**Tomcat的默认连接器**

Tomcat中的连接器是一个独立的模块，可以被插入到servlet容器中。Tomcat中使用的连接器必须满足以下要求：

实现org.apache.catalina.Connector接口；

负责创建实现了org.apahe.catalina.Request接口的request对象；

负责创建实现了org.apache.catalina.Response接口的response对象

Tomcat4中的默认连接器的**工作原理**：它会等待引入的HTTP请求，创建request对象和response对象，然后调用org.apache.catalina.Container接口的invoke()方法，将request对象和response对象传给servlet容器。Invoke()方法的签名如下：

Public void invoke(org.apache.catalina.Request request, org.apache.catalina.Response response)

在invoke()方法内部，servlet容器会载入相应的servlet类，调用其service()方法，管理session对象，记录错误消息等操作。

**Connector接口**

Tomcat的连接器必须实现org.apache.catalina.Connector接口。在接口中声明了很多方法，其中最重要的是getContainer()、setContainer()、createRequest()和createResponse()

setContainer()方法用于将连接器和某个servlet容器相关联。getContainer()方法返回与当前连接器相关联的servlet容器。createRequest()方法会为引入的HTTP请求创建request对象，相应的，createReponse()方法会创建一个response对象

**servlet容器**

servlet容器是用来处理请求servlet资源，并为web客户端填充response对象的模块。Servlet容器是org.apache.catalina.Container接口的实例。在tomcat中，共有4种类型的容器，分别是：Engine、Host、Context和Wrapper。

**Container接口**

Tomcat中的severlet容器必须要实现org.apache.catalina.Container接口。而且需要将servlet容器的实例作为参数传入到连接器的setContainer()方法中，这样连接器才能调用servlet容器的invoke()方法。

四种类型的容器——

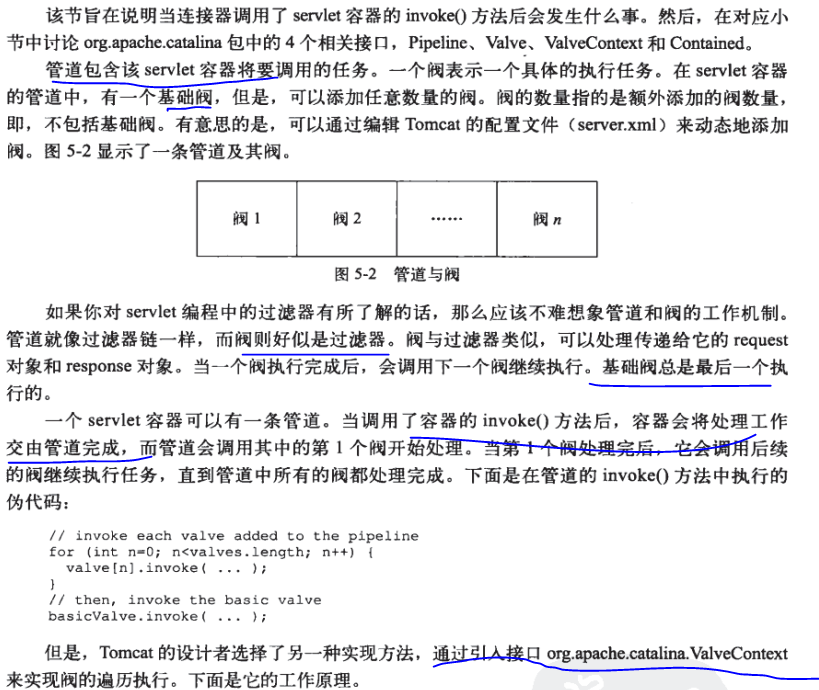
Engine：表示整个catalina servlet引擎；

Host：表示包含有一个或多个Context容器的虚拟主机；

Context：表示一个web应用程序。一个context可以有多个wrapper；

Wrapper：表示一个独立的servlet。

1. **管道任务**



**Pipeline接口**

对于pipeline接口，首先要提到的一个方法是invoke()方法，servlet容器调用invoke()方法来开始调用管道中的阀和基础阀。通过调用pineline接口的addValue()方法，可以向管道中添加新的阀，同样，也可以调用removeValue方法从管道中删除某个阀。最后，调用setBasic方法将基础阀设置到管道中，调用其getBasic方法可以获取基础阀。基础阀是最后调用的阀，负责处理request对象及其对应的response对象。

**Value接口**

阀是Value接口的实例，用来处理接收到的请求。该接口有两个方法，invoke()方法和getInfo()方法，getInfo()返回阀的实现信息。

**ValueContext接口**

该接口有两个方法，invokeNext()和getInfo()。getInfo()方法会返回ValueContext的实现信息。

**Contained接口**

阀可以选择是否实现org.apache.catalina.Contained接口，该接口的实现类可以通过接口中的方法至多与一个servlet容器相关联。

1. **Wrapper接口**

Wrapper级的serlvet容器是一个org.apache.catalina.Wrapper接口的实例，表示一个独立的servlet定义。Wrapper接口继承自Container接口，又添加了一些额外的方法。Wrapper接口的实现类要负责管理其基础servlet类的servlet生命周期，即，调用servlet的init()、service()、destroy()等方法。由于Wrapper已经是最低级的servlet容器了，因此不能再向其中添加子容器。

Wrapper接口中比较重要的方法是load()和allocate()方法。allocate()方法会分配一个已经初始化的servlet实例。而且，allocate()方法还有考虑下该servlet是否实现了javax.servlet.SingleThreadModel接口。load()方法载入并初始化servlet类。

1. **Context接口**

Context接口的实例表示一个web应用程序，一个context实例可以有一个或多个Wrapper实例作为其子容器。Context接口中比较重要的方法是addWrapper()方法和createWrapper（）方法

五、servlet容器 P82

**5.5. Wrapper应用程序 P89**