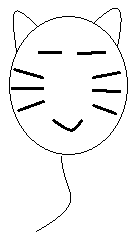
LightCat说明

LightCat，是一个轻巧的Http服务器。她是在本人参考了SpringMVC和Tomcat的核心处理算法的设计出来的精简web服务器。她支持常见的Http请求：普通请求，上传，下载。

其Logo如下：（轻巧—轻如气球）



相对于Tomcat，她具备如下特点：

1. **树形匹配URI ，更高效**：

LightCat其实本身就是一个树，而其发布的每个应用也是一棵树：发布的应用中的资源（静态，动态）的路径会被拟合成一棵uri树,而这些资源都犹如挂在路径树上面的。

树形匹配URI，只需要匹配uri的片段节点，其实会比tomcat的全路径匹配高效得多！

1. **更细腻的拦截**： URI匹配树，树上的每个节点都是一个uri片段，我们可以轻松在每个（某个）节点上配置过滤器Filter拦截，一旦配置，那么访问挂靠在此节点下的所有叶节点的请求都

会被拦截过滤！（这是参考了springMVC的拦截思想）

1. **砍掉formard**：不好意思，出于轻便思考，取消了内部跳转的功能。每个请求，都只在一个Action中处理完毕，为何还要跳来跳去的呢？

D. **打算仅支持Get、Post 两种提交**：这是其轻巧的原因之一：其实我们在开发应用的过程中，会使用到Put，Delete，Head等这些提交Method的情况是极少的，所以LightCat计划将其他的不常见的提交Method取消。不过目前还留着。

E. **调用栈浅**：相对于Tomcat,LightCat的调用栈是非常浅的，因为砍掉了好多不常用的概念！

F．**杜绝页面存在动态语言**，例如jsp中的java片段。页面(VIEW层)就只是HTML代码！！其他的内容，待浏览器加载完html页面后再通过js发出请求获取数据后渲染。

G．除了访问的是静态资源（html,css,js,图片等），其他与后台交互，都是通过JSON。

.

.

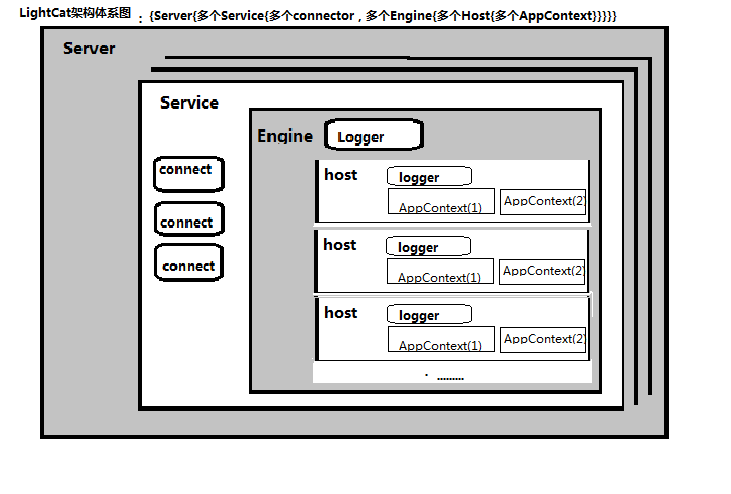
.

.

.

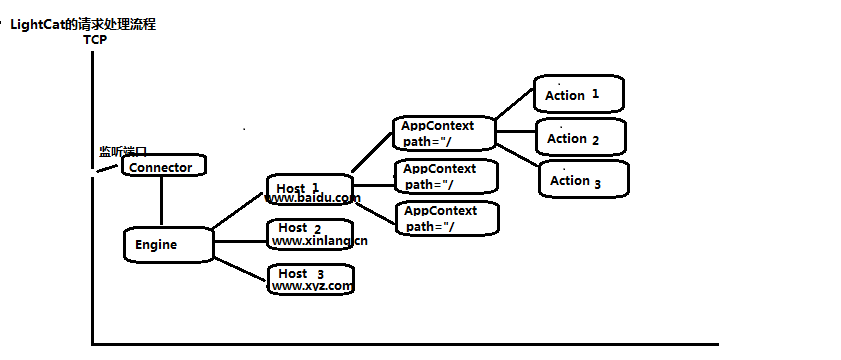
.

（1）LightCat架构图：



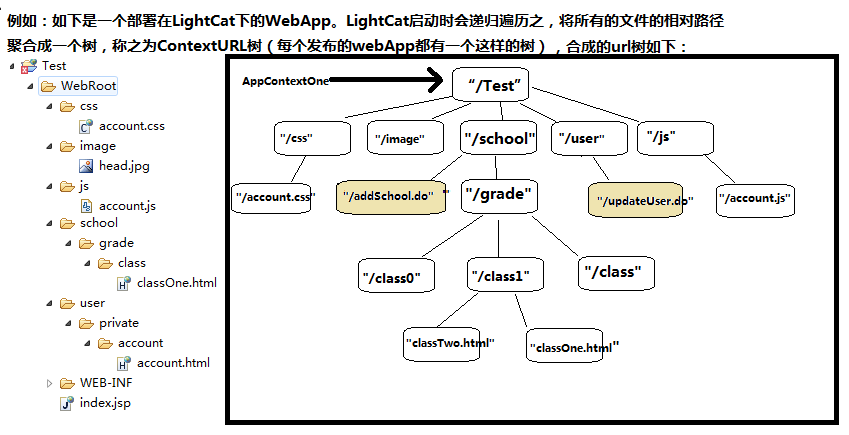
* Server(服务器) 容器，是Tomcat构成的顶级构成元素，所有一切均包含在Server中，Server的实现类 Server可以包含一个到多个Services;
* 次顶级元素Service容器指明了该Service归属的Server；
* 主机(Host)、上下文(Context)和引擎(Engine)，它们都是容器。但是，它们是有父子关系的，在主机(Host)、上下文(Context)和引擎(Engine)这三类容器中，引擎是顶级容器，直接包含是主机容器，而主机容器又包含上下文容器，所以引擎、主机和上下文从大小上来说又构成父子关系。
* 连接器(Connector)的作用就是把来自客户端的请求转发到Container(容器)。

（2）**LightCat请求处理流程：**



（3）LightCat的URI匹配算法：树节点递归匹配。

假如现在有一个应用：**Test**:其目录如下：



Test应用中内置了一个URL树，树的根节点是此应用的名字。

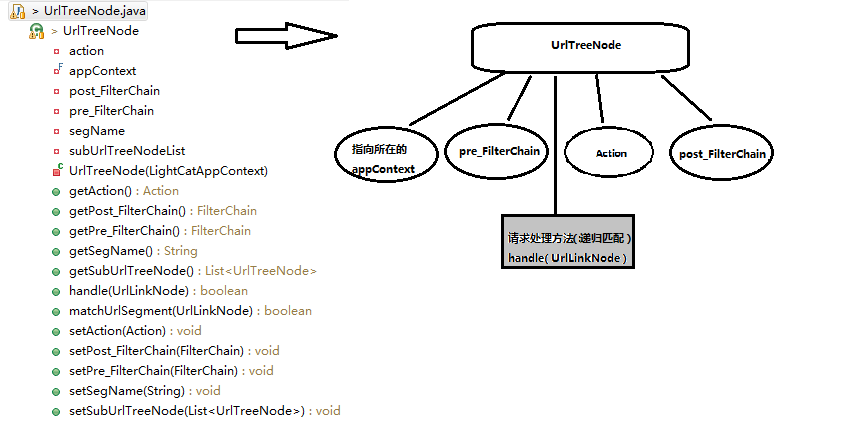
ContextURL树上的每一个节点，都是路径segment。树中的路径包括了Action对象的url和静态资源的url。

（4）这是一棵什么树？

ContextURL树的每个节点，不仅仅是封装了路径segment，还封装了处理器Action，前后过滤器（Filter）!!这意味着，ContextURL树每个segment都可以配置过滤器拦截请求！！！

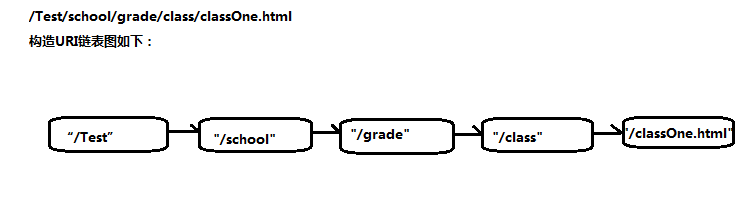
一下是ContextURL树的节点结构图：

UrlTreeNode：



（4）那么，如何和请求的路径递归匹配？

首先将http请求行的uri合成一条uri链表：



于是在和ContextUrl树开始递归匹配（在匹配的过程中每个环节都过滤（如果配置有过滤器的话），），一直到终点目标为止。（当然如果目标不存在/被拦截了，匹配就会终止！）

详见源码：UrlTreeNode 的 handle(UrlLinkNode)