Web 2.0-Comet-Continuations模式在局部放电实时监测中的应用

王默玉,刘 岩,刘 林,李成榕,王 辉

(华北电力大学 计算机科学与技术学院,北京 102206)

摘 要:针对传统Web模式的实时系统具有的用户体验不理想、数据延迟高和响应速度慢等缺点,通过对Ajax技术和Web工作模式深入分析研究,采用Web 2.0、Comet、Continuations相结合的方式,结合电缆终端局部放电实时监测系统例证,将三维谱图网格化并结合div技术模拟桌面应用程序局部刷新。该模式下系统具有无闪烁、实时性高的特点,并且即使在连接大量客户端时,服务器也只需消耗较少的CPU和内存资源。

关键词:计算机应用;Web模式;Comet;Ajax;Continuations;实时监测;局部放电

中图分类号:TP311 **文献标识码**:A **文章编号**:1671-5497(2009)01-0164-06

Application of Web 2.0-Comet-Continuations model in real-time monitoring of partial discharge

WANG Mo-yu, LIU Yan, LIU Lin, LI Cheng-rong, WANG Hui

(*College of Computer Science and Technology, North China Electric Power University, Beijing* 102206, *China*)

Abstract: The real-time system based on the traditional Web mode is characterized by unsatisfactory user experience, high latency for the browser, low response rate and so on. Consequently, taking the monitoring of partial discharge on the cable terminals as an example, through the analysis of the Ajax technique and Web mode, with the combination of the Web 2.0, Comet and Continuations, a real-time monitoring system was developed based on Web and discretization of the 3-D map coupled with div technique to simulate the local renovation of the application program on desk. Based on this mode the system is characterized by no-flashing display, high quality real-time response and need of the server for little resource of CPU and memory against huge quantity of client ends.

Key words: computer application; Web mode; Comet; Ajax; Continuations; real-time monitoring; partial discharge

为了更好地解决电网设备的远程实时监测问题,人们提出了B/S结构,该结构具有分布性强、维护方便、开发简单且共享性强、总体拥有成本低等优势,然而在传统的Web工作模式下构建实时监测系统要及时刷新整个页面,存在用户界面上有明显的闪烁现象以及通信量大等问题。随着Web 2.0中一些相关技术的出现使得上述问题的解决成为可能,例如监测股市行情的大盘走势图

收稿日期:2008-04-22.

基金项目:北京市重点实验室项目(SYS100790442).

作者简介:王默玉(1961-),女,副教授. 研究方向:计算机技术与应用. E-mail:wangmoyu@ncepu.edu.cn.

通信作者:刘岩(1981-),男,硕士研究生. 研究方向:计算机技术与应用. E-mail:liuyan00808@yahoo.com.cn

就是一个实时监测系统。为了实现一个性能良好的基于 Web 的电缆终端局部放电实时监测系统，本文作者对 Web 2.0 中相关重要技术进行了详细的分析研究，并通过对几种方案的试验对比发现，对于 java 语言开发服务器端，以 Web 2.0、Comet、Continuations 相结合的模式是理想的选择，能够很好地满足电缆终端局部放电实时性的要求，该系统在福建某变电站部署完成后一直运行良好。

1　相关概念

Web 2.0 是以 Blog、TAG、SNS、RSS、Wiki 等应用为核心，依据六度分割、XML、Ajax 等新理论和技术实现的互联网新一代模式。Web 2.0 与 Web 1.0 的区别是：Web 1.0 的主要特点在于用户通过浏览器获取信息，以数据为核心；而 Web 2.0 是以人为核心，更注重用户的交互作用。Web 2.0 采用 Ajax 等技术可以为用户构建类似桌面应用程序的胖客户端，而这种胖客户端在显示动态数据方面其优势远胜于 Web 1.0 的瘦客户端[1]。

Ajax 是 Asynchronous JavaScript and XML 的缩写，意思是异步的 JavaScript 与 XML。Ajax 实际上是 CSS、JavaScript、XHTML、XML、XMLHttpRequest、DOM 以及 XSLT 等七种技术的综合[2]，旨在构建基于 Web 的胖客户端。具体构建方式为：①使用 XHTML 和 CSS 标准化呈现[3]；②使用 DOM 来动态显示和进行交互；③使用 XML 和 XSLT 来进行数据交换和显示[3]；④使用 XMLHttpRequest 来获取异步信息[4]；⑤使用 JavaScript 将上述六项技术整合起来。

Ajax 应用程序可以采用服务器与浏览器的连接始终打开并在数据可用时发送给浏览器的方式，这种长期连接技术被称为 Comet，也就是说 Comet 是 Ajax 技术的一种应用模式[5]。

在使用 Ajax 技术的异步 Web 2.0 应用程序中，内存消耗会随线程数的增加而急剧增长。为了解决该问题，Web 服务器 Jetty 6 引入了一个新特征 Continuations。当一个 java Filter 或 Servlet 处理一个 Ajax 请求时，有可能需要一个 Continuations 对象，该对象用于有效挂起 Ajax 请求和释放当前线程的资源，在超时之后或者 Continuations 对象调用了 resume 方法时该请求被唤醒[6]。

2　Web2.0-Comet-Continuations 的优势

2.1　传统 Web 应用程序

在传统的 Web 应用程序交互方式中，客户端和服务器的数据交互是同步（synchronous）响应的。其响应方式如图 1 所示。

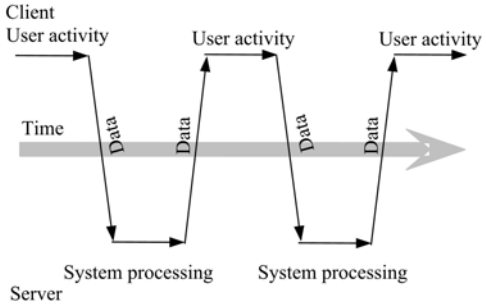

图 1　传统 Web 应用同步响应模式

Synchronous modes of traditional Web applications

用户发出一个 HTTP 请求到服务器，服务器对其进行处理后再返回一个新的 HTML 页到客户端，每当服务器处理客户端提交的请求时，客户都只能空闲等待，哪怕只是很小的交互，或者只需从服务器端得到很简单的一个数据，都要返回一个完整的 HTML 页。由这种模式实现的实时系统中一般都采用定时刷新页面的策略，即定时向服务器发出请求以便从服务器获取最新数据。这种实时系统的缺点是：①频繁地刷新整个页面会使用户感觉到整个屏幕在不停地闪烁，也很难在这样的页面上做一些操作；②造成了大量无效通信，即使实时系统只需更新页面中很小的一个部分或服务器端根本没有新数据，在该模式下也要将无需更新的大部分数据重新从服务器传输一遍。这样的实时系统构建出来是低效的，其实时性能等必然会受到很大的限制。

2.2　Ajax 应用程序

由图 2 显示的 Ajax 工作模式不难看出，基于 Ajax 应用程序数据显示和数据传输是异步进行的，也就是当一次数据请求或数据传输正在进行时，原来的数据显示界面可以不受影响。在新旧两个界面变化的过渡时间上，传统模式包括了请求传输时间、服务器端处理时间、新数据传输时间和新数据显示时间四部分；而 Ajax 模式只需要新数据显示时间，另外三部分的时间用户是感觉不到的。因此，在 Ajax 模式下的实时系统界面类似

于桌面应用程序,图像可以平滑地切换刷新,闪烁现象也可以消除。同时异步获取数据,可以要求服务器只返回客户端更新需要的有效数据,减少了数据通信量,节省了带宽和传输时间。

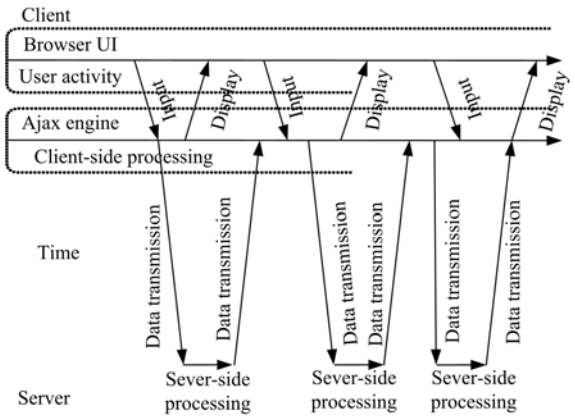

图 2 Ajax 应用异步响应模式

Fig. 2 Asynchronous modes of applications based on Ajax

2.3 Comet 应用程序

上述 Ajax 工作模式对于实时系统仍然存在不合理的地方:只能由浏览器建立 Web 浏览器和服务器之间的 HTTP 连接(见图 2),服务器只能被动地等待客户端的请求,收到请求后将新的数据作为响应返回给客户端,服务器无法在改动发生时将变化“推送”给浏览器。基于 Ajax 的实时应用程序可以使用两种基本的方法解决这一问题:一种方法是浏览器定期向服务器发出轮询以进行更新,另一种方法是长期连接技术 Comet。

轮询方法的主要缺点是:当扩展到更多客户机时,将生成大量的通信量。每个客户机必须定期访问服务器以检查更新,这为服务器资源添加了更多负荷。而使用 Comet 策略的优点之一是其显而易见的高效性,客户机不会像使用轮询方法那样生成大量的通信量,并且事件发生后可立即发布给客户机。Comet 的工作模式见图 3。

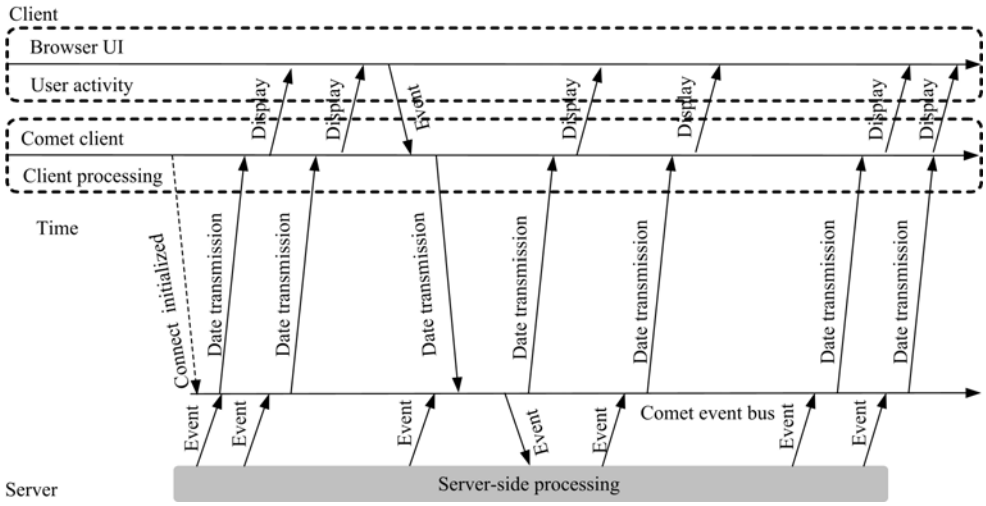

图 3 Comet 应用异步响应模式

Fig. 3 Asynchronous modes of applications based on Comet

由图 3 所示的 Comet 应用响应模式可以看出,服务器端可以在任意时刻向客户端发送数据,不一定要等待客户端的服务请求。数据在一个预先打开的连接上传送,这种方法明显减少了数据的传输延迟[7]。在该模式下的实时系统中,服务器端有新数据时可以即时将服务器端的最新数据推送到客户端。

2.4 Comet-Continuations 应用程序

如果说以上 Ajax/Comet 主要是解决客户端问题的技术,那么 Continuations 技术就是用来解决服务器端相应问题的。在基于 Ajax/Comet 模式的异步 Web 2.0 应用程序中,内存消耗会随线程数的增加而急剧增长。由于采用 java servlet 作为服务器端的开发语言,而传统的 java servlet API 像大多数 java APIs 一样是同步的,并且处理 HTTP 请求的持久性问题时采用的是一个请求分配一个线程的方式。这种方式在 Web 1.0 模式下能够很好地工作;但 Web 2.0 的应用程序(尤其是使用了 Ajax 和 Comet 技术的应用程序)则呈现出了新的数据传输路线图。在这种传输结构下,若仍用以前的 thread per connection 和 thread per request 方式,保持长期连接处于打开状态会消耗服务器资源。当等待状态的 servlet 持有一个持久性请求时,该 servlet 会独占一个线

程，这将限制 Comet 对传统 servlet 引擎的可伸缩性，因为客户机的数量会很快超过服务器栈能有效处理的线程数量。这种情况利用 java 的 NIO 库可以得到改善，NIO 库允许使用异步 IO，并且仅对请求将被处理时的线程分配连接。当连接处于两个请求处理之间的间隔时，相应的线程将被返回给线程连接池并且连接被加入到 NIO 选择集合中来监测新的请求，这种工作方式称为 thread-per-active-request 模式[8]。为了较好地支持该模式，Web 服务器 Jetty 6 引入了 Continuations 技术。

为了更好地说明三种模式在资源消耗方面的优劣，引用了文献[8]的测试结果（见表 1）。表中 Web 1.0 列显示了一个 Web 1.0 的服务器处理 10 000 个用户的连接，仅用 500 线程和 36 MB 的线程栈就足够了；表中 Web2.0＋Comet 列显示了在 Web 2.0 下采用 Ajax/Comet 技术的应用程序处理 10 000 用户的连接，却使得代价剧增到需要 10 600 线程和 694 MB 的线程栈，这样的代价可能会超过当前服务器的承受能力；而 Web 2.0 ＋ Comet ＋ Continuations 列显示出引入了 Continuations 技术后的 Ajax/Comet 模式要比引入之前所耗代价大大降低，仅需要 875 个线程和 57 MB 的栈，这样的代价仅次于 Web 1.0 的，当前服务器完全可以满足这样的要求，说明引入 Continuations 技术取得了非常大的提高。

通过以上论述，不难看出在 Web 2.0、Comet、Continuations 相结合模式下的应用程序能够带给用户较好的用户体验，具有数据延迟少、资源消耗少、数据传输效率高、实时响应性好等优势。由于服务器端可以直接将数据推送到客户端，在这样的环境下，实时系统实时响应性比较高。

表 1　3 种模式性能对比表

Table 1　Performances comparison of three modes

Parameter	Formula	Web 1.0	Web 2.0＋Comet	Web 2.0＋Comet＋Continuations
Users	u	10000	10000	10000
Request/Burst	b	5	2	2
Burst period/s	p	20	5	5
Request duration/s	d	0.200	0.15	0.175
Poll duration/s	D	0	10	10
Request rate/(req·s^{-1})	$rr=u*b/20$	2500	4000	4000
Poll rate/(req·s^{-1})	$pr=u/d$	0	1000	1000
Total/(req·s^{-1})	$r=rr*pr$	2500	5000	5000
Concurrent/(req·s^{-1})	$C=rr*d+pr*D$	500	10600	10700
Min threads	$T=c$	500	10600	—
	$T=r*d$	—	—	875
Stack memory	$S=64*1024*T$	32 M	694 M	57 M

3　应用设计

本系统选用 Oracle9i 作为服务器的数据库平台，以 Jetty 6 为 Web 服务器，以 Jbuilder 2005 作为开发平台，使用的语言有 Java Servlets、Jsp、JavaScript、DWR 2.0 等。

在电缆终端局部放电实时监测系统中，Web 2.0＋Comet＋Continuations 模式主要用于将下位机产生的三维谱图实时地“推送”到客户端显示。如图 4 所示的三维谱图，纵坐标表示一个采集周期内最大放电量，横坐标表示相位，图中 2 个小方块为放电信号（可用不同颜色表示一个采集周期内放电次数的不同级别，实际系统是用红、黄、蓝、黑 4 种颜色依次表示 4 个不同的放电等级，严重程度是依次递减的）。如果每次将整个三维谱图图片传到客户端，不仅占用较大的带宽，而且还会造成屏幕的闪烁，效果较差。经过对三维谱图的分析获知，各张谱图的不同主要是在图中放电信号的标示位置不断变换。因此，将大图片网格化为 43 * 47 个小图片，第一次显示时重新将它们组合成大图片，以后每次只是根据放电量最大值和相位确定放电信号标示应该覆盖哪两个小图片就可以了，其他大部分图片不用改变。因此，系统只有当实时界面被初次打开时才需要将所有小图片从服务器“推送”到客户端，以后只需将放电量最大值、相位和放电次数级别三个数据“推送”到客户端，然后客户端代码负责正确地用信号标示覆盖定位到的小图片。这样可充分发挥 Comet 模式的优势，使通信由原来每次传送 40.5 K 的图片变成传送几个字节的数据，大大减少了

通信量,通过移动信号标示小图片的位置实现了局部刷新。图4为实时显示的效果(小方块为黑色,表示放电现象不严重)。依据放电量和相位定位小图片确定位置的代码如下:

newX=parseInt(5+((41−5+1) * x/360))

newY=2-parseInt(((y−1.00) * ((43−2+1)/(1.00−(−1.00)))))

式中:newX、newY为被定位到的小图片的坐标;x、y分别表示相位和放电量最大值。

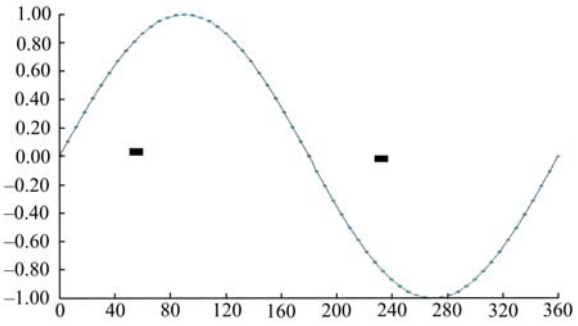

图4 局部放电三维谱图

Fig. 4 3D pattern chart of partial discharge

实验证明,在一台配置为CPU 1.7 GHz,内存1G的机器上运行服务器端,其CPU占用率一般为2%左右,且客户端数量的增加对服务器端的资源消耗量影响不大;而客户端大多数时间CPU占用率为3%左右,只有刷新的瞬间为30%左右。

Jetty 6是java开发的一个Web服务器,它可以扩展大量同步连接,使用Java™语言的非阻塞I/O(java.nio)库并使用一个经过优化的输出缓冲架构。本文使用的是单个servlet线程,实际上需要在ThreadPool使用三个线程:Jetty服务器本身使用一个线程,另一线程运行HTTP连接器,侦听到来的请求,第三个线程执行servlet代码。其Jetty配置清单如下:

```
〈? xml version=“1.0”?〉...
〈Configure id=“Server” class=“org.mortbay.jetty.Server”〉
〈Set name=“ThreadPool”〈New class=“org.mortbay.thread.BoundedThreadPool”〉
  〈Set name=“minThreads”〉3〈/Set〉
  〈Set name=“lowThreads”〉0〈/Set〉
  〈Set name=“maxThreads”〉3〈/Set〉
〈/New〉〈/Set〉〈/Configure〉
```

DWR全称为Direct Web Remoting。DWR项目是在Apache许可下的一个开源的解决方案,它供给那些想要以一种简单的方式使用Ajax和XMLHttpRequest的开发者。DWR 2版本中最显著的特性就是Reverse Ajax,在服务器端通过Java异步调用JavaScript,将服务器端事件“推入”到客户机。客户端DWR代码透明地处理已建立的连接并解析响应,因此从开发人员的角度来看,事件是从服务器端用Java代码轻松地发布到客户机中。结合使用Comet模式和Jetty 6的Continuations API可以方便地在DWR 2中将Comet和Continuations与Reverse Ajax技术结合使用。DWR提供了一个控制器servlet,它将在Java对象之上直接转交客户机请求。为了使服务器端的应用程序被控制器加载执行,需要先配置DWR如下:

```
〈dwr〉〈allow〉〈create creator=“new” javascript=“Tracker” scope=“application”〉
  〈param name=“class” value=“developerworks.jetty 6.ReverseAjax”/〉
  〈/create〉〈/allow〉
〈/dwr〉
```

DWR经过配置之后可以感知ReverseAjax类的存在并构造Reverse Ajax对象。在ReverseAjax构造器中启动线程开始实时更新数据,代码如下:

```
public ReverseAjax() {
WebContext wctx=WebContextFactory.get();
sctx = serverContextFactory.get(wctx.getServletContext());
new Thread(this).start();
}。
```

代码首先获取ServerContext对象sctx,因为ServerContext接口提供了DWR的Reverse Ajax功能。ServerContext对象可以察觉到当前查看给定页面的所有Web客户机,并提供一个ScriptSession进行相互通信。ScriptSession用于从Java代码将JavaScript片段推入到客户机。

```
script.appendScript(“setSignalPos(”)
.appendData(strX).appendScript(“);”);
Collection〈ScriptSession〉 sessions = new HashSet〈ScriptSession〉();
sessions.addAll(sctx.getScriptSessionsByPage(RealTimeUrl));
for (ScriptSession session:sessions) {
    session.addScript(script); }
```

上述代码将调用客户端函数 setSignalPos(strX)的 JavaScript 语句加入到 script 对象中，再将 script 对象加入到 session 对象中，DWR 会自动将 ScriptSession 的内容发送到客户端。

DWR 的 Reverse Ajax 有三种不同机制。一为轮询方法；二为 piggyback；三为使用长期的、Comet 风格的连接。当运行在 Jetty 下时，DWR 能够自动检测并切换为使用 Contiuations，实现非阻塞 Comet。为此需要对 DwrServlet 的 web.xml 配置如下：

〈servlet〉…〈init-param〉
〈 param-name 〉 activeReverseAjaxEnabled 〈/param-name〉〈param-value〉true〈/param-value〉〈/init-param〉
〈init-param〉〈param-name〉initApplicationScopeCreatorsAtStartup〈/param-name〉
〈param-value〉true〈param-value〉
〈/init-param〉〈load-on-startup〉1〈/load-on-startup〉〈/servlet〉

第一个 servlet init-param，activeReverseAjaxEnabled 将激活 Comet 功能。第二个 initApplicationScopeCreators -AtStartup 通知 DWR 在应用程序启动时初始化 ReverseAjax。

客户端要建立 Comet 模式的连接，通过 dwr.engine.setActiveReverseAjax(true)给服务器发送建立请求。

4　结　论

(1)通过对 Ajax 技术和 Web 工作模式的分析研究发现，Web 2.0、Comet、Continuations 相结合改变了传统 Web 的交互模式，使服务器的角色由被动转为主动，同时该模式极大地降低了数据延迟，提高了系统的响应能力；Continuations 特性使得系统从根本上节约了资源，能利用有限的资源支持大量的用户。

(2)采用 Web 2.0、Comet、Continuations 相结合的技术实现了一个基于 Web 的实时监测系统，从实践上证明了结论(1)。该系统能够满足无闪烁、实时性能高的要求，并且在连接大量客户端时，服务器也只需消耗较少的 CPU 和内存资源。

(3)在将这种模式应用到电缆终端局部放电实时监测系统的过程中，根据局部放电三维谱图的特点，将三维谱图网格化并结合 div 技术模拟桌面应用程序局部刷新。这种解决方案获得了较好的用户体验，极大地减少了通信量，提高了系统实时性，具有实用价值。

(4)Web 2.0、Comet、Continuations 相结合的模式不仅适用于局部放电监测系统，在解决基于 B/S 的实时系统通信问题和页面显示问题上具有普适性。

参考文献：

[1] Paulson Linda Dailey. Building rich web applications with Ajax[J]. IEEE Computer Society, 2005: 14-17.

[2] Garrett Jesse James. Ajax: a new approach to web applications. [2006-09-26]. [EB/OL]. http://www.adaptivepath.com/ideas/essays/archives/000385.php.

[3] 吕林涛，万经华，周红芳. 基于 AJAX 的 Web 无刷新页面快速更新数据方法[J]. 计算机应用研究，2006(11)：199-200.
Lü Lin-tao, Wan Jing-hua, Zhou Hong-fang. Research of not refurbishing and updating data method in AJAX web application[J]. Application Research of Computers, 2006(11): 199-200.

[4] 陈郑伟，彭岩，庄力可. 基于 Ajax 的电子政务平台的研究与应用[J]. 计算机工程与应用，2007，43(5)：196-199.
Chen Zheng-wei, Peng Yan, Zhuang Li-ke. E-government software research and application based on Ajax technology[J]. Computer Engineering and Applications, 2007, 43(5): 196-199.

[5] Sonntag Michael. Ajax security in groupware[C]// Proceeding of the 32nd EuroMicro Conference on Software Engineering and Advanced Applications, IEEE: IEEE Computer Society, 2006.

[6] Wilkins Greg. Continuations. [2006-10-06]. [EB/OL]. http://docs.codehaus.org/display/JETTY/Continuations, 2007.

[7] Russell Alex. Comet: low latency data for the browser [2006-11-02]. [EB/OL]. http://alex.dojotoolkit.org/?p=545, 2006.

[8] Wilkins Greg. Ajax, Comet and Jetty. [2006-11-02]. [EB/OL]. http://www.webtide.com/downloads/whitePaperAjaxJetty.html, 2006.