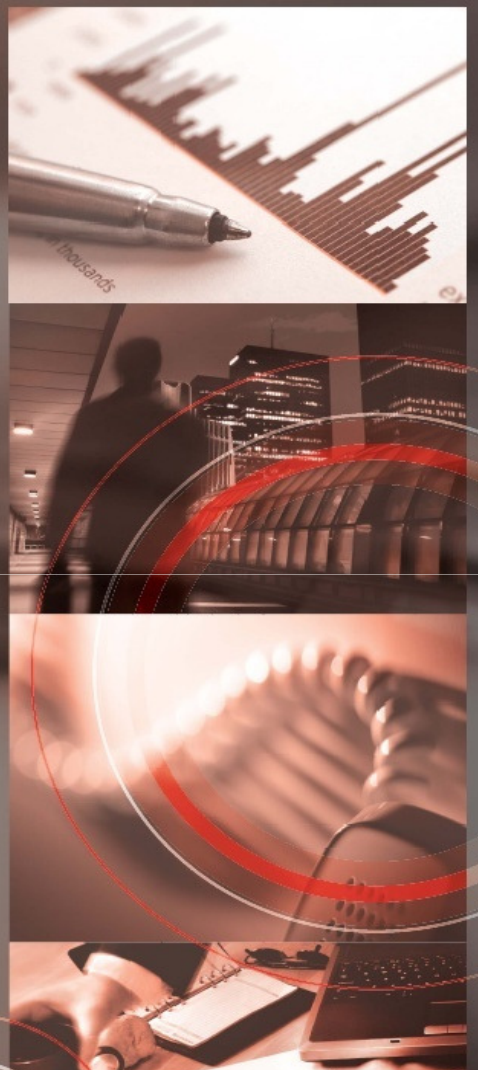
# Performance Testing Web 2.0




Business Partner hp invent

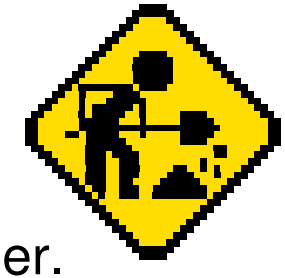
***Stuart Moncrieff (Load Testing Guru)***

***www.jds.net.au / www.myloadtest.com***

jds australia

# Foundations of Web 2.0 (a history lesson)

- 1993
  - The National Center for Supercomputing Applications releases Mosaic, which would become the first popular graphical web browser.



- 1995
  - Microsoft licenses Mosaic from Spyglass as the basis for Internet Explorer 1.0. Spyglass signs a deal to collect a royalty for every copy of the browser software that Microsoft sells. Microsoft gives the browser away for free.
  - Netscape Navigator 2.0 released with support for **JavaScript** (invented by Netscape) and **Java Applets**. Java immediately develops a reputation as a “slow” language as Java applets perform poorly even on Intel’s brand new 200 MHz Pentium Pro CPU.

# Foundations of Web 2.0 (a history lesson)

- 1996
  - July: Hotmail is launched, providing an example of a desktop application being replaced by a browser-based application.
  - August: Internet Explorer 3.0 released, with support for **ActiveX**, frames and **JScript**.
  - November: Macromedia **Flash** 1.0 released.
- 1998
  - **XML** 1.0 becomes a W3C recommendation (like HTML, it is a recommendation, not a standard).


- 1999
  - March: IE 5.0 is released with the first implementation of **XMLHttpRequest()**, which allows HTTP requests to be made without refreshing the entire web page. Microsoft intends this to be used by the Outlook 2000 web client.
  - July: **RSS** 0.91 released, makes content syndication easy

# Foundations of Web 2.0 (a history lesson)

- 2000
  - August: **Flash** Player 5 is released, supporting **ActionScript** 1.0.
  - September: Web Services Description Language (**WSDL**) 1.0 developed by IBM, Microsoft, and Ariba to describe web services for their **SOAP** toolkit.
- 2005
  - February: **Google Maps** released, providing a great example of Ajax in action – as the user drags the map, the grid squares are downloaded from the server and inserted into the page; the page is not reloaded.
  - April: First video uploaded to YouTube, which uses Flash to deliver streaming video to users without requiring them to install a separate video codec or player.

# Foundations of Web 2.0 (a history lesson)

- 2006
  - May: Google Releases the **Google Web Toolkit** (GWT), allowing developers to develop applications in Java, and then cross-compile it to JavaScript to run in a web browser.
  - July: RFC 4627 released, defining **JSON** (JavaScript Object Notation), an alternative to SOAP.
- 2007
  - April: Microsoft releases **Silverlight** as an alternative to Flash.
- 2008
  - December: Sun releases **JavaFX** 1.0 as an alternative to Flash.


# What is Web 2.0?*


- Web 2.0 tag cloud

* A buzzword coined by Tim O'Reilly in 2004 :)

# Web 2.0 – things a load tester cares about

- Rich Client – web browser as an application platform
  - Ajax
  - Flash, Silverlight, JavaFX, Java Applets, ActiveX components
  - JavaScript-based toolkits and widgets (Dojo, jQuery, GWT, Pyjamas)
- Consumable API
  - Easy syndication (RSS, Atom)
  - Web Services (REST, SOAP/XML, JSON)

# Web 2.0 – Consequences for load testers

- Rich client
  - It becomes far more difficult to create automated scripts for load testing. Consequences for scripting time, and required skill level.
  - End-user response times can now include significant client-side time, which load testing tools do not measure.
- Consumable API
  - Usage patterns more unpredictable due to automated requests to your API. Badly behaved consumer applications may cause problems.
    - Example: popular Australian e-commerce site. High website usage could cause outage for call centre.
    - Limit exposure through system architecture. Define upper limit through policy enforcement
  - Opportunity to test/tune components of “composite” applications separately before testing the system as a whole.
    - Example: Amazon.com’s redesign to a SOA architecture

# Web load test scoping questions

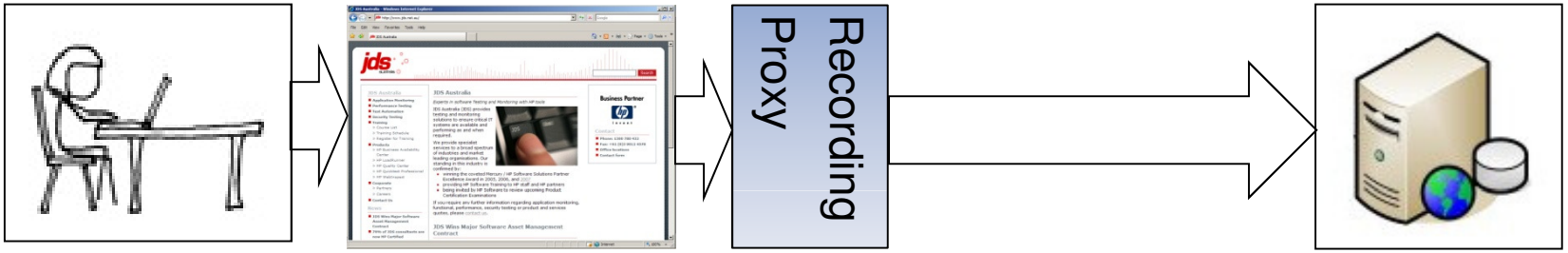
- Standard questions
  - Number of business processes in scope
  - Expected number of concurrent users and peak hour transaction rate for each business process
  - System architecture (how many servers, what is on each server, what software components)
- New question
  - *Does it use Ajax or rich client components like Java applets, or ActiveX objects?*
    (will require a short Proof-of-Concept before providing an estimate)

# Top 12 .com.au websites (by traffic)

| Site | Ajax? | API? | Non-static Flash? |
|---|---|---|---|
| google.com.au | Yes | Public | |
| ebay.com.au | Yes | Public | |
| ninemsn.com.au | Yes | | Video streams |
| news.com.au | Yes | RSS | Video streams |
| realestate.com.au | | | |
| smh.com.au | Yes | | Video streams |
| commbank.com.au | | | |
| bom.gov.au | | | |
| abc.net.au | | RSS | Video/audio streams, navigation |
| theage.com.au | Yes | | Video streams |
| seek.com.au | Yes | Private | Yes |
| bigpond.com | Yes | | Video streams, some navigation |

# How load testing tools work: Recording

- Record HTTP traffic as user steps through a business process.

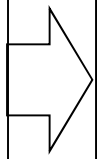



- physical Interface
  - mouse clicks
  - keystrokes
- browser events
  - client-side code
- HTTP
- TCP
  - IP
  - Ethernet
- HTTP
  - server-side code

# How load testing tools work: Generation

- Script is generated from recording log
- Provides a programming API that simplifies creation of HTTP requests
- Provides a scripting language for logic/flow control and modification of HTTP requests

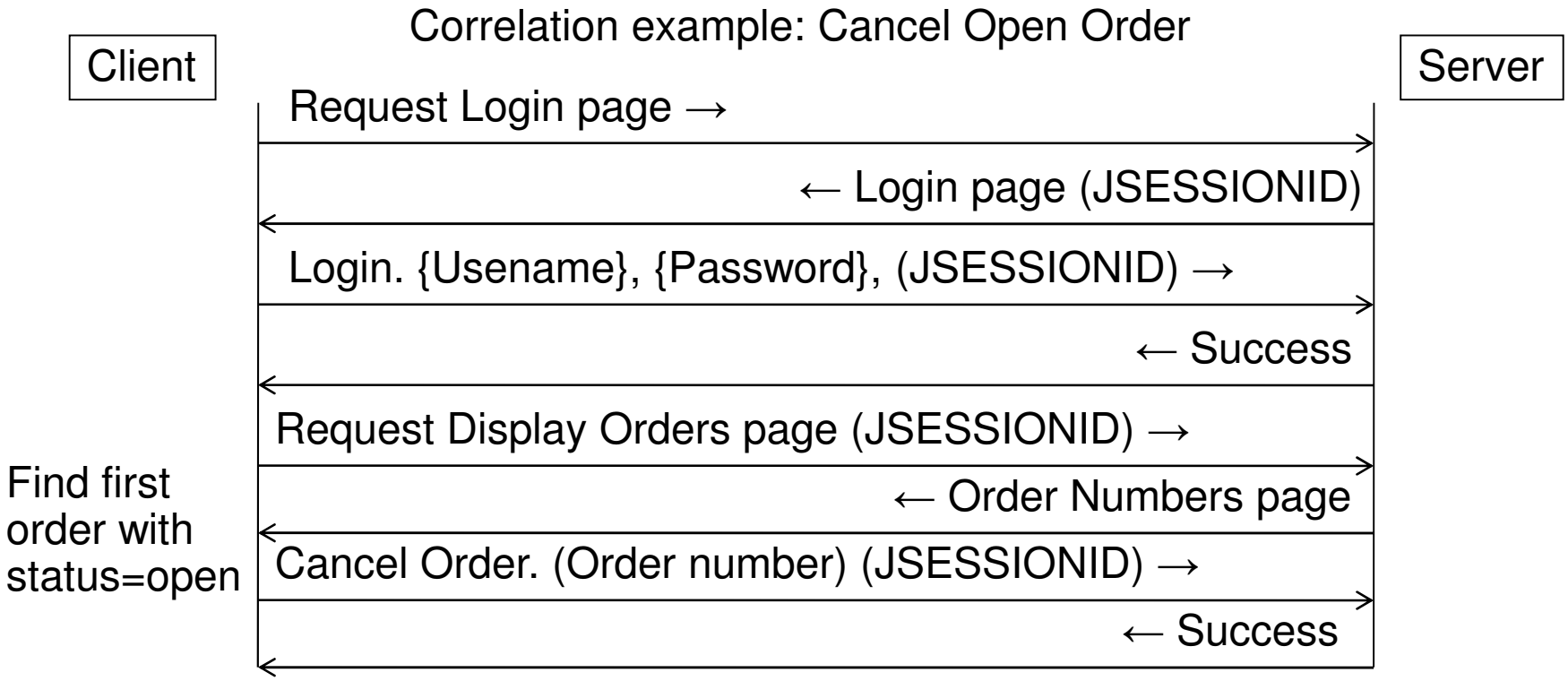
```
POST /login.jsp HTTP/1.1
Accept: */*
Referer: https://www.jds.net.au/
Accept-Language: en-us
Content-Type: application/x-www-form-urlencoded
UA-CPU: x86
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0)
Host: www.jds.net.au
Content-Length: 39
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: JSESSIONID=(J2EE7571700)ID0483470195228

userId=testuser&password=qwerty123
```



```
web_submit_data("login.jsp",
    "Action=https://www.jds.net.au/login.jsp",
    "Method=POST",
    "Snapshot=t1.inf",
    "Mode=HTML",
    ITEMDATA,
    "Name=userId", "Value=testuser", ENDITEM,
    "Name=password", "Value=qwerty123", ENDITEM,
    LAST);
```

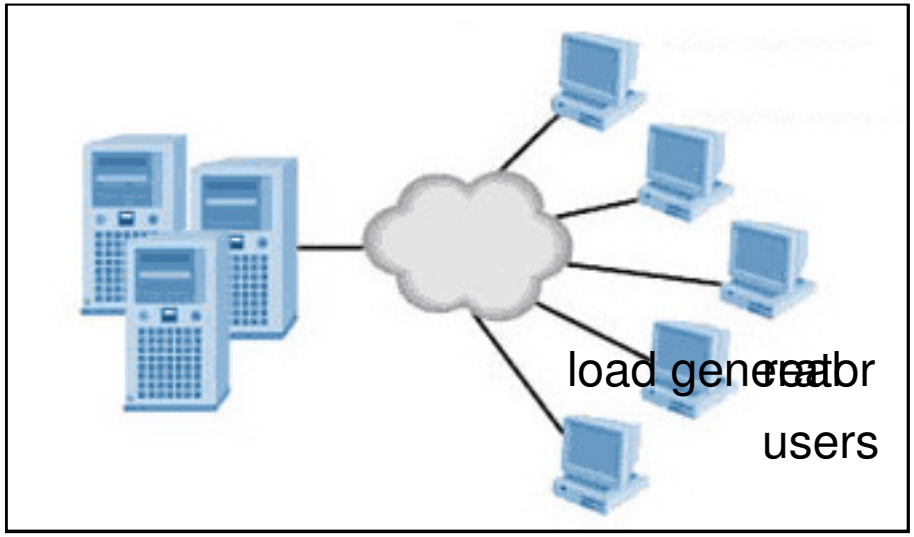
# How load testing tools work: Modification

- Parameterise inputs
- Correlation – capture values from server response to be used in next HTTP request

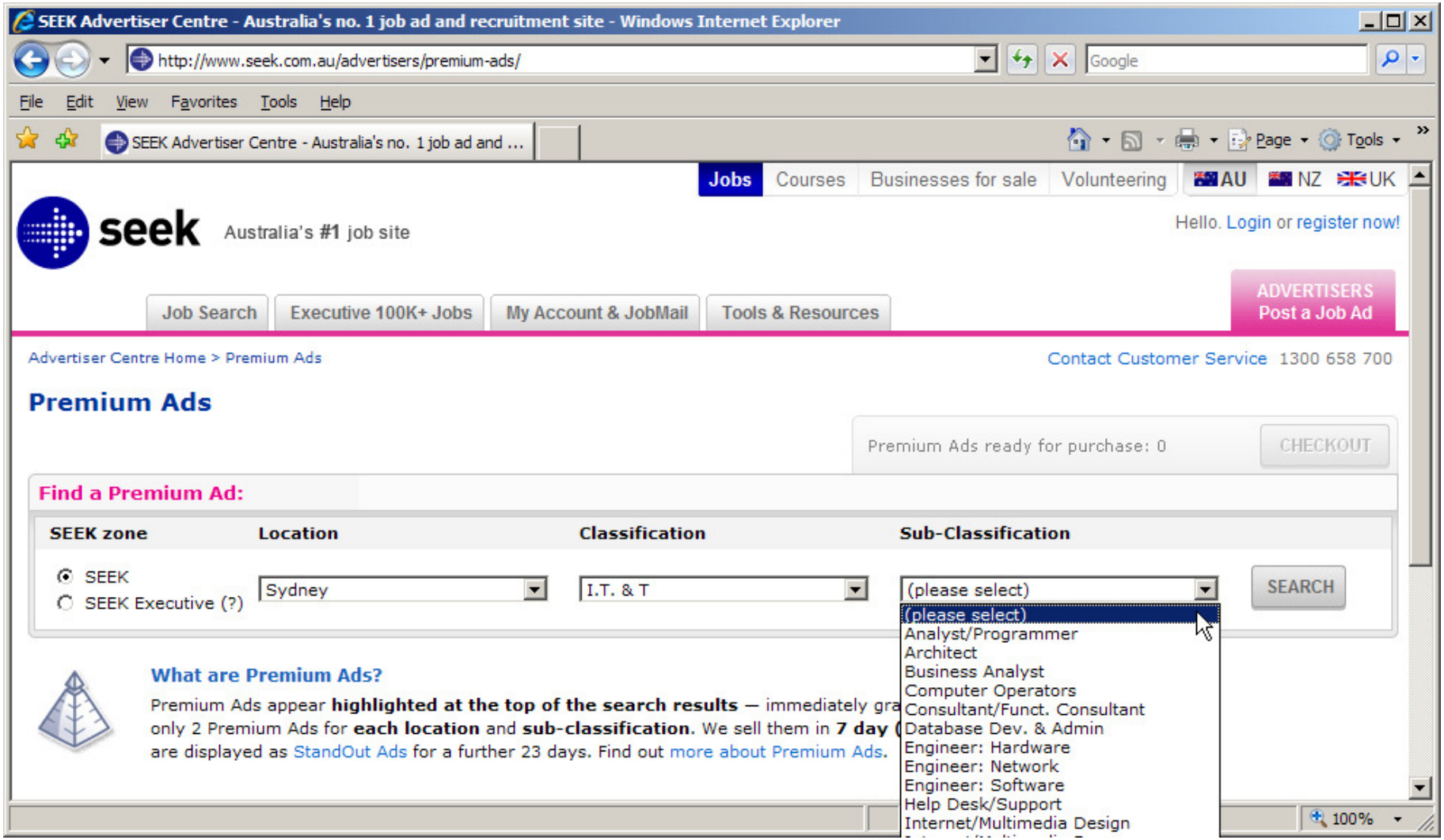


# How load testing tools work: Replay

- Replay engine sends and receives HTTP traffic (based on script).
- Browser is not involved. Nothing displayed on screen.
- Hundreds of virtual users per computer.
- Response times measure server and network time, not client-side performance.



# Ajax example: seek.com.au



# Ajax example: seek.com.au (JSON)

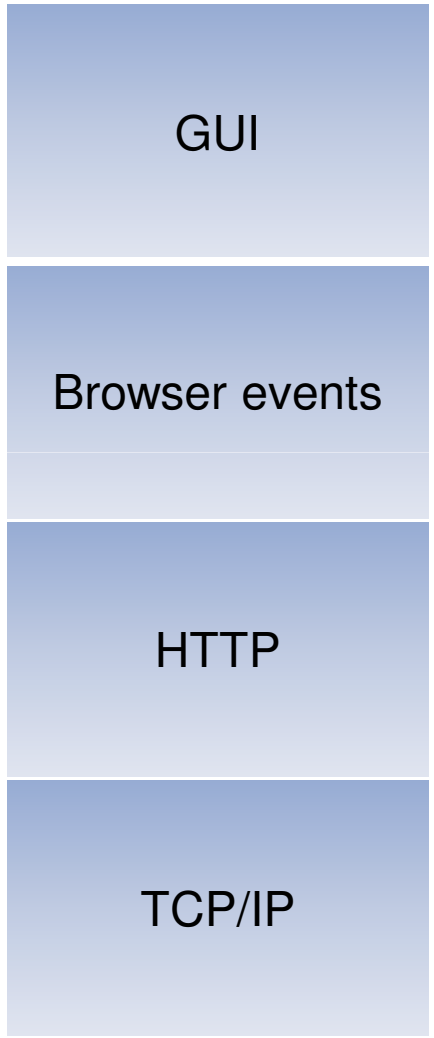

```
{
  "theRequest":{
    "__type":"SEEK.Services.Client.SaveCartRequest",
    "Items":[
      {
        "__type":"SEEK.Services.ResultItem",
        "Key":{
          "__type":"SEEK.Services.ItemCode",
          "Code":"ONHFXCEY-69597554555555-75597554780404-6557-6755-6857-6N2N5"
        },
        "HasAlert":false,
        "PriceBand":1,
        "Status":1,
        "Description":"",
        "JobID":0,
        "InCart":true,
        "FromDate":"2009/04/14",
        "ToDate":"2009/04/20",
        "InventoryCode":null,
        "UserOwnsJobContent":false,
        "ReferenceNumber":"",
        "Price":220
      }
    ]
  }
}
```

# Why is this a problem?

- Complex processing on the client side can make correlation really difficult
  - In Seek example, it is necessary to re-implement client-side processing of JSON objects.
  - This increases the script development phase of your performance testing cycle.
  - This restricts the number of people who are able to create scripts (due to higher level of technical ability required).

# Next-generation load testing tools

- Moving up the software stack.
- From HTTP level to browser event level
- A “high level” script that executes client-side code and triggers browser events e.g. onLoad, onClick
- Does not render GUI, so many virtual users can be run on a single load generator.
- But requires more memory and CPU per virtual user than an HTTP-based script.



# Current vs. next-generation tools

```
web_custom_request("SaveNewRowsToState",
    "URL=http://www.seek.com.au/advertisers/Services/ViewPremiumAds.asmx/SaveNewRowsToState",
    "Method=POST",
    "TargetFrame=",
    "Resource=0",
    "RecContentType=application/json",
    "Referer=http://www.seek.com.au/advertisers/premium-ads/",
    "Snapshot=t4.inf",
    "Mode=HTML",
    "EncType=application/json; charset=utf-8",
    "Body={\"newRows\":[{\"__type\":\"SEEK.Services.Client.SearchResult\",\"SearchResultID\":\"1002$1200$1302\",\"Result
    "ReferenceNumber\":\"\",\"Price\":220},{\"__type\":\"SEEK.Services.ResultItem\",\"Key\":{\"__type\":\"SEEK.Services.
    "__type\":\"SEEK.Services.ItemCode\",\"Code\":\"ONHFXCEY-62587554555555-78587554780404-6557-6755-6857-67NOR\"},\"Has
    "ONHFXCEY-79587554555555-85587554780404-6557-6755-6857-68NQ3\"},\"HasAlert\":false,\"PriceBand\":1,\"Status\":1,\"De
    ":1,\"Status\":1,\"Description\":\"\",\"JobID\":0,\"InCart\":false,\"FromDate\":\"2009/03/31\",\"ToDate\":\"2009/04/
    "ToDate\":\"2009/04/13\",\"InventoryCode\":null,\"UserOwnsJobContent\":false,\"ReferenceNumber\":\"\",\"Price\":220}
    "\"\",\"Price\":220},{\"__type\":\"SEEK.Services.ResultItem\",\"Key\":{\"__type\":\"SEEK.Services.ItemCode\",\"Code\
    "SEEK.Services.ItemCode\",\"Code\":\"ONHFXCEY-73597554555555-59507554780404-6557-6755-6857-6R875\"},\"HasAlert\":fal
    "\"},\"HasAlert\":false,\"PriceBand\":1,\"Status\":1,\"Description\":\"\",\"JobID\":0,\"InCart\":false,\"FromDate\":
    ":false,\"FromDate\":\"2009/05/12\",\"ToDate\":\"2009/05/18\",\"InventoryCode\":null,\"UserOwnsJobContent\":false,\"
    "UserOwnsJobContent\":false,\"ReferenceNumber\":\"\",\"Price\":220},{\"__type\":\"SEEK.Services.ResultItem\",\"Key\"
    "SEEK.Services.ResultItem\",\"Key\":{\"__type\":\"SEEK.Services.ItemCode\",\"Code\":\"ONHFXCEY-57517554555555-535175
    "ONHFXCEY-54517554555555-60517554780404-6557-6755-6857-6949R\"},\"HasAlert\":false,\"PriceBand\":1,\"Status\":1,\"De
    ":1,\"Status\":1,\"Description\":\"\",\"JobID\":0,\"InCart\":false,\"FromDate\":\"2009/06/16\",\"ToDate\":\"2009/06/
    "ToDate\":\"2009/06/29\",\"InventoryCode\":null,\"UserOwnsJobContent\":false,\"ReferenceNumber\":\"\",\"Price\":220}
    ":\"\",\"Price\":220}],\"Category\":[{\"__type\":\"SEEK.Services.InventoryCategory\",\"ClassificationType\":1,\"Clas
    "AUSKPRL\",\"FirstPurchased\":6,\"FirstContent\":18,\"FirstAlert\":18},{\"__type\":\"SEEK.Services.Client.SearchResu
    ":null,\"UserOwnsJobContent\":false,\"ReferenceNumber\":\"\",\"Price\":500},{\"__type\":\"SEEK.Services.ResultItem\"
    "SEEK.Services.ResultItem\",\"Key\":{\"__type\":\"SEEK.Services.ItemCode\",\"Code\":\"ONHFXCEY-62587554555555-785875
    "ONHFXCEY-79587554555555-85587554780404-6557-6755-6856-61833\"},\"HasAlert\":false,\"PriceBand\":2,\"Status\":0,\"De
    ":2,\"Status\":0,\"Description\":\"\",\"JobID\":0,\"InCart\":false,\"FromDate\":\"2009/03/31\",\"ToDate\":\"2009/04/
    "ToDate\":\"2009/04/13\",\"InventoryCode\":null,\"UserOwnsJobContent\":false,\"ReferenceNumber\":\"\",\"Price\":500}
    ":\"\",\"Price\":500},{\"__type\":\"SEEK.Services.ResultItem\",\"Key\":{\"__type\":\"SEEK.Services.ItemCode\",\"Code
    "SEEK.Services.ItemCode\",\"Code\":\"ONHFXCEY-73597554555555-59507554780404-6557-6755-6856-6ON25\"},\"HasAlert\":fal
    "\"},\"HasAlert\":false,\"PriceBand\":2,\"Status\":0,\"Description\":\"\",\"JobID\":0,\"InCart\":false,\"FromDate\":
    ":false,\"FromDate\":\"2009/05/12\",\"ToDate\":\"2009/05/18\",\"InventoryCode\":null,\"UserOwnsJobContent\":false,\"
    "UserOwnsJobContent\":false,\"ReferenceNumber\":\"\",\"Price\":500},{\"__type\":\"SEEK.Services.ResultItem\",\"Key\"
    "SEEK.Services.ResultItem\",\"Key\":{\"__type\":\"SEEK.Services.ItemCode\",\"Code\":\"ONHFXCEY-57517554555555-535175
    "ONHFXCEY-54517554555555-60517554780404-6557-6755-6856-6656R\"},\"HasAlert\":false,\"PriceBand\":2,\"Status\":0,\"De
    ":2,\"Status\":0,\"Description\":\"\",\"JobID\":0,\"InCart\":false,\"FromDate\":\"2009/06/16\",\"ToDate\":\"2009/06/
    "ToDate\":\"2009/06/29\",\"InventoryCode\":null,\"UserOwnsJobContent\":false,\"ReferenceNumber\":\"\",\"Price\":500}
    ":\"\",\"Price\":500}],\"Category\":[{\"__type\":\"SEEK.Services.InventoryCategory\",\"ClassificationType\":1,\"Clas
    "FirstPurchased\":18,\"FirstContent\":18,\"FirstAlert\":18}]}",
```

# Client-Side Performance

- Was not a problem previously, as client-side time is trivially small for *traditional* web applications.
  - Client-side processing time can now exceed server + network time by a factor of 5 for some applications.
- JavaScript profiling (FireBug Profiler)

**Profile (360.517ms, 6560 calls)**

| Function | Calls | Percent | Time | Avg | Min | Max | File |
|---|---|---|---|---|---|---|---|
| tt_HideSrcTagsRecurs | 1696 | 19.44% | 70.101ms | 0.041ms | 0ms | 70.101ms | tooltip.js (line 458) |
| merge | 7 | 13.89% | 50.072ms | 7.153ms | 0ms | 30.043ms | jquery.js (line 753) |
| tt_GetDivW | 1 | 11.11% | 40.058ms | 40.058ms | 40.058ms | 40.058ms | tooltip.js (line 195) |
| displayAd | 2 | 11.11% | 40.057ms | 20.029ms | 20.028ms | 20.029ms | dap.js (line 101) |
| _D | 114 | 5.56% | 30.044ms | 0.264ms | 0ms | 10.015ms | hbxdigg.js (line 35) |
| classFilter | 2 | 5.56% | 20.029ms | 10.015ms | 10.014ms | 10.015ms | jquery.js (line 1243) |
| _IL | 1321 | 5.56% | 20.028ms | 0.015ms | 0ms | 10.014ms | hbxdigg.js (line 38) |
| _EV | 294 | 2.78% | 10.015ms | 0.034ms | 0ms | 10.015ms | hbxdigg.js (line 104) |

- Memory leaks and overconsumption
  - Watch out for JavaScript allocating large amounts of memory

# Summary

- Glossed over…
  - Include questions about Web 2.0 technologies in your performance testing scoping phase.
  - Usage profiles are more difficult to predict with external consumers. Limit upper boundary using Policy Enforcement.
  - If you have a composite application, use the opportunity to performance test components independently.
- Focused on…
  - Next generation load testing tools which operate at a higher level than HTTP will simplify and shorten the scripting phase.
  - Client-side performance and resource usage/leakage must now be tested for.
- Leverage load testing tools which handle Flash etc.
  - Don’t just pick a load testing tool at random.