"插件"与我们通常所说软件"模块"的一个区别是：插件能自我描述，加载运行在插件容器中

这种情况下，spring的各个组件实际上都是在spring这个容器中运行的。spring容器本身运行的流程，是spring设计者规定的（相当于servlet的规范）。同时，设计者在流程中预先就保留了一些扩展点，等着后来的人（spring的用户）去自行扩展

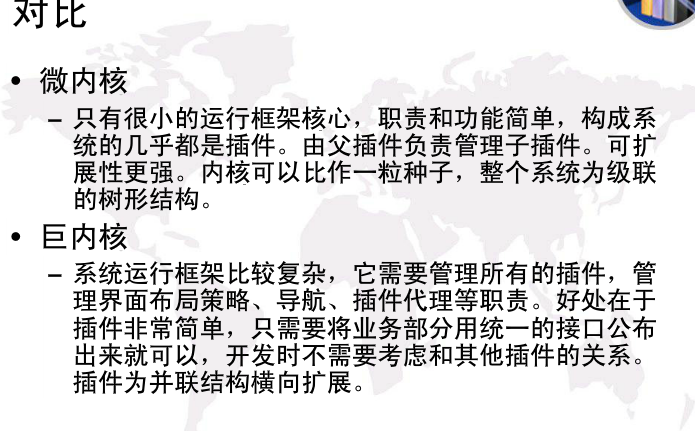
因此，spring能不能扩展，能在哪些环节被扩展，是在设计的时候就决定的。试想如果rod johnson在一开始，就没有设计查找并执行PostBeanDefinitionProcess这个环节，那我们就不能在这个环节上对spring系统进行扩展了

日常我们写的系统，也是这样。以后能不能扩展，在哪里扩展，怎么扩展，都是在设计的时候就决定的。如果系统中，某个环节调用某个组件，都被固定下来，是“写死的”，那么这个环节就不具扩展性了。当然这个时候，系统依然是容器，只是容器中的组件是不可变的

1. 可扩展性的要素   
     
   总结上面的例子，一个系统是否具有扩展性，是在设计之初就固定下来的。   
     
   那么系统要有扩展性，就至少需要3个要素：   
     
   1、在规定了流程的前提下，允许某些环节扩展   
   2、将允许扩展的部分，以API的方式对外部提供   
   3、对外部提供规则
2. 所谓的插件   
     
   写本文的原因，其实是最近在分析一个系统。这两天在研究它的插件体系，想到这么多就写下来   
     
   搞清楚上面的内容，插件也就不复杂了   
     
   从用户的角度来看，要写一个插件，就是拿到API，然后按照规则写扩展组件   
     
   从系统（容器）设计者的角度看，我的系统要支持插件扩展，就是：   
     
   1、规定流程，设计扩展点（包括加载机制）   
   2、把扩展点打包成API，作为二次开发的SDK提供给用户   
   3、告诉用户，应该怎么使用这个API

系统来说，并不知道插件的具体功能，仅仅是为插件留下来的预订接口，系统启动时候根据插件的配置寻找插件，根据预订接口把插件挂接到系统中。

那模式主要有两种类型，一种是微内核级联树形结构，一种是巨大内核，(管理容器，并列结构。



插件界面 xnml iframe模式

苏占玖：开放式插件系统研究.pdf - Java开发 - OPEN开源文档.htm