*Long Polling*을 지원하는 자바기반의 *Push Engine*

박종은[O*], 권오진[*], 이홍창[*], 이명준[**]
울산대학교 컴퓨터·정보통신공학부
[O*]cjswowhddms, [*]{ojini5,myhyunii}@mail.ulsan.ac.kr, [**]mjlee@ulsan.ac.kr

A Java Push Engine Supporting Long Polling

Jong-Eun Park[O*], O-Jin Kwon[*], Hong-Chang Lee[*], Myungjoon-Lee[**]
School of Computer Engineering & Information Technology, University of Ulsan

요 약

인터넷에서 정보를 제공하는 일반적인 형태는 클라이언트의 요청에 따라 웹 서버가 응답하여 제공하는 방식이다. 이러한 전통적인 방식은 클라이언트의 명시적인 요청에 따라 정보가 제공되기 때문에 급변하는 정보를 클라이언트가 적시에 제공받기에는 불편함이 따른다. 이러한 문제를 해결하기 위하여 다양한 연구들이 시도되고 있으며, Server Push는 전통적인 방식의 문제점을 극복하기 위하여 웹 서버에서 정보가 변할 때마다 클라이언트로 적시에 정보를 제공하는 방식이다.

본 논문에서는 Server Push 기반의 웹 어플리케이션 개발을 효과적으로 지원하는 Java Push Engine에 대하여 기술한다. Java Push Engine은 자바 기반으로 구현되어 플랫폼에 독립적인 Push 어플리케이션 개발을 지원하며, 다중화 통신을 지원하여 많은 클라이언트의 요청을 동시에 처리 할 수 있다. 또한 Java Push Engine과 함께 효과적인 비동기 통신을 수행하는 Push 클라이언트를 개발하기 위한 Ajax 기반의 Java Push Engine Java 스크립트에 대해서도 기술한다.

1. 서 론

인터넷을 통하여 사용자들은 정보를 검색하고 원하는 정보를 받을 수 있다. 인터넷에서 정보를 제공하는 일반적인 형태는 클라이언트의 요청에 따라 서버가 응답하여 제공하는 방식이다. 서버는 클라이언트의 요청을 상태별로 다양한 응답을 할 수 있는 여러 메소드를 제공한다. 인터넷에는 많은 사용자들을 위한 다양한 애플리케이션이 등장하고 있으며, 애플리케이션은 변화하는 사용자의 요구사항에 따라 여러 맞춤형 서비스들을 제공하고 있다.

인터넷의 서버/클라이언트 데이터 전송 방식은 클라이언트가 원하는 정보를 요청하여야 서버는 클라이언트가 원하는 정보를 응답한다. 이 경우 클라이언트가 지속적으로 원하는 정보를 제공 받으려면 서버에게 주기적으로 요청을 하여야 한다. 이러한 전송방식은 서버가 수동적으로 정보를 제공하는 애플리케이션에는 적합하지만, 이를 이용하여 서버가 능동적으로 정보를 제공하는 애플리케이션을 개발할 경우 많은 어려움이 따른다. 오늘날의 정보화 시대는 금융정보, 플랜트와 선박의 모니터링 정보, 관심 분야의 뉴스와 같은 정보를 신속하고도 효과적으로 제공받는 방법이 필요하다.

Server Push[1,2]는 사용자의 요청에 의해서가 아니라, 웹상의 정보 전달이 서버에 의해 개시되는 것이다. Server Push는 서버에서 정보가 변경되면 이벤트가 발생하여 연결된 클라이언트에게 능동적으로 변경된 정보를 제공한다. 클라이언트는 즉각적인 정보를 받음과 동시에 확인이 가능하고, 클라이언트의 명시적인 요청이 없어도 서버의 정보를 쉽게 제공 받는다. 정보를 효과적으로 제공받는 방법에 대하여 다양한 연구가 시도 되었다. 그 중, 사용자가 마치 자신의 PC가 새로운 정보를 갱신해서 보는 것처럼 생각하게 되는 Polling[3]방식이 있다. 이 방식은 클라이언트가 지속적인 요청을 하여 서버가 계속적인 응답을 해주는 방식으로 이 경우는 서버의 부하가 생긴다. 그래서 서버에서 Push를 해주는 Server Push 기술이 개발되었으며 대표적으로 HTTP Streaming[4], Long Polling[5]방식이 있다.

Weelya[6]사의 Ajax Push Engine[7]은 Server Push 기술을 지원하는 대표적인 오픈 소스 소프트웨어이다. 그러나 C언어로 작성 되어있고, Unix 계열의 커널에서 동작되어 지는 E-poll Driven[8]방식의 다중화를 사용하여 리눅스 기반에서만 서버 구축을 할 수 있는 단점이 있다.

본 논문에서는 기존에 개발된 Ajax Push Engine의 단점을 보완하기 위하여 개발된 자바기반의 Java Push Engine에 대해 기술한다. Java Push Engine은 Push 기술을 이용한 다양한 사용자의 웹 어플리케이션을 개발을 쉽게 할 수 있는 확장성 있는 다양한 메소드를 제공한다. 또한 클라이언트는 제공되는 스크립트 파일의 Ajax 함수를 이용하여 효과적으로 Push 서버와 비동기적인 통신이 가능한 웹 어플리케이션을 개발 할 수 있다.

본 논문의 구성은 다음과 같다. 서론에 이어 2장에서는 Push 서비스를 구축하는 기법과 관련 연구에 대하여 살펴보며, 3장에서는 Java Push Engine의 정책과 구조를 보여준다. 4장에서는 3장에서 개발된 Java Push Engine을 이용하여 Push 서비스를 제공하는 간단한 웹 어플리케이션의 구현을 보여준다. 마지막으로 5장에서는 결론을 살펴보도록 한다.

2. 관련 연구

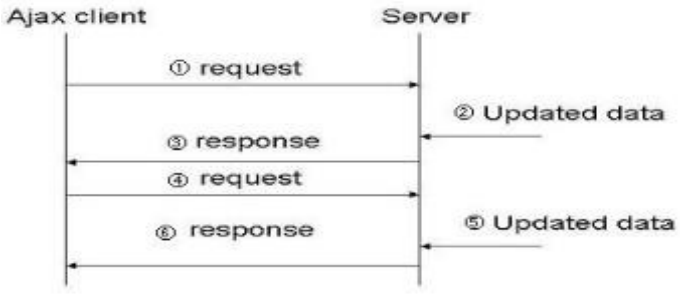
2-1. Long Polling

Push 기술 혹은 Server Push는 서버에 연결된 다수의 클라이언트들이 능동적으로 보내는 정보를 명시적인 요청이 없더라도 서버가 전송 받을 수 있는 인터넷 통신 방법으로서 [표 1]과 같이 설명할 수 있다.

[표 1] HTTP에서 Push를 구현하기 위한 기법들

기법	특징
Polling	· 클라이언트가 주기적으로 서버에게 요청을 보내어 서버의 변경된 정보를 즉시 응답받는 방법. · 서버 구현이 용이. · 네트워크 및 서버/클라이언트에 부하가 심함
Long Polling	· AJAX 등 비동기 프레임워크에서 사용 · 서버는 클라이언트의 요청을 받으면 연결을 유지하고 정보가 변경되는 이벤트가 발생하면 응답 후 HTTP 트랜잭션 종료. · 클라이언트는 응답을 받으면 재요청.
Streaming	· 실시간 스트리밍처럼 서버와 클라이언트 간의 연결을 유지하며 정보가 변경되는 이벤트가 발생하면 HTTP Chunked[9]방식으로 데이터를 전송 · 구현과 예외처리가 어려우며 호환성이 떨어짐

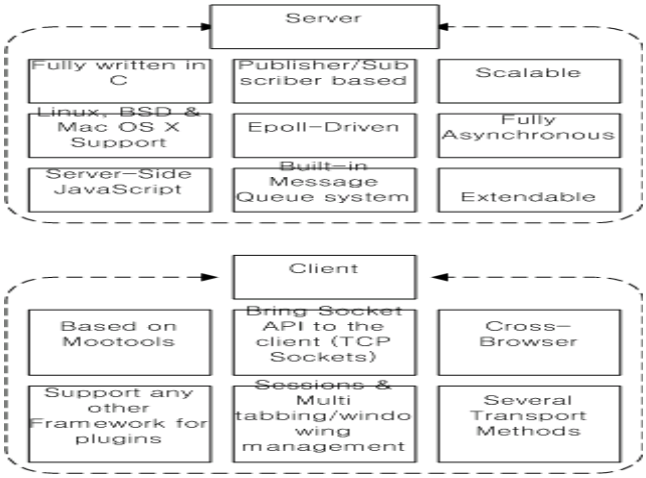
Long Polling은 Ajax와 Flex의 비동기 통신을 지원하는 기술의 보급으로 Polling의 네트워크 및 서버/클라이언트 부하 문제와 Streaming의 구현과 호환성 문제의 단점을 극복하기 위해 등장한 Push기술 적용 기법이다. [그림 1]은 서버가 클라이언트가 접속되면 전송해줄 데이터가 있을 때까지 연결 상태를 유지하며 클라이언트가 데이터를 전송받으면 서버와 재연결되는 통신 흐름을 보여준다.



[그림 1] Long Polling기법의 통신 흐름

2-2. Ajax Push Engine

Ajax Push Engine은 Ajax 클라이언트와 Mootools[10] 라이브러리로 구성된 클라이언트와 통신하며 분산된 정보를 연결하거나 데이터를 클라이언트에 능동적으로 Push 하는 기능을 제공한다. 채널(Channel)이라는 서버와 클라이언트와의 데이터 전송의 통신 경로를 제공하며 Long Polling을 기반으로 하는 대표적인 Push 엔진이다. 그리고 클라이언트들의 다양한 요구를 빠르게 처리하기 위한 다중화 통신 전략으로 E-poll driven방식을 사용한다. E-poll driven방식은 Unix계열 운영체제에서 어플리케이션 개발에 지원된다. Ajax Push Engine의 서버와 클라이언트는 다음과 같은 특징을 가진다.



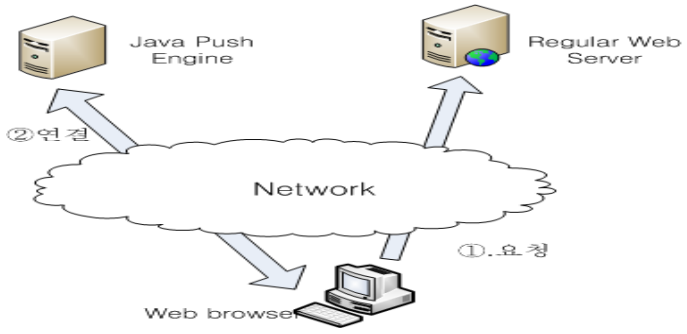
[그림 2] Ajax Push Engine 서버와 클라이언트 특징

3. Java Push Engine 정책과 구조

본 장에서는 플랫폼에 독립적인 Java 언어 기반의 Java Push Engine의 기본 정책과 클라이언트의 관리 체계 그리고 서버와 클라이언트의 구조에 대해서 기술한다.

3.1. Java Push Engine 정책

Java Push Engine은 [그림 3]과 같이 동작 한다.



[그림 3] Java Push Engine의 연결 과정

[그림 3]에서 Java Push Engine을 이용한 웹 서버에 ①사용자가 접속 요청을 하게 되면 웹을 이용하는 사용자는 Java Push Engine 클라이언트가 ②Java Push Engine과 연결이 되어 Long Polling방식의 Server Push 서비스를 받게 된다. [그림 3]과 같은 연결 과정은 Long polling 서비스를 제공하기 위한 방법이다. 이와 같은 연결 방법은 클라이언트와 Push 서버간의 연

결 설정 후 Push 서버에서 클라이언트를 관리할 수 있는 장점을 가진다.
또한 Server Push 기반의 웹 어플리케이션의 용이한 개발을 위한 메소드와 효과적인 채널 관리를 제공한다.

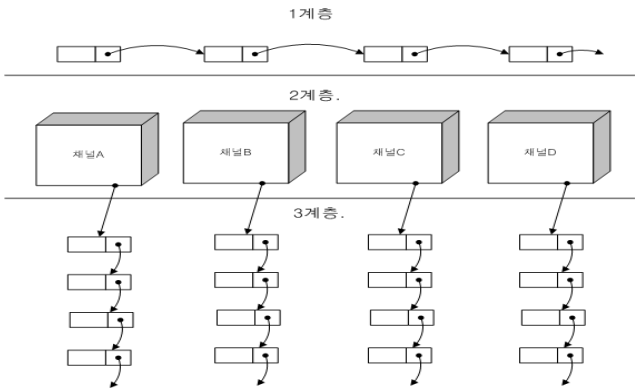
3.2. Java Push Engine 채널과 클라이언트 관리 방법

Java Push Engine의 채널관리는 서버와 독립적으로 존재하는 ChannelAT.xml 파일을 통하여 이루어지며, Push Server는 xml 파일의 수정을 통하여 통신 채널을 변경하며 서버는 이를 즉각 반영하도록 구성되어 있다. Java Push Engine의 채널의 속성은 [표 2]와 같다.

[표 2]Java Push Engein 채널의 속성

이름	목적
title	채널 명
max	해당 채널의 최대 접속 인원 수
isopen	채널의 열림과 닫힘 상태
isgroup	그룹을 위한 추가 적인 기능
type	공개/ 비공개 채널 설정
password	비공개 채널의 접속 비밀번호 설정

개인과 그룹을 위한 채널을 생성 할 수 있으며 공개 채널과 비공개 채널을 생성하고, 비공개 채널의 경우 클라이언트가 채널에 접속 시 비공개채널의 password를 확인하여 클라이언트의 접속을 확인한다. 또한 접속하는 클라이언트를 효율적으로 관리하기 위하여 그림[4]와 같은 구조를 채택하고 있다.



[그림 4] Java Push Engien의 채널관리 구조

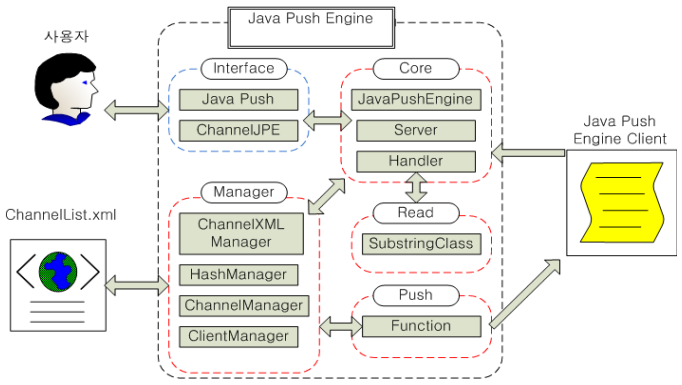
[그림 4]는 채널과 접속된 클라이언트를 관리하기 위하여 3계층 구조 사용을 보여준다. 1계층은 각각의 채널을 관리하기 위한 연결형 구조로 구성된다. 2계층은 xml파일의 채널의 속성 값을 저장하는 테이블 형식으로 구성된다. 3계층은 해당 채널에 접속된 클라이언트를 관리하기 위한 연결형 구조로 구성된다. 클라이언트가 접속 하면 1계층에서 해당 채널을 불러와서 해당채널의 종류와 상태를 확인 한 후 접속이 가능하면 클라이언트를 관리하는 3계층의 연결형 구조에 클라이언트를 삽입한다. 이와 같은 구조로 서버의 채널에 오류가 발생하면 해당 채널에 접속되어 있는 클라이언트들에게 에러 메시지를 전달한다.

3.3. Java Push Engine 서버 구조

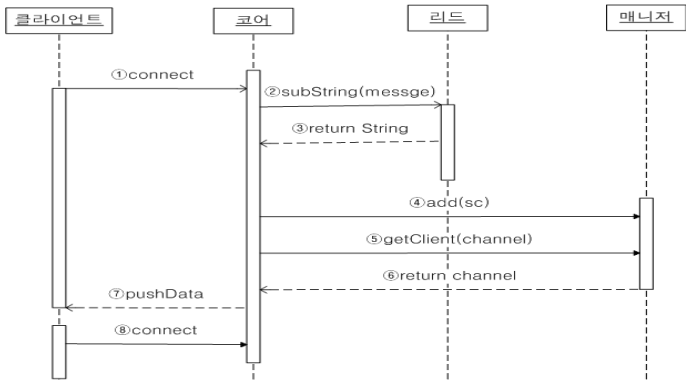
Java Push Engine의 서버는 클라이언트와 서버간의 Long Polling 통신을 위한 Core 패키지와 유동성 있는 채널관리를 위한 Manager 패키지, 데이터 전송을 위한 Push 패키지, 클라이언트로부터 전송되는 정보를 분석하기 위한 Read 패키지와 웹 어플리케이션 개발을 위한 Interface 패키지로 구성된다. 각각의 패키지와 클래스의 기능은 [표 3]과 같다.

[표 3]Java Push Engien의 주요 클래스와 기능

패키지	클래스	기능
Core	Handler	요청을 처리
	JavaPushEngine	동작 방법 제공
	Server	서버를 시작
Interface	ChannelJPE	채널 관리 제공
	JavaPush	Push 방법 제공
Manager	BlackListXMLManager	접근 제어
	ChannelManager	1계층 관리
	ChannelXML	채널을 구성
	ClientManager	3계층 관리
	HashManager	2계층 관리
Push	Function	데이터 Push
Read	SubstringClass	요청을 분석



[그림 5] Java Push Engine의 서버 구조도

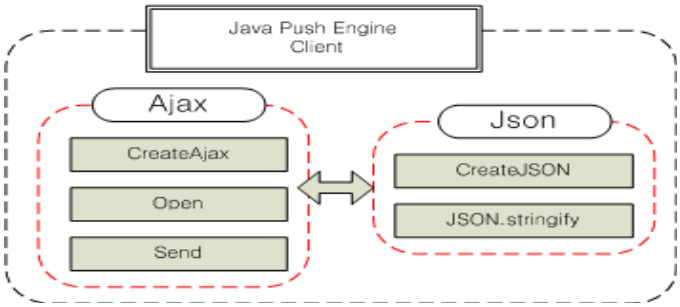


[그림 6] 서버의 데이터 흐름 순서도

[그림 6]은 클라이언트가 서버에 접속 시 서버의 동작을 보여준다. Core 패키지에서는 클라이언트의 접속을 확인한다. Read 패키지에서는 클라이언트가 요청하는 것을 분석한다. Manager 패키지는 클라이언트의 정보를 저장한다. Core 패키지는 클라이언트로 전송할 데이터가 생성되면 Manager 패키지로부터 해당 채널의 클라이언트의 정보를 받는다. 전송받은 데이터는 해당 채널의 클라이언트에게 전송한다. 위 과정은 클라이언트의 요구를 확인하고, 해당 채널의 클라이언트를 관리하고, 전송할 데이터 생성 시 서버에서 클라이언트로 데이터를 전송한다.

3.3. Java Push Engine 클라이언트 구조

Java Push Engine과 Long Polling 통신 기술을 구현하기 위하여 클라이언트는 Ajax와 JSON[11]기반의 기술을 사용하여 서버와 효과적인 통신을 한다. 클라이언트는 서버의 특정 채널에 구독(subscribe)을 하게 되고 다른 클라이언트나 서버가 해당 채널에 정보를 배포(publish)하면 변경된 데이터는 구독한 클라이언트에게 자동으로 전달된다.



[그림 7] Java Push Engine의 클라이언트 구조도

[그림 7]은 Java Push Engine 클라이언트의 기본적인 구조를 보여준다. 웹 어플리케이션 개발자는 클라이언트 내부에서 Ajax 객체를 생성한다. Ajax 객체는 JSON기반의 기술을 사용하여 서버로 보내는 메시지를 생성한다. Send 객체는 서버와의 연결을 시도한다. 클라이언트의 요청은 채널 데이터를 포함하여 전달하고, 서버는 데이터를 분류하여 클라이언트를 접속 채널로 구분한다. 일반적인 Push 기술의 모델에서 사용할 수 있는 함수를 다음 [표 4]로 정의 한다.

[표 4] Publish/Subscribe 모델 함수

이름	설명
ChannelEnter(inChannel,location)	· 특정 채널의 정보를 구독하기 위하여 서버에 연결을 요청하는 메소드다. inChannel은 접속하려는 채널 명이며, location은 서버의 정보를 제공 받을 위치.
sendMsg(outChannel,msg)	· 채널에서는 특정 채널(outChannel)로 메시지를(msg)를 전달하는 함수.

4. Java Push Engine 의 활용

Java Push Engine에서 서버의 Push 기술을 이용하려면 [표 5]에서와 같은 클래스를 이용하여야 한다.

[표 5] Push 기술 구현을 위한 객체

클래스	메소드	목적
JavaPushEngine	· basicServerStartLinux() · basicServerStartWindowsXP	Java Push Engine을 가동 시키는 클래스며, 메소드는 운영제제에 따라 선택.
ChannelJPE	· createCh() · deleteCh() · modifyCh() · getChannel()	Channel.XML를 변경하기 위한 추가, 삭제, 수정 기능 제공.
JavaPush	· push() · getClient()	원하는 특정 채널로 원하는 데이터를 Push.

[표 5]는 JavaPushEngine 클래스의 메소드는 OS에 따라 선택할 수 있다. 윈도우xp환경과 리눅스 환경에 따라 동작 하도록 하는 메소드를 제공 한다. ChannelJPE 클래스의 메소드는 서버에서 관리하는 채널에 편의를 제공한다. 메소드는 채널명을 사용하여 생성, 삭제, 수정, 정보받기 등의 기능을 수행한다.

[그림 8]은 Java Push Engine을 이용하여 현재의 시간을 특정 채널에 전송하는 소스 코드의 일부를 보여준다.

```
public class Test1{
 private JavaPushEngine jpe = new JavaPushEngine();
 private JavaPush jp = new JavaPush(jpe.cXML);
 public void testt() {
  jpe.basicServerStartWindowsXP(7268,
  "203.250.77.134");//
  testMethod();
  }
 public void testMethod(){
  Thread thread = new Thread(){
   public void run(){
    while(true){
     currentDate = new Date();
     jp.push(currentDate.toString(),"ChannelC");
     ...
```

[그림 8] 활용되는 서버 소스 코드 일부

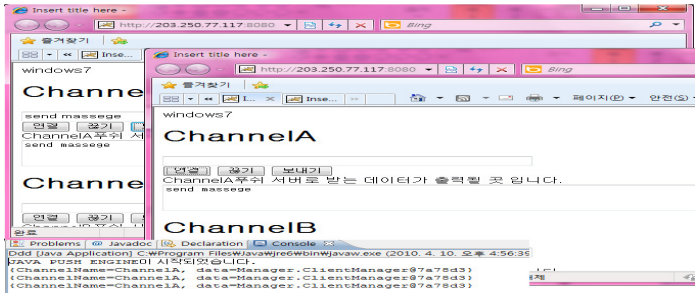
[그림 9]는 Publish/Subscribe 모델을 지원하는 함수를 활용하여 웹 Subscribe 기능을 수행하는 간단한 웹 어플리케이션의 소스 코드로서, 서버로부터 현재의 시간을 클라이언트의 특정 텍스트에 제공받는 웹 어플리케이션의 소스 코드의 일부를 보여준다. 클라이언트는 “ChannelC”인 채널에 접속하여 웹 컴포넌트의 id 값이 time인 위치에 출력한다. Java Push Engine은 스크립

트 함수의 지원으로, [그림 9]와 같은 간단한 웹 어플리케이션부터 복잡한 웹 어플리케이션의 개발도 지원한다.

```
...
<script type="text/javaScript"
src="../../config.js">
src="../../client/client.js">
</script>
...
<h1>현재 시간은</h1>
<textarea id="time" style="width:30%;
height:30px;"></textarea>
<script type="text/javaScript">
ChannelEnter("ChannelC","time");
</script>
```

[그림 9] 클라이언트의 소스 코드 일부

[그림 10]은 Java Push Engine을 이용한 Push 기반의 웹 어플리케이션들의 모습을 보여준다.



[그림 10] Java Push Engine을 이용한 웹 어플리케이션

5. 결 론

본 논문에서는 Long Polling Push기술을 이용하는 Java Push Engine의 서버와 클라이언트의 개발에 대하여 기술하였다. Java Push Engine은 Java로 개발되어 Ajax Push Engine과는 다르게 플랫폼에 독립적이며, 제공되는 메소드를 통하여 애플리케이션의 목적에 따라 쉽게 정보 제공방법을 구현하여 적용할 수 있는 장점이 있다. 클라이언트의 자바 스크립트 파일은 Ajax 객체의 사용으로 서버와의 비동기 통신을 제공한다.

향후 보다 다양한 서비스가 가능하도록 서비스 종류를 확장하고, 서버의 성능을 높일 수 있는 기법의 개발이 필요할 것으로 생각된다.

6. 참고 문헌

[1] Shishir Gundavaram, "CGI Programming on the World Wide Web," 1st Edition, March 1996
[2] "http://en.wikipedia.org/wiki/Push_technology," Push Technology
[3] http://en.wikipedia.org/wiki/Polling_(computer_science) Polling (computer science)
[4] "http://ajaxpatterns.org/HTTP_Streaming," HTTP Streaming
[5] "http://en.wikipedia.org/wiki/Push_technology, "Long polling
[6] "http://www.weelya.com,“ PROJECT
[7] "http://www.ape-project.org," What's APE?
[8] "http://linux.die.net/man/4/epoll", "http://java.sun.com/javase/6/docs/technotes/guides/io/enhancements.html," java.nio
[9] "http://en.wikipedia.org/wiki/Chunked_transfer_encoding." Chunked transfer encoding
[10] "http://mootools.net/docs/core," a compact javascript framework
[11] "http://www.json.org," Introductin JSON