为何选择JSF不选StrutsJava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__E4_B8_BA_E4_BD_95_E9_80_89_E6_c104_644437.htm 对于是选择JSF还是选Struts，很多人都十分的困惑。JSF和Struts两个框架到底有什么区别？JSF和Struts推荐选择那一个比较好呢？ 总的来说，我建议在新项目中优先考虑JSF。虽然常常有一些商业上的因素迫使我们为现有的项目选择了Struts，而且那些解决方案还有待考验，但是，让我们面对一个事实：JSF比Struts好多了。 下面是我选择JSF而不选Struts的十大理由： 1.Components(组件) 2.Render Kits 3.Renderers 4.Value Binding Expressions(值绑定表达式) 5.Event Model(事件模型) 6.Extensibility(可扩展性) 7.Managed Beans(Dependency Injection 依赖注入) 8.POJO Action Methods 9.JSF is the standard Java-based web app framework (JSF是java web应用程序的标准框架) 10.Theres only one Struts(只有一个Struts) 10.Theres only one Struts(只有一个Struts) Struts是一个开源产品，然而JSF是一个标准。这个细节常常被新的JSF学习者忽略，其实这是显而易见的，因为我们有多个JSF的实现。虽然JSF还很不成熟，但是我们已经有了2个优秀的JSF实现可以选择：Sun的参考实现和Apache的MyFaces。另一方面，我们只有一个 Struts。 9.JSF is the standard(JSF是标准) JEE 5.0要提供一个JSF的实现，这表明JSF不久将会无处不在。这可能与你无关，但是和工具供应商密切相关。现在大概有50个java web应用程序框架，工具供应商不会情愿去支持一个特别的框架，但是他们会毫不犹豫的去支持一个标准。而且不止供应商，开源项目也会迅速的聚集在JSF的四周，争先恐后的去实

现相同的功能。比如说，直到我们去实现本质上和Shale的Tapestry差不多的视图的时候，我才知道Facalets。(从长远来看，我相信这种冗余是件好事，会给我们带来好处) 8.POJO Action Methods Struts的行为是和Struts的API绑定在一起的，但是JSF的行为方法可以在POJPO中实现。这意味着你不用在表单和模型对象之间实现一个多余的行为层。顺便说一下，在JSF里面没有行为对象，行为在模型对象中实现。但是也请注意一点：如果你愿意你也可以生成与JSF独立的行为对象。在Struts里面，你有Form. Bean和Action Bean。Form. Bean包含数据而Action Bean包含逻辑。OO狂会想去合并前2者，在Struts你办不到。但是在JSF中，你可以分开数据和逻辑，也可以合并到一个对象中，一切由你决定。 7.Managed Beans(Dependency Injection 依赖注入) 和Spring一样，JSF也使用了依赖注入(DJ)(或控制反转(IoC))去实例化和初始化Bean。Struts的确为你生成了Form. Bean和Action Bean，但是JSF可以为你生成各种各样的Managed Bean。 6.Extensibility(可扩展性) 这个很重要。JSF有6个对象实现了这个框架的大部分功能，而且你可以很容易的用你自己的实现代替原有实现。比如你想加一个自定义参数在JSF 表达式语言里面，或是添加一个自己的视图控制器以便于区分组件和HTML。事实上Shale实现了上面的功能。如果你还没有满足，JSF提供了几个地方你可以轻松的控制JSF的生命周期。Shale给你的会更多。 5.Event Model(事件模型) JSF的事件模型使你可以对值改变，动作，JSF生命周期阶段变换等作出反应。在JSF1.1中，那些事件都是在服务器端处理的，这肯定是一个缺陷，好在JSF2.0计划支持客户端事件，拭目以待吧。 4.Value Binding Expressions(

值绑定表达式) 在Struts中，你负责把数据从Form传递到模型对象。你实现的Action的execute方法是把Form作为一个参数。然后你再手动的把数据从Form. Bean里面取出放到模型对象里面。你要为应用里面的每个Form做这些事情，然而在JSF里面，你只需像这样：#{model.property} 就够了，其他的交给JSF来处理。 3.Renderers 你有看过Struts的标签的源代码吗? 它直接生成HTML。JSF组件标签什么都不生成，它和服务器上的一对component-renderer对应。Component维护组件状态，rendered负责获得视图。重点是renderers是可插拔的，即你可以根据自己需求实现然后替代掉默认实现。比如说我在NFJS上面的Felix谈话中举例说明了怎么去实现一个自定义的label renderer。你只需要配置你的renderer，JSF就会自动在你的应用程序里面使用他。 2.Render Kits 在几年前我曾经有份Struts咨询工作，我们必须同时支持浏览器和无线设备，非常痛苦。但是用JSF来完成那个任务非常容易，因为你可以生成你自己的render kit-为一种特定显示技术的renderers的集合-然后配置到JSF里面。 1.Components(组件) 组件是Struts和JSF之间最大的区别。就像Swing一样，JSF提供丰富的底层构件去开发组件然后添加到标准的组件集。那些底层构件让你很容易的生成自己的组件并且和别人共享。现在我们到处都能看到自定义组件跳出来，比如说Oracle的 ADF和MyFaces，两者都提供了丰富的组件集，就像javascript日历，tree等等。当然，组件只是一部分。典型的是，组件都和一个独立的 renderer对应，这给我们带来了真正的好处(看第3条)。但是和JSF中的很多东西一样，你不一定要墨守成规。只要你愿意，你可以实现render 自己的组件，虽然这样你会失去给组件加入别

的renderer的能力。 