SSH 三大框架复习笔记

By fydstudio@live.com

第一部分：Struts 复习笔记

Struts：基于 MVC 设计模式的 Java Web FrameWork，提高 Java Web 开发速度

（Model：实现业务逻辑的 JavaBean 或 EJB 组件；View：一组 JSP 文件组成；Control：ActionServlet 和 Servlet）

Struts **组件：**

1、ActionServlet extends HttpServlet，接收请求信息，根据配置文件 struts-config.xml 中<action>元素 path 属性的值将请求交给对应的 action 对象处理（ActionServlet 创建 action 对象）

2、Action 命名对象 ,接收 ActionServlet 请求 ,调用 JavaBean 对请求做具体处理并将响应信息返回给 ActionServlet

3、ActionForm 对表单数据自动封装、验证、重置

4、Struts-config.xml 配置文件，对 ActionForm，数据源，Action 以及跳转路径的配置；网站国际化信息资源的配置；框架、插件的配置

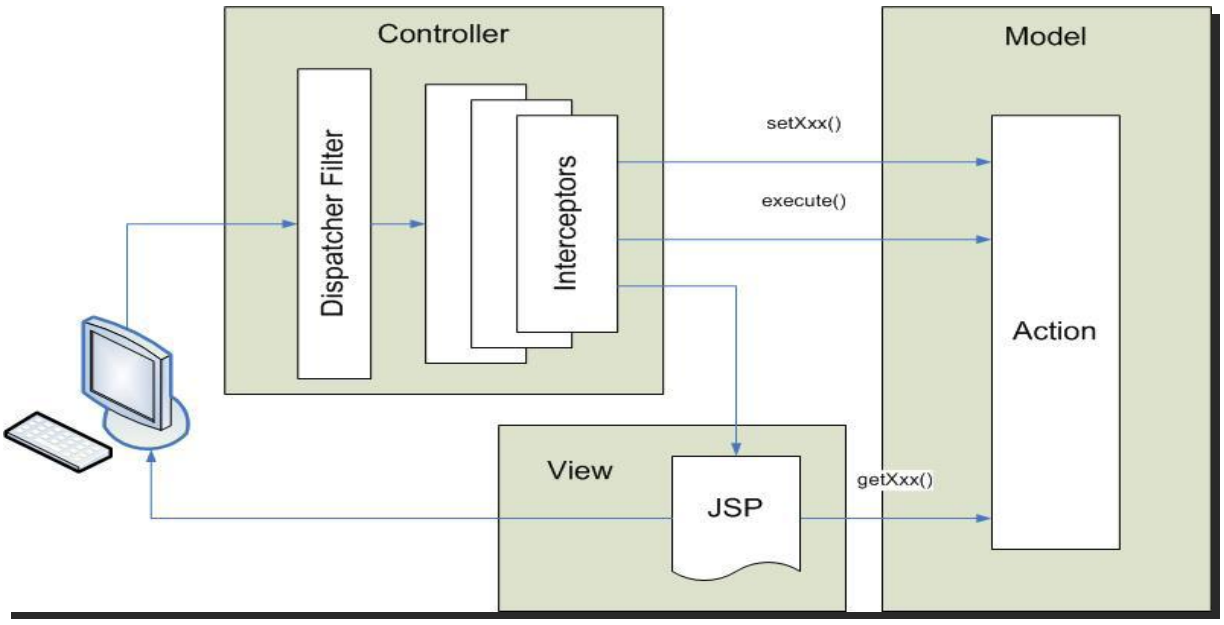
Struts **工作流程：**

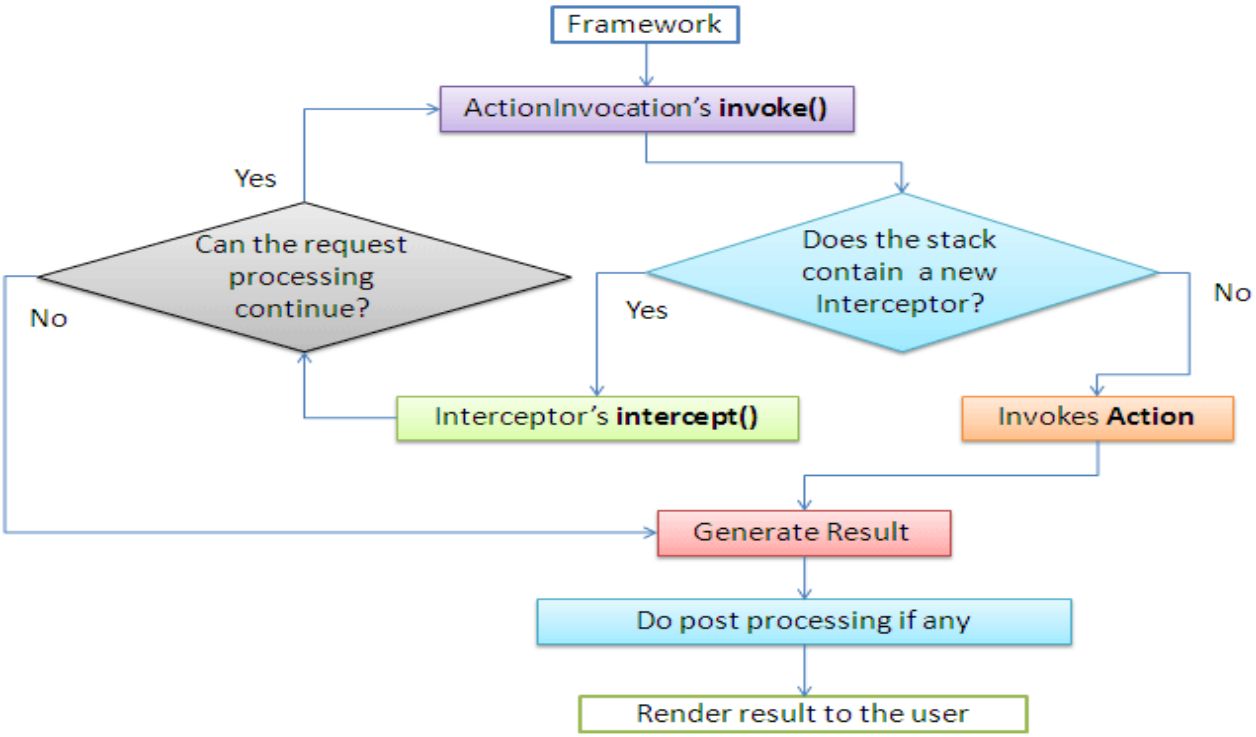
1、当 web server 启动时，实例化 ActionServlet，调用其 init 方法，server 通过解析 struts-config.xml 文件获取配置信息（SAX 解析），将配置信息存放在各种配置对象中

2、Client 发送一个*.Do 的请求时，ActionServlet 查找 ActionMapping 配置信息<action>元素 path 属性值，判断是否存在该路径请求，if false：404 error

3、If TRUE：判断<action>元素是否有 name 属性，if false：创建 Action，调用 execute 方法响应 Client 请求

4、If TRUE：创建对应的 ActionForm 对象（ActionForm 对象仅默认创建一次），通过反射机制自动封装数据，创建 Action，调用 execute 方法响应 Client 请求





一个请求在 Struts2 框架中的处理大概分为以下几个步骤

1 客户端初始化一个指向 Servlet 容器（例如 Tomcat）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 来决定这个请求是否需要调用某个 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。返回结果通常是（但不总是，也可 能是另外的一个 Action 链）一个需要被表示的 JSP 或者 FreeMarker 的模版。在表示的过程中可以使用 Struts2 框架中继承的标签。在这个过程中需要涉及到 ActionMapper

Struts **的配置**

（1）. web.xml 文件

主要完成对 StrutsPrepareAndExecuteFilter 的配置，它的实质是一个过滤器，它负责初始化整个 Struts 框架并且处理所有的请求。

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

（2）. struts.xml 文件

框架的核心配置文件就是这个默认的 struts.xml 文件。

Result **的默认类型是** dispatcher 用来转向页面，通常处理 JSP。

```
<package name="tt1" namespace="/test" extends="struts-default">
        <action name="test" class="com.asm.Test1Action">
            <result name="success">/forward/test.jsp</result>
        </action>
</package>
```

（3）. Action 的配置

一个 Action 可以被多次映射(只要 action 配置中的 name 不同)

name：action 名称

class: 对应的类的路径

method: 调用 Action 中的方法名

如果不配置，默认执行 execute **方法**

（4）. struts-default.xml

此文件是 struts2 框架默认加载的配置文件，它定义了 struts2 一些核心 bean 和拦截器，它会自动包含(included)到 struts.xml 文件中(实质是通过<package extends="struts-default">)，并为我们提供了一些标准的配置。

（5）. 其它配置文件

struts.properties struts-plugin.xml

关于 namespace

链接地址

```
<a href="<%=path%>/test/test.action">test1</a>
```

struts.xml 中的内容：

```
<package name="tt1" namespace="/test" extends="struts-default">
        <action name="test" class="com.asm.Test1Action">
            <result name="success">/forward/test1.jsp</result>
        </action>
</package>
```

四个隐藏对象：request、session、application、response

Struts **中提交数据的两种方式：**

方法一：URL 参数

方法二：Form 表单

Action **类中获得隐藏对象的方法：**

方法一：非 IoC（控制反转 Inversion of Control）方式

```
HttpServletRequest request = ServletActionContext.getRequest();
HttpServletResponse response = ServletActionContext.getResponse();
ServletContext application = ServletActionContext.getServletContext();
HttpSession session = request.getSession();
```

方法二： IoC 方式，Action 类实现对应的接口，如当需要引入 Request 对象时，方法如下

```
public class MemberAction extends ActionSupport   implements ServletRequestAware{
        HttpServletRequest   request;
        ...
        public void setServletRequest(HttpServletRequest arg0) {
        this.request=arg0;
        }
        ...
}
```

其他接口：

1. ServletRequestAware
2. ServletResponseAware

接口方法：

setServletResponse(HttpServletResponse response)

1. SessionAware

接口方法：

setSession(Map session)

- 这种方法获得的是 session 的属性映射(setAttribute)

Struts **提交验证的两种方法：**

方法一：利用 JavaScript 进行验证

方法二：重载 ActionSupport 中的方法 validate

如何自定义一个拦截器？

第一步：自定义一个实现 Interceptor 接口（或者继承自 AbstractInterceptor）的类。

第二步：在 strutx.xml 中注册上一步中定义的拦截器。

第三步：在需要使用的 Action 中引用上述定义的拦截器，为了方便也可将拦截器定义为默认的拦截器，这样在不加特殊声明的情况下所有的 Action 都被这个拦截器拦截。

拦截器和过滤器的区别

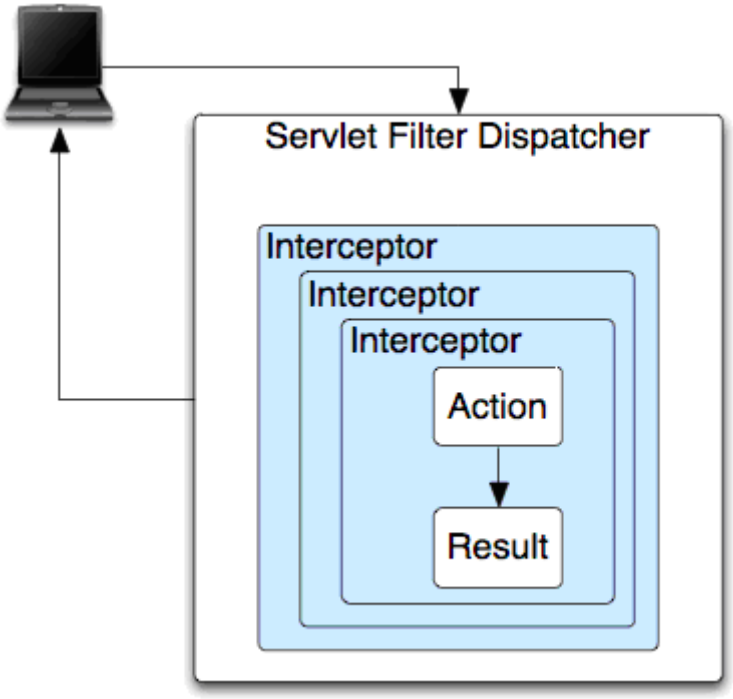
拦截器是基于 java 的反射机制的，而过滤器是基于函数回调

过滤器依赖于 servlet 容器，而拦截器不依赖于 servlet 容器

拦截器只能对 action 请求起作用，而过滤器则可以对几乎所有的请求起作用

器可以访问 action 上下文、值栈里的对象，而过滤器不能，过滤器主要对 request 和 response 进行处理

在 action 的生命周期中，拦截器可以多次被调用，而过滤器只能在容器初始化时被调用一次



使用拦截器的优点

高灵活性

Action 类的目标更加专注.

保证了代码的可读性和重用性.

测试过程更加容易

MVC **设计模式浅析：**

强制将 application 的输入，处理和输出分开，分为 Model-View-Control

Model：业务数据和业务逻辑，模型能为多个视图提供数据，提高应用可重用性

View：视图是与用户交互的界面，显示相关数据并接收用户的输入数据，视图不进行任何实际的业务处理，查询模型的业务状态，但是不能更改模型，接收模型处理的信息，动态显示结果

Control：接收用户请求，并调用对应的模型和视图响应请求

优点：1、多个视图能共享一个模型，提高应用的可重用性

2、模型是自包含的，和视图、控制器保持相对独立，可以方便地改变数据层和业务逻辑实现，构成良好的松耦合系统

3、控制器提高了 application 的可配置型和灵活性

第二部分：Hibernate **复习笔记**

Hibernate**：**解决数据库平台的移植，对 JDBC API 作轻量级的封装，ORM 思想（对象----关系型映射）

什么是持久化：Persistence ,就是把内存中的数据保存到磁盘上供以后所用 ,持久化的实现过程大多通过数据库来完成。

对象关系映射（Object Relational Mapping，简称 ORM**）**

ORM 是一种为了解决面向对象与关系数据库存在的互不匹配的现象的技术。简单的说，ORM 是通过使用描述对象和数据库之间映射的元数据，将 java 程序中的对象自动持久化到关系数据库中。本质上就是将数据从一种形式转换

到另外一种形式。

以面向对象的形式完成对数据库的操作：

1、对象和数据库表的映射关系；

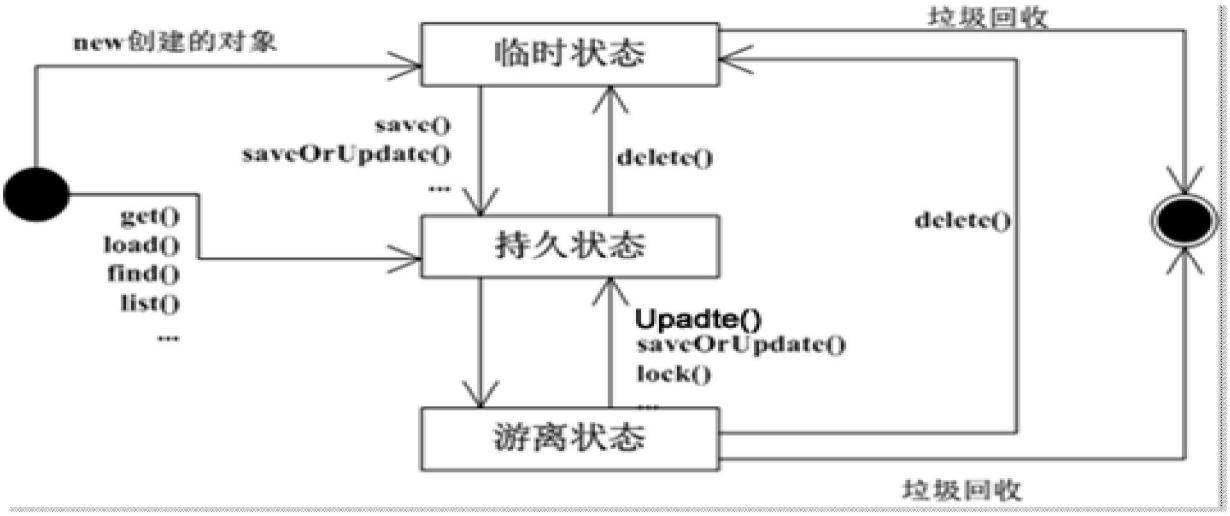
2、对象的属性和表的字段需要建立映射关系；

3、对象之间对应数据库表之间的映射关系

hibernate.cfg.Xml：代表整个 application 全局配置，<session-factory>对应一个数据库连接信息的配置；配置文件中包含了通过 Dialect 接口实现数据库方言机制；

Pojo：持久化的一个对象，类似于 java Vo 对象；pojo 类必须实现 Serializable 接口，否则序列化异常；需要提供默认的无参构造函数；

持久化状态的变化：临时状态，持久化状态，游离状态；



SessionFactory：重量级缓存（二级缓存），包含了 hibernate 所有映射文件的源程序；对于数据库表的基本操作语句进行预定义；充当 application 和 DB 之间的代理，一个数据源仅对应一个 sessionfactory，支持并发访问，支持线程安全；

Session：轻量级缓存（一级缓存）； 缓存了当次操作的 pojo 对象； 减少运行时间，提高性能；缓存 hibernate 生成的 sql 语句；非线程安全；session 不能共享，每次操作需要开启关闭 session；

监视 pojo 对象的状态，当 pojo 对象数据发生变化，hibernate 自动更新数据库保证 pojo 和 DB 同步；

Session **缓存的三大作用：**

1，减少数据库的访问频率，提高访问性能。

2，保证缓存中的对象与数据库同步，位于缓存中的对象称为持久化对象。

3，当持久化对象之间存在关联时，Session 保证不出现对象图的死锁。

Session **会话**

```
Session session = sessionFactory.openSession();
Session session = sessionFactory.getCurrentSession();
session.beginTransaction();
//数据库操作
session.getTransaction().commit();
```

Hibernate **配置步骤**

第一步：创建一个实体类，包含：对应的实体属性、默认的构造函数（无参数的构造函数）、每个属性的 getter 和 setter 方法

第二步：创建映射文件(Student.hbm.xml)　**！注意一下加粗字体**

```
<hibernate-mapping package="com.fyd.entity">
    <class name="Student" table="stu_info" lazy="false"> <--! Lazy 延迟加载 -->
        <id name="stu_id" column="stu_id">    <--! type 可以不写，系统会自动识别 -->
            <generator class="assigned"/>    <--! generator 主键生成策略 -->
        </id>        <--! Id 用来定义主键 只能有一个<id></id> -->
        <property  name="stu_name"  column="stu_name" />
        <property  name="stu_address"  column="stu_ address " />
        <many-to-one name="classes" column="classNo" lazy="false" not-null="true"/>
        <--! 如果需要可以添加 many-to-one 或 one-to-many 等 -->
</class>
</hibernate-mapping>
```

第三步：定义 Hibernate 配置文件

位置：在项目的 src 目录下，创建一个应用 Hibernate 的配置文件，Hibernate.cfg.xml

映射文件的加入<mapping resource="com/hibernate/Student.hbm.xml" />

最后，测试这个 Hibernate 应用

装载对象：

load( ) 方法

Object　load（Class entityClass,Serializable id）

用于从数据库加载指定的持久化对象到 Session 缓存中，**如果指定的记录不存在则抛出异常**，这与 get（ ）方法不同。

get（ ）方法

Object　get（Class entityClass,Serializable id）

从数据库加载指定的持久化对象到 Session 缓存中，如果指定的记录不存在则返回 null。

第三部分：Spring **复习笔记**

Spring：是一个开源的控制反转(Inversion of Control ,IoC)和面向切面(AOP)的轻量级容器框架，它的主要目得是简化企业开发。

控制反转 IOC**(** Inverse of Control **):**Spring 核心概念 ;实现了对象之间关系的反转 ,对象之间的直接依赖关系由 IOC 容器控制；IOC 为设计模式上的概念，分为 Dependency Injection（依赖注入）和 Service Locator（服务定位器）

简单的讲：由容器来管理对象之间的依赖关系（而不是对象本身来管理），就叫“控制反转”或“依赖注入”

面向切面编程 AOP **(** Aspect oriented programing **):** Spring 的核心特色；切面：代理对象为真实对象提供的某个服务于业务核心的领域服务；

事务：原子性（Atomic），一致性（Consistency），隔离性（Isolation），持久性（Durability）

Struts **和** Spring **的组成 步骤：**

第一步 加载 Jar 包（struts2-spring-plugin-2.1.8.jar、spring.jar、commons-logging.jar）

第二步 配置 Spring 的监听器（在 web.xml 中）,负责 启动 Spring，开始工作

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

或者配置 Servlet（两种方法任选一种）

```
<servlet>
    <servlet-name>context</servlet-name>
    <servlet-class>org.springframework.web.context.ContextLoaderServlet
        </servlet-class>
</servlet>
```

第三步：依照 Spring 的规则，配置相关的 bean 定义。

例如：在 RegisterBean.xml 中添加 RegisterAction 类的 bean 定义

```
<bean id="registerService" class="com.ssh.labwork1.RegisterServiceImpl" >
        <property name="rdao"><ref bean="registerDAO"/></property>
</bean>
<bean id="registerDAO" class="com.ssh.labwork1.RegisterDAOImpl" />
<bean id="registerActionBean"   //这里的名字在 Struts2 的 Action 定义中要用
class="com.ssh.labwork1.RegisterAction" scope="prototype">
        <property name="rs"><ref bean="registerService"/></property>
</bean>
```

第四步：修改原来的 Struts2 中关于 Action 的模块配置文件，将 Action 中的 class 属性的值保持和 Spring 中的定义的 bean 的 id 属性的值保持一致

```
<?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-2.0.dtd">
<struts>
 <package name="register" namespace="/register" extends="struts-default">
---------- <!-- 以下是 Struts2 独立的配置 -->
<!--
 <action name="register_*" method="{1}" class="com.struts2.labwork3.RegisterAction">
   <result name="success">/labwork3/register_detail_xhtml.jsp</result>
 </action>
-->
---------- <!-- 以下是 Struts2 集成 Spring 的配置 -->
<action name="register_*" method="{1}" class="registerActionBean">
    <result name="input">/sshLabwork1/register_xhtml.jsp</result>
    <result name="success">/sshLabwork1/register_detail_xhtml.jsp</result>
 </action>
</package>
</struts>
```

第五步：添加一个默认 Spring 加载文件 ApplicationContext.xml,此文件的默认位置在 WEB-INF 目录下

```
<context-param>   //当该文件不在默认位置时，在 web.xml 中设置一个如下参数
    <param-name>contextConfigLocation</param-name>
    <param-value>
     /WEB-INF/classes/spring/applicationContext.xml
    </param-value>
</context-param>
```

第四部分：SSH 框架相关内容

log4j 在 Web 开发中的应用

log4j 是一个能够帮助程序员把信息输出到不同媒介的工具。它可以将输出的信息分成不同的级别，可以自定义格式化输出的信息。

Log4j 由三个重要的组件构成：

Logger -日志信息的记录器

Appender -日志信息的输出目的地

Layout -日志信息的输出格式。

Log4j 一共存在 6 个基本的日志级别，从低到高：TRACE、DEBUG、INFO、WARN、ERROR 以及 FATAL。

Log4j 配置步骤：

第一步：所需的 jar 文件

引入 log4j 相应版本的 jar 文件及其关联文件，例如 commons-logging.jar 和 log4j-1.2.15.jar

第二步：编写一个配置文件

位置：放在 src 下，部署时在 classes 目录下

文件名：

基于 xml 的 log4j.xml

基于属性文件的 log4j.properties

第三步：在代码中输出信息

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">
<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/" debug="false">
<appender name="CONSOLE" class="org.apache.log4j.ConsoleAppender">
        <errorHandler class="org.jboss.logging.util.OnlyOnceErrorHandler"/>
        <param name="Target" value="System.out"/>
        <param name="Threshold" value="DEBUG"/>
        <layout class="org.apache.log4j.PatternLayout">
            <!-- The default pattern: Date Priority [Category] Message\n -->
            <param name="ConversionPattern" value="%d{ABSOLUTE} %-5p [%c{1}] %m%n"/>
        </layout>
</appender>
<category name="com.student.web">
        <priority value="WARN"/>
         <appender-ref ref="CONSOLE"/>
    </category>
</log4j:configuration>
```

```
Logger   logger = Logger.getLogger(this.getClass().getName());
```

```
logger.warn("----helloo json");
```

Ajax（Asynchronous JavaScript and XML）**在** Web **开发中的应用**

不是一个新的技术，事实上，它是一些旧有成熟的技术以一种全新更加强大的方式整合在一起。

Ajax **的关键技术：**

使用 XHTML(HTML)和 CSS 构建标准化的展示层

使用 DOM 进行动态显示和交互

使用 XML 和 XSLT 进行数据交换和操纵

使用 XMLHttpRequest 异步获取数据

使用 JavaScript 将所有元素绑定在一起

Ajax **应用程序的优势在于：**

1. 通过异步模式（**异步发送请求**），提升了用户体验
2. 优化了浏览器和服务器之间的传输，减少不必要的数据往返，减少了带宽占用
3. Ajax 引擎在客户端运行，承担了一部分本来由服务器承担的工作，从而减少了大用户量下的服务器负载。

XMLHttpRequest **对象的方法属性**（XMLHttpRequest 对象是 Ajax **的核心**）

方法：

open()：建立到服务器的新请求

send()：向服务器发送请求

onreadystatechange：指定回调函数

readyState：提供当前 HTML 的就绪状态

status：服务器响应的状态代码

responseText：服务器返回的请求响应文本

属性	说明
readyState	对象状态 0=未初始化； 1=读取中 2=已读取；3=交互中； 4=完成
responeText	服务器返回内容的文本
responeXML	服务器返回的 XML 文档对象
status	服务器返回的状态码，如 404，200 等 200：正常；404：没有找到文件或目录；500：服务器内部错误
statusText	服务器返回的状态码的文本信息