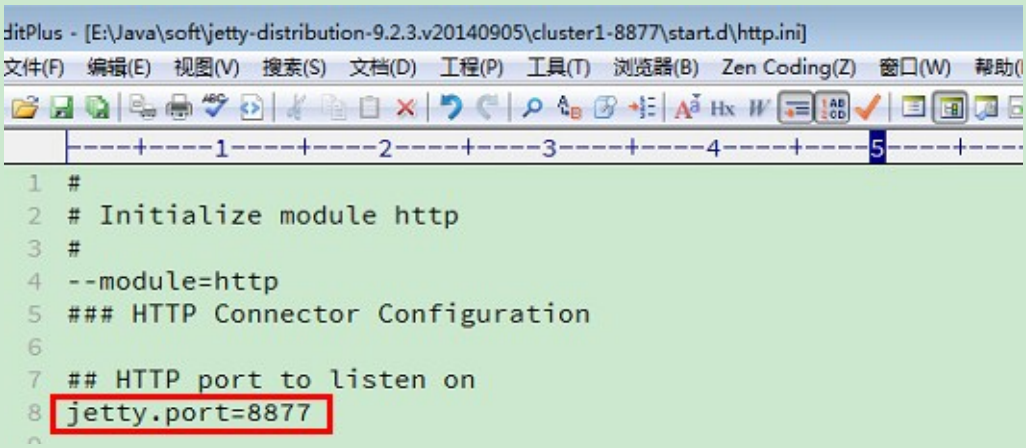
使用自带的 BalancerServlet

同一台机器上的一个 Jetty 实例做负载均衡器，另外两个 Jetty 实例做为 slave(s),还有另一台机器上的 Jetty 做 slave.
首先创建 `Balancer`，创建一个 Java 应用程序即可,最主要的就是设置以 `balancerMember.` 开头的初始参数.它的作用就是将请求转发到其它的 Jetty 实例.

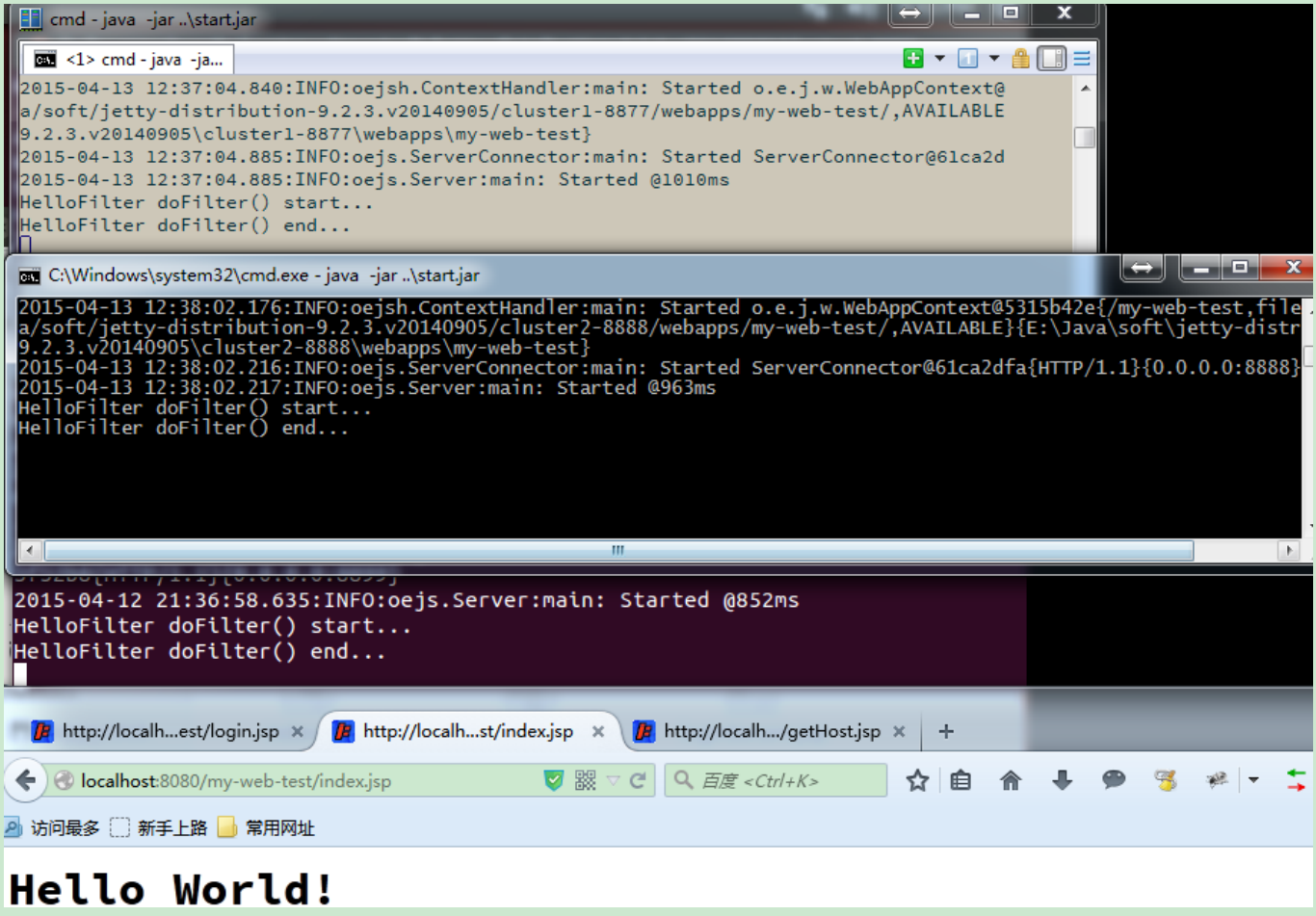
```
1.  public class BalancerServer {
2.      private boolean stickySessions;
3.      private Server balancer;
4.      private static final String CONTEXT_PATH = "/*";
5.      private static final String SERVLET_PATH = "";
6.
7.      public static void main(String[] args) throws Exception {
8.          BalancerServer server = new BalancerServer();
9.
10.         server.startBalancer();
11.     }
12.
13.     private void startBalancer() throws Exception {
14.         ServletHolder balancerServletHolder = new ServletHolder(BalancerServlet.class);
15.         //balancerServletHolder.setInitParameter("stickySessions", String.valueOf(stickySes
    sions));
16.         //balancerServletHolder.setInitParameter("proxyPassReverse", "true");
17.         // 配置多个节点
18.         balancerServletHolder.setInitParameter("balancerMember." + "cluster1" + ".proxyTo",
     "http://localhost:8877");
19.         balancerServletHolder.setInitParameter("balancerMember." + "cluster2" + ".proxyTo",
     "http://localhost:8888");
20.         balancerServletHolder.setInitParameter("balancerMember." + "cluster3" + ".proxyTo",
     "http://192.168.119.134:8899");
21.
22.         balancer = createServer(balancerServletHolder, 8080);
23.         balancer.start();
24.         balancer.join();
25.     }
26.
27.     private Server createServer(ServletHolder servletHolder, int port) {
28.         Server server = new Server(port);
29.         ServerConnector connector = new ServerConnector(server);
30.         server.addConnector(connector);
31.
32.         ServletContextHandler context = new ServletContextHandler(server, CONTEXT_PATH, Ser
    vletContextHandler.SESSIONS);
33.         context.addServlet(servletHolder, SERVLET_PATH + "/*");
34.
35.         return server;
36.     }
37. }
```

从上面的代码中可以看到我们配置了两个 `slave` 节点.创建两个目录,如下:

里面的内容一样.修改各自的 `http.ini` 文件中的端口配置即可.



另外还有一台 Ubuntu 机器上的配置与此类似.接着启动这 4 个 Jetty 实例.然后用浏览器发出请求.这里我们都是用 `http://localhost:8080` .可以看到请求都平均到了每个 Jetty 实例.



更具体的信息可以参考 `org.eclipse.jetty.proxy.BalancerServletTest` 或阅读 `org.eclipse.jetty.proxy.BalancerServlet` 类相关的源码.

如果说其中的一台(slave1)挂了,那么当一个请求正好落到 slave1 的时候, 是不会主动切换到其他的 slave

上去的，这点是不如 Apache Http Server 的.

使用 Apache Http Server 做负载均衡器

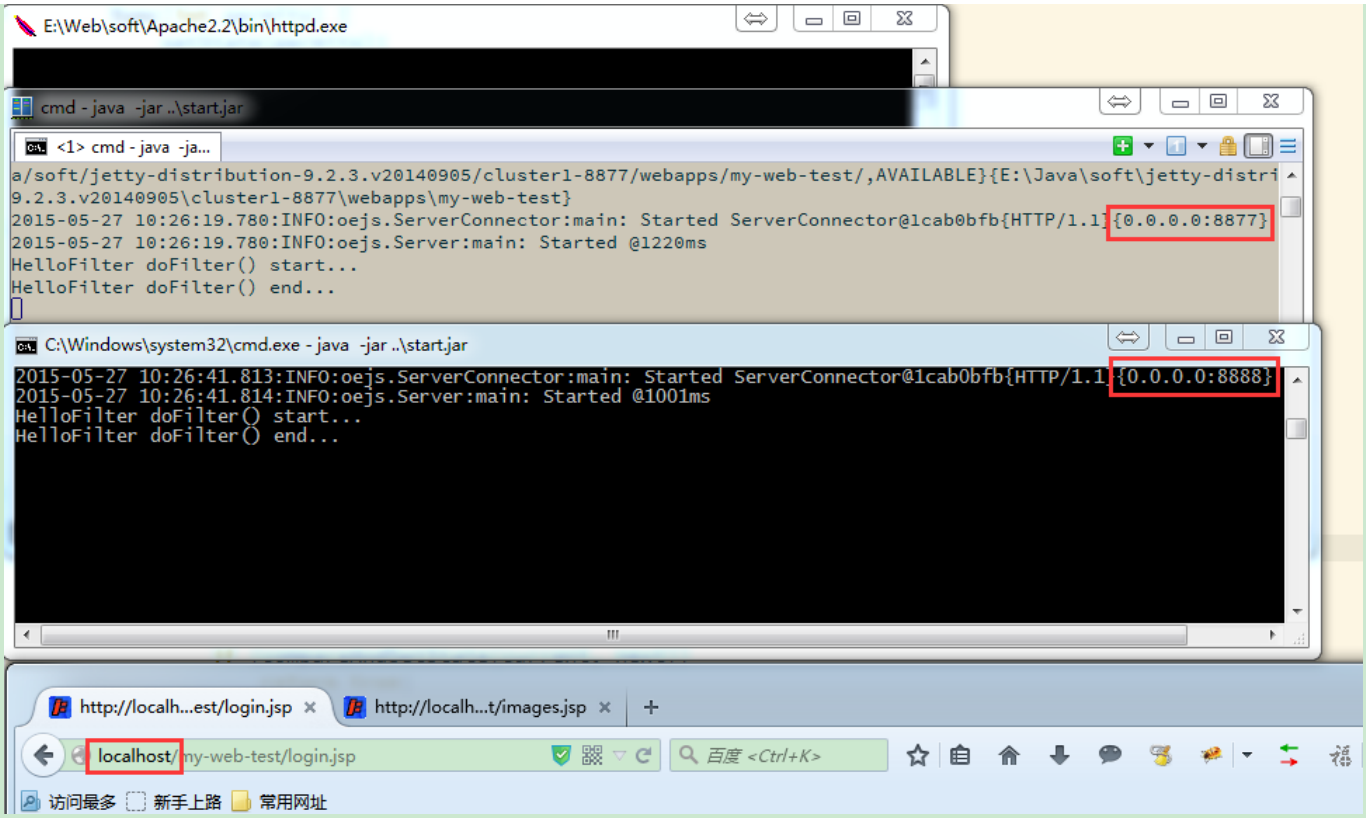
Apache Http Server 与 Tomcat 集成，大家都很熟悉了，google 一下可以搜索出很多相关信息.如果换成 Apache 与 Jetty，貌似搜索不出啥信息.其实也很简单，也就是修改下 Apache Http Server 的配置，如下:

httpd.conf

```
# 负载均衡配置
LoadModule proxy_module modules/mod_proxy.so
#LoadModule proxy_ajp_module modules/mod_proxy_ajp.so
LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
LoadModule proxy_connect_module modules/mod_proxy_connect.so
#LoadModule proxy_ftp_module modules/mod_proxy_ftp.so
LoadModule proxy_http_module modules/mod_proxy_http.so

ServerName localhost:80
<VirtualHost *:80>
    ServerAdmin admin@china1168.com
    ServerName localhost
    ServerAlias localhost
    <proxy balancer://testcluster>
        # 主要修改这里的配置
        BalancerMember http://localhost:8877 loadfactor=1 route=jetty-8877
        BalancerMember http://localhost:8888 loadfactor=1 route=jetty-8888
    </proxy>
    ProxyRequests Off
    ProxyPass /server-status !
    ProxyPass /balancer-manager !
    ProxyPass / balancer://testcluster/ stickysession=JSESSIONID
    ProxyPassReverse / balancer://testcluster/
</VirtualHost>
```

看下运行结果:



如果某一台服务器挂掉，那么 Apache Http Server 可以自动的切换到另一台.这比 Jetty 自带的 `BalanceServlet` 要做的更好.

另外如果是同一个 URL 请求，Apache Http Server 会将此请求定位到第一次选择的 Jetty 服务器,这样以后这个请求的 URL 都会落到同一个 Jetty 服务器上.这样做有啥好处，最容易想到的就是避免同一个请求发送到多台 Jetty 服务器，从而产生多余的 `session` 信息,发送到多台服务器,那么每台服务器都会新建一个 `session` .不过这个问题也可以通过做 Jetty 的 session 集群来避免.另外，Jetty 也可以实现相同的方式，那么就是在 `sessionid` 后面追加集群节点的名称.比如,在返回响应之前修改 `sessionid` ，假设原来为 `JSESSIONID=151y2tr488szfj3lok2uwh68m` ,那么我们可以修改为 `JSESSIONID=151y2tr488szfj3lok2uwh68m.cluster1` .可以查看 `BalancerServlet#selectBalancerMember()` 方法的源码,它的内部会调用 `extractBalancerMemberNameFromSessionId` .不过有个前提就是需要配置 `BalancerServlet` 的初始参数 `stickySessions` 为 `true` .

```
1.   // 选择节点
2.   private BalancerMember selectBalancerMember(HttpServletRequest request) {
3.       if (_stickySessions) {
4.           // 会调用 extractBalancerMemberNameFromSessionId() 方法
5.           String name = getBalancerMemberNameFromSessionId(request);
6.           if (name != null) {
7.               BalancerMember balancerMember = findBalancerMemberByName(name);
8.               if (balancerMember != null)
9.                   return balancerMember;
10.          }
11.      }
12.      int index = (int) (counter.getAndIncrement() % _balancerMembers.size());
13.      return _balancerMembers.get(index);
14.  }
15.
16.
17.  // 从 sessionid 中提取 balance member name
18.  // sessionid;balance member name
```

```
19.   private String extractBalancerMemberNameFromSessionId(String sessionId) {
20.       int idx = sessionId.lastIndexOf(".");
21.       if (idx > 0) {
22.           String sessionIdSuffix = sessionId.substring(idx + 1);
23.           return sessionIdSuffix.length() > 0 ? sessionIdSuffix : null;
24.       }
25.       return null;
26.   }
```

到底选择哪种方式好，这个根据自个的需求来即可.