibernate.org)去下载最新的版本。

