JSF请求处理过程（二）请求处理过程总览Java认证考试 PDF转换可能丢失图片或格式，建议阅读原文https://www.100test.com/kao_ti2020/644/2021_2022_JSF_E8_AF_B7_E6_B1_82_E5_c104_644751.htm 请求处理过程总览（FacesServlet#service）这总览，很明显是看FacesServlet的service方法。在FacesServlet的初始化过程中，构造出了全局的 FacesContextFactory对象和LifeCycle对象。可以把FacesContextFactory看做是一个“ 请求包装工厂 ”，于是很明显，每当一个请求到达FacesServlet的时候，第一步便是拿着请求，到包装工厂里面包装一下，而包装的结果就是一个FacesContext.代码如下： 1 FacesContext context = facesContextFactory.getFacesContext（servletConfig.getServletContext（）， request， response，lifecycle）；在包装过程中，实际上是创建了一个com.sun.faces.context.FacesContextImpl对象，FacesContextImpl类继承了jsf-api项目中的javax.faces.context.FacesContext.FacesContextImpl的构造方法的第一个参数是一个叫做 ExternalContext的接口的实现，查看其源代码，可以看到ExternalContextImpl类耦合了Servlet API，而FacesContextImpl与Servlet API无关。实际上，在这里，做到了JSF可以不仅仅使用在Servlet环境中，正如ExternalContext接口的注释中所说，在 Servlet环境中使用JSF和在Portlet环境中使用JSF的不同，实际上就是使用了不同的ExternalContext.在FacesContextFactoryImpl中构造FacesContextImpl的代码如下： 1 FacesContext ctx = new FacesContextImpl(new ExternalContextImpl((ServletContext) sc,(ServletRequest)

request,(ServletResponse) response),lifecycle). FacesContextImpl的构造方法中，还做了另外一件事情，就是根据配置确定了RenderKitFactory，显然不同的 RenderKitFactory可以产生不同的RenderKit，而不同RenderKit对象是针对不同客户端的，所以对于浏览器、移动设备等等，会有不同的RenderKit.FacesContextImpl的构造方法中代码如下： 1 this.externalContext = ec. 2 setCurrentInstance(this). 3 this.rkFactory = (RenderKitFactory)FactoryFinder.getFactory(FactoryFinder.RENDER_KIT_FACTORY). 在代码中我们经常使用FacesContext.getCurrentInstance（）这个静态方法来获取与当前请求对应的FacesContext 对象，实际上是在FacesContext类里面有一个静态的ThreadLocal对象用来存放了当前请求线程对应的FacesContext对象，于是上面的代码中setCurrentInstance（this）就是把当前构造出来的这个FacesContext对象放到了ThreadLocal里面。 FacesContext创建出来以后，正如上面所说，要让他经过LifeCycle这个“ Filter Chain ”的逐步处理了。那么，Filter Chain里面放的是一个一个Filter，那么LifeCycle这个Chain里面放的是什么呢？答案是Phases. FacesServlet让FaceContext通过LifeCycle的处理，分成了两个部分。一个部分是调用LifeCycle的execute方法，执行逻辑，第二个部分是调用LifeCycle的render方法，呈现响应。FacesServlet.service中代码如下： 1 lifecycle.execute(context). 2 lifecycle.render(context). 在LifeCycleImpl这个实现中，存放了一个Phase对象的数组，存放了7个Phase.其中第一个是null，然后依次是视图重建、应用请求值、验证、更新模型值、执行

应用程序、呈现响应。在execute方法中，调用了从视图重建开始到执行应用程序为止的5个Phase，而在render方法中，调用了最后一个Phase，也就是呈现响应。在LifeCycleImpl类中，代码如下： //The Phase instance for the render() method private Phase response = new RenderResponsePhase(). // The set of Phase instances that are executed by the execute() method // in order by the ordinal property of each phase private Phase[] phases = { null, // ANY_PHASE placeholder, not a real Phase new RestoreViewPhase(), new ApplyRequestValuesPhase(), new ProcessValidationsPhase(), new UpdateModelValuesPhase(), new InvokeApplicationPhase(), response }. 在Servlet Filter中，可以由每一个Filter来决定是否要调用下一个Filter，从而决定是否让请求继续通过Filter Chains中的后续Filter，是链式调用的过程。而在LifeCycle的execute方法中，是用一个for循环顺序执行几个Phase.在每一个Phase执行完之后，都会检查FaceContext对象中是否设置了停止后续处理直接呈现响应的标志（renderResponse）或者已经完成了响应无需后续处理也不需要经过呈现响应阶段了（responseComplete），如果标志为true，那么就不再执行后续Phase. LifeCycleImpl的execute方法主要代码如下： 1 for (int i = 1, len = phases.length -1 . i 100Test下载频道开通，各类考试题目直接下载。详细请访问www.100test.com