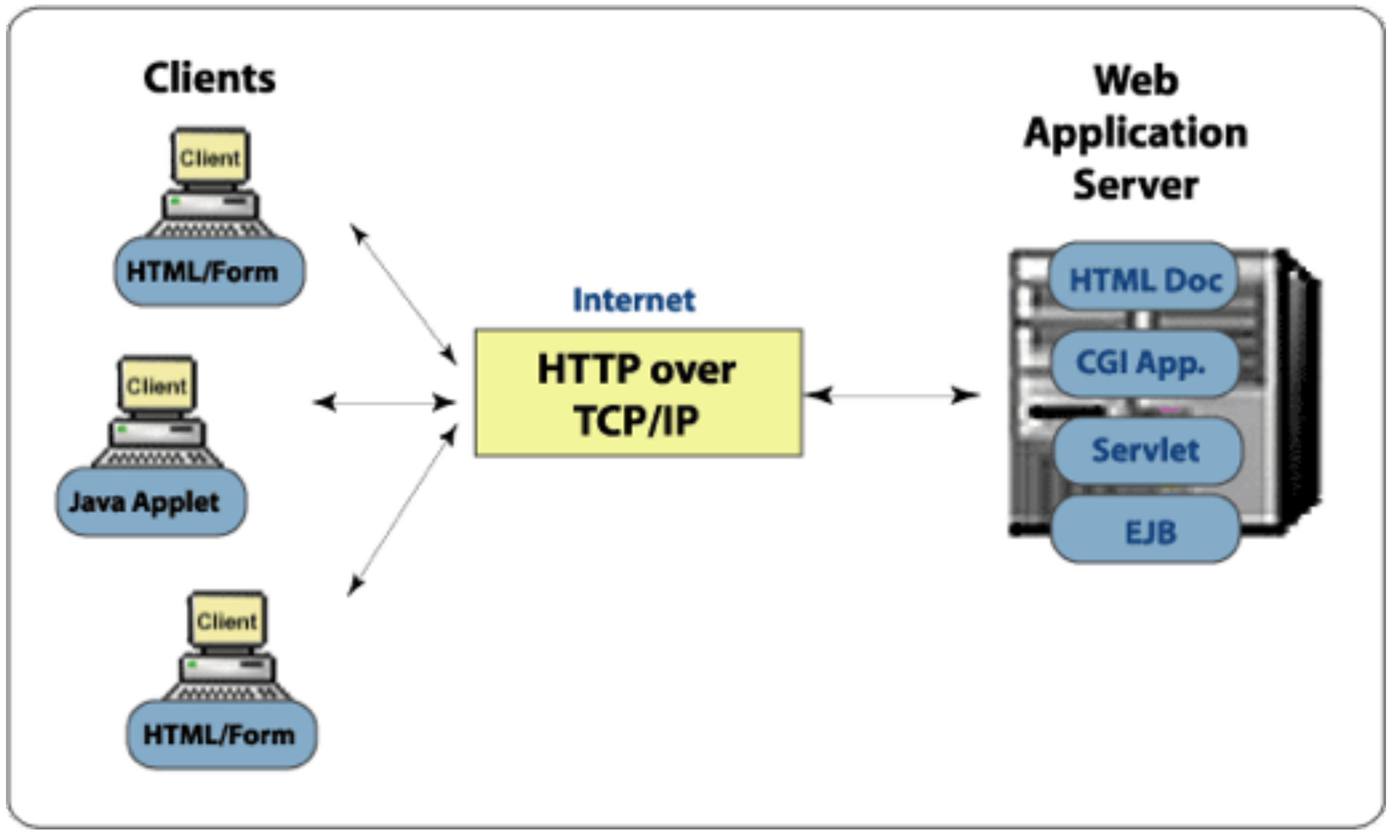
Web Applications Testing

Automated testing and verification

JP Galeotti, Alessandra Gorla

Why are Web applications different



Web 1.0: Static content

Client and Server side execution

Different components and technologies co-exist

Web 2.0: Dynamic contents

Heavy use of JavaScript and Ajax.

Users can collaborate, socialize and share...

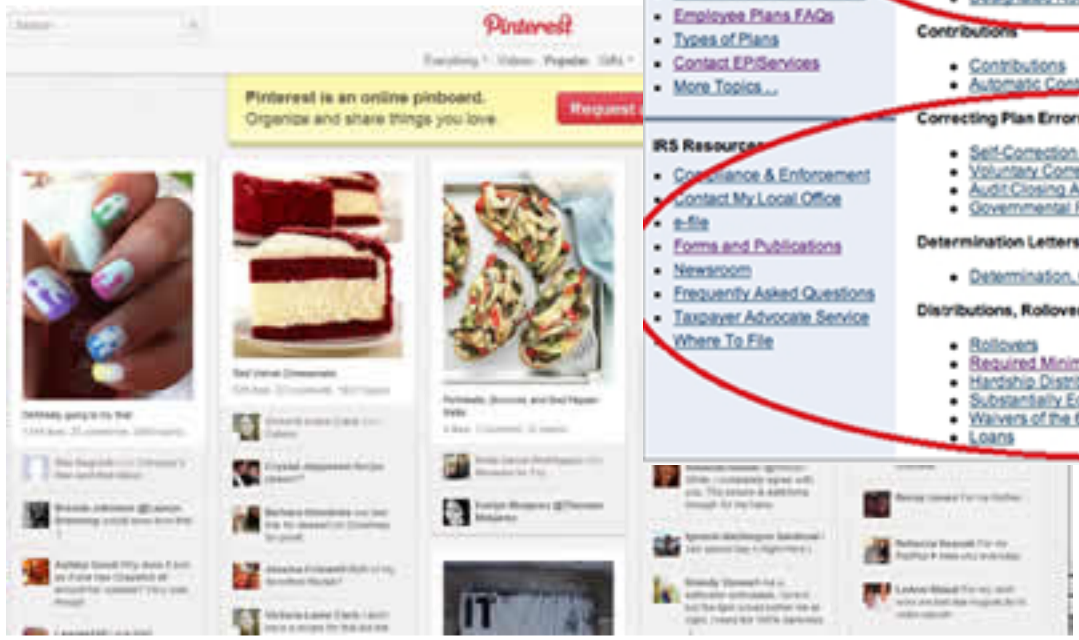
Important qualities in Web applications

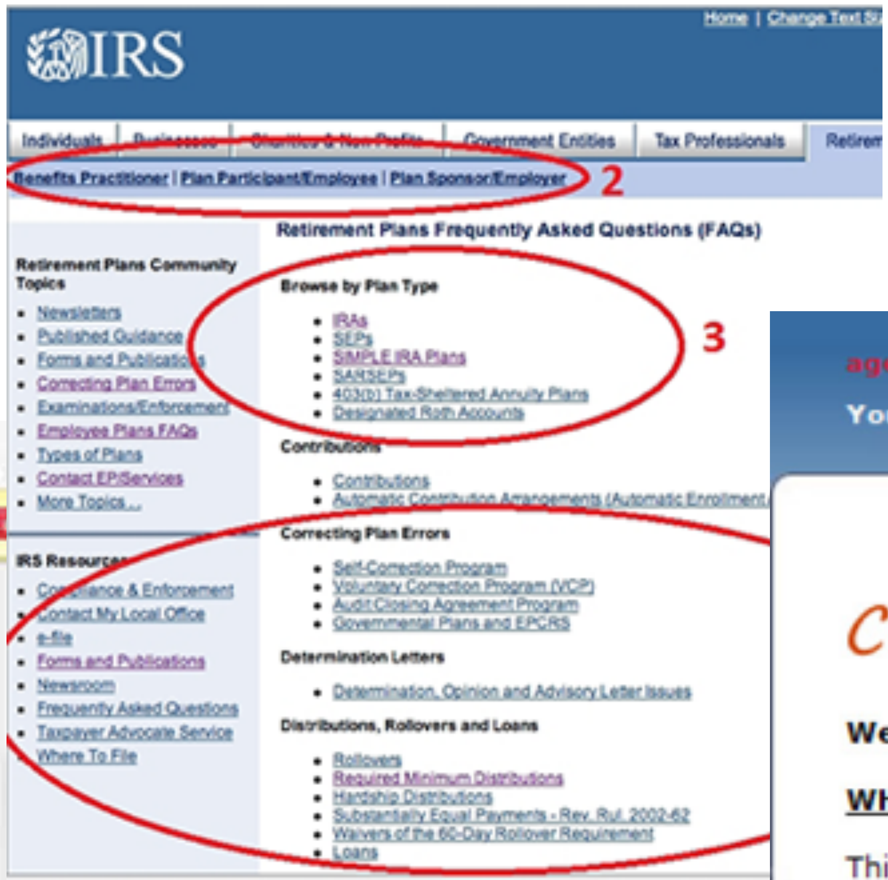
- Reliability
- Usability
- Performance
- Robustness/Scalability
- Security

Usability

- Assessing the ease of use and learnability of web application
    - Useful
    - Ease of use
    - Ease of learning
    - Pleasant look

Usability : Bad examples


Pinterest
Pinterest is an online pinboard.
Organize and share things you love.


Home | Change Text Si
IRS
Individuals | Businesses | Charities & Non-Profits | Government Entities | Tax Professionals | Retirem
2
3

Benefits Practitioner | Plan Participant/Employee | Plan Sponsor/Employer

Retirement Plans Frequently Asked Questions (FAQs)

Retirement Plans Community Topics

- Newsletters
- Published Guidance
- Forms and Publications
- Correcting Plan Errors
- Examinations/Enforcement
- Employee Plans FAQs
- Types of Plans
- Contact EP/Services
- More Topics...

IRS Resources

- Compliance & Enforcement
- Contact My Local Office
- e-file
- Forms and Publications
- Newsroom
- Frequently Asked Questions
- Taxpayer Advocate Service
- Where To File

Browse by Plan Type

- IRAs
- SEPs
- SIMPLE IRA Plans
- SARSEPs
- 403(b) Tax-Sheltered Annuity Plans
- Designated Roth Accounts

Contributions

- Contributions
- Automatic Contribution Arrangements (Automatic Enrollment

Correcting Plan Errors

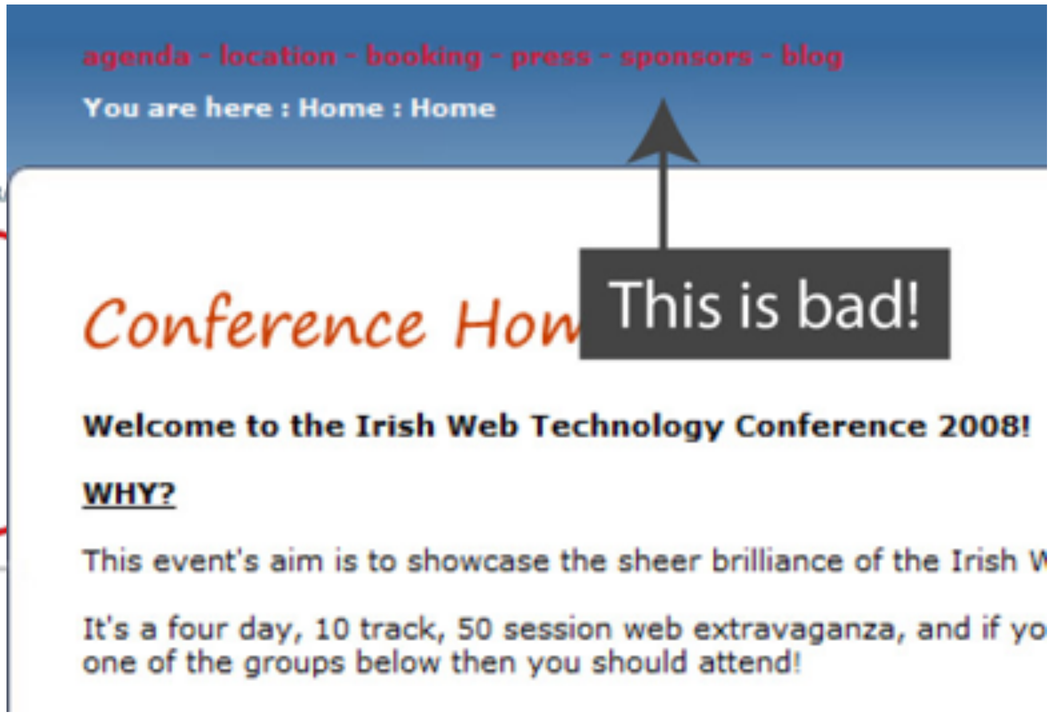
- Self-Correction Program
- Voluntary Correction Program (VCP)
- Audit Closing Agreement Program
- Governmental Plans and EPCRS

Determination Letters

- Determination, Opinion and Advisory Letter Issues

Distributions, Rollovers and Loans

- Rollovers
- Required Minimum Distributions
- Hardship Distributions
- Substantially Equal Payments - Rev. Rul. 2002-62
- Waivers of the 60-Day Rollover Requirement
- Loans


agenda - location - booking - press - sponsors - blog
You are here : Home : Home
Conference Hom
This is bad!

Welcome to the Irish Web Technology Conference 2008!

WHY?

This event's aim is to showcase the sheer brilliance of the Irish W

It's a four day, 10 track, 50 session web extravaganza, and if yo
one of the groups below then you should attend!

A lot of information
in the homepage

Several clicks before
finding wanted information

Bad colors

Assessing Usability

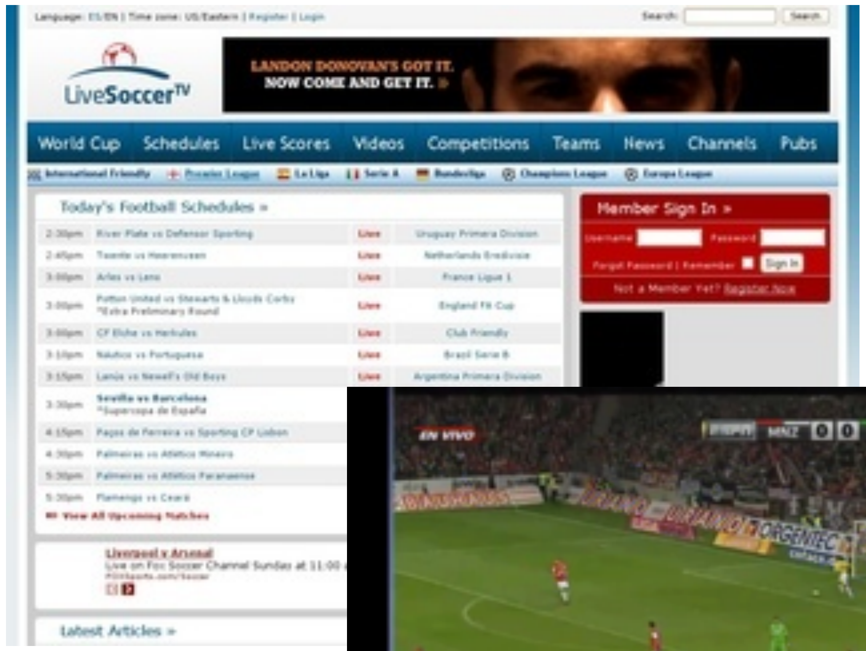
- Some properties can be *verified* to assess usability:
  - Use colors that are compatible with color-blind people
  - Have a limited number of clicks to reach any information on the website
  - Provide a maximum number of links per page.
  - ...

Usability testing

- Usability testing techniques can be applied to web applications as well:
    - **Exploratory** testing to investigate mental model of users
    - **Comparison** testing to evaluate different options
    - **Usability** validation testing to assess the overall usablity

Performance

- Assure that the web application can deal with expected workloads.
    - E.g. I should expect usually between 100 to 1000 user requests per hour.
- Workloads can vary



Load test

- Simulate expected workload to test the ability of web application to deal with it.
- Goals:
  - Evaluate if the current infrastructure (HW and SW) is enough to satisfy the required SLA given the amount of expected workload.
  - Estimate how many requests can be currently handled.
- Typically done when the system is ready

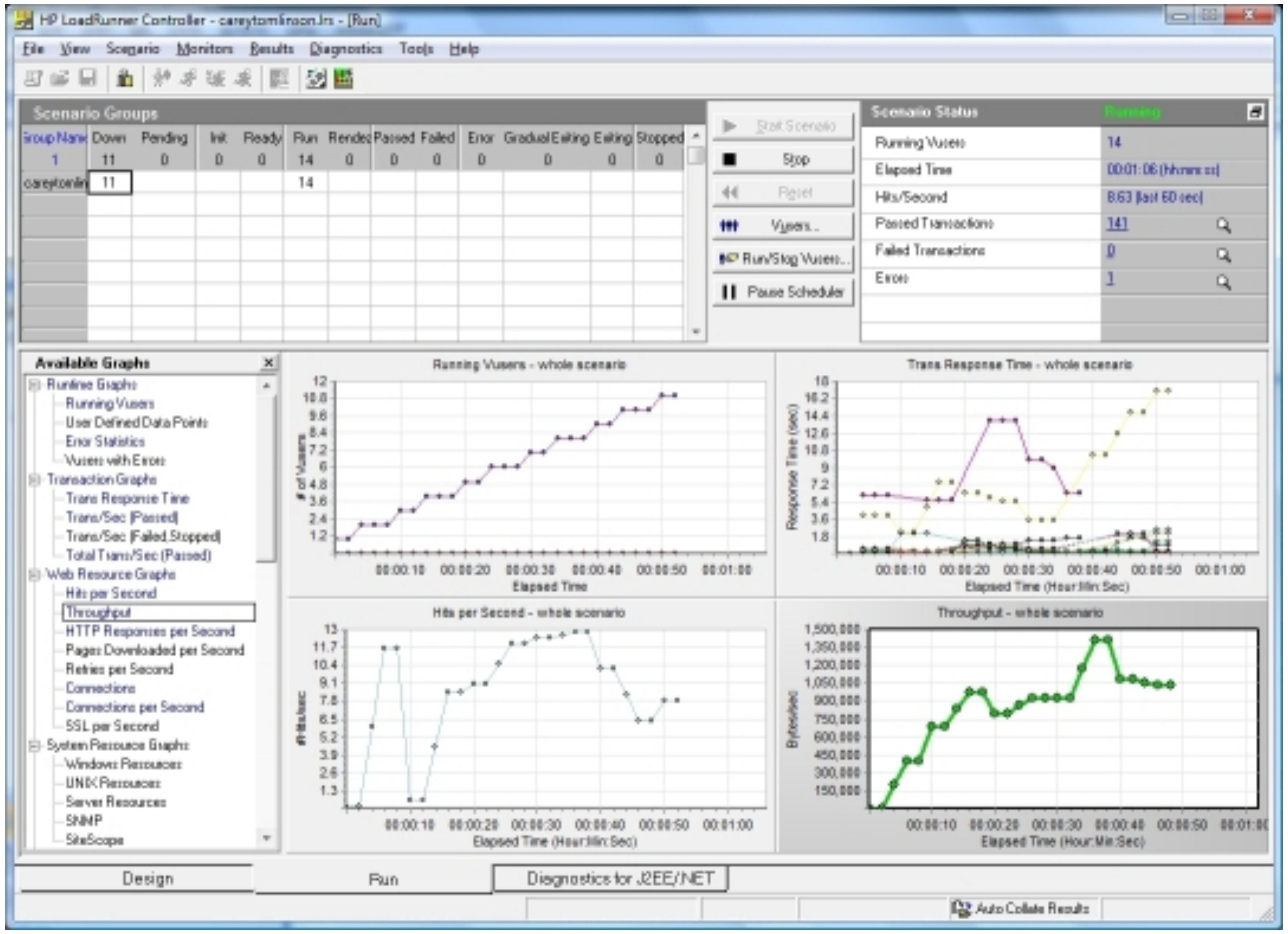
Approach for Load Testing

- Identify performance acceptance criteria
    - Page should load fast --> Page should be fully loaded in less than 3 seconds.
- Identify key scenarios
    - E.g. Browse catalog; Select item; Add to cart; buy;
- Create a workload model
    - Define workload distribution, considering different scenarios
- Identify the target load levels
- Identify metrics
- Design specific tests
- Run tests
- Analyze results

Load test

- Historically tests were manually done (asking real users to perform tests)
- Now, lots of automation.
  - Tools can register activities of real users and create scripts to reproduce the same activities.
  - Scripts can be configured to create a set of virtual users who do same activities at the same time.

LoadRunner from Mercury (now HP)



Scenario Groups

Group Name	Down	Pending	Init	Ready	Run	Rendez	Passed	Failed	Error	Gradual Exiting	Exiting	Stopped
1	11	0	0	0	14	0	0	0	0	0	0	0
careytomlin	11				14							

Scenario Status Running

Running Vusers	14
Elapsed Time	00:01:06 (hh:mm:ss)
Hits/Second	8.63 (last 60 sec)
Passed Transactions	141
Failed Transactions	0
Errors	1

Industrial standard for load testing.

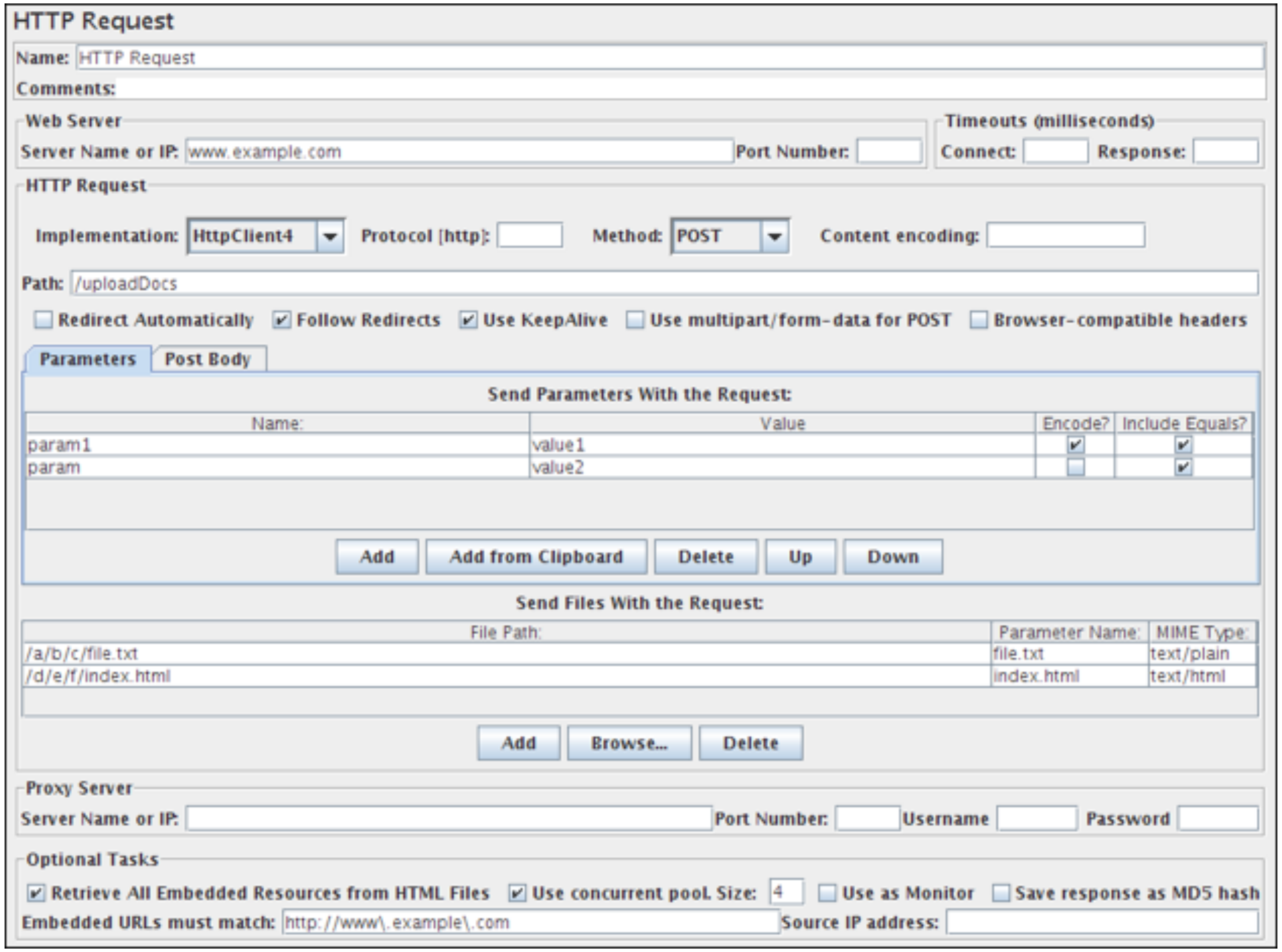
Can emulate the load generated by any number of users, from few to several thousands.

Supports different types of applications and systems: web server, database, ERP, firewall...

JMeter - the open source alternative

- GUI desktop application designed to load test functional behavior and measure performance. It was originally designed for testing Web Applications but has since expanded to other test functions.
- It can be used to simulate a heavy load on a server, network or object to test its strength or to analyze overall performance under different load types
- Remote Distributed Execution: it is possible to generate load using multiple test servers (by running multiple server components of JMeter remotely).
  - Possible to control it by a single JMeter GUI to gather results.

Create a test plan: Sampler



HTTP Request

Name: HTTP Request

Comments:

Web Server

Server Name or IP: www.example.com Port Number:

Timeouts (milliseconds)

Connect: Response:

HTTP Request

Implementation: HttpClient4 Protocol [http]: Method: POST Content encoding:

Path: /uploadDocs

☐ Redirect Automatically ☑ Follow Redirects ☑ Use KeepAlive ☐ Use multipart/form-data for POST ☐ Browser-compatible headers

Parameters | Post Body

Send Parameters With the Request:

Name:	Value	Encode?	Include Equals?
param1	value1	✓	✓
param	value2		✓

Add | Add from Clipboard | Delete | Up | Down

Send Files With the Request:

File Path:	Parameter Name:	MIME Type:
/a/b/c/file.txt	file.txt	text/plain
/d/e/f/index.html	index.html	text/html

Add | Browse... | Delete

Proxy Server

Server Name or IP: Port Number: Username Password

Optional Tasks

☑ Retrieve All Embedded Resources from HTML Files ☑ Use concurrent pool. Size: 4 ☐ Use as Monitor ☐ Save response as MD5 hash

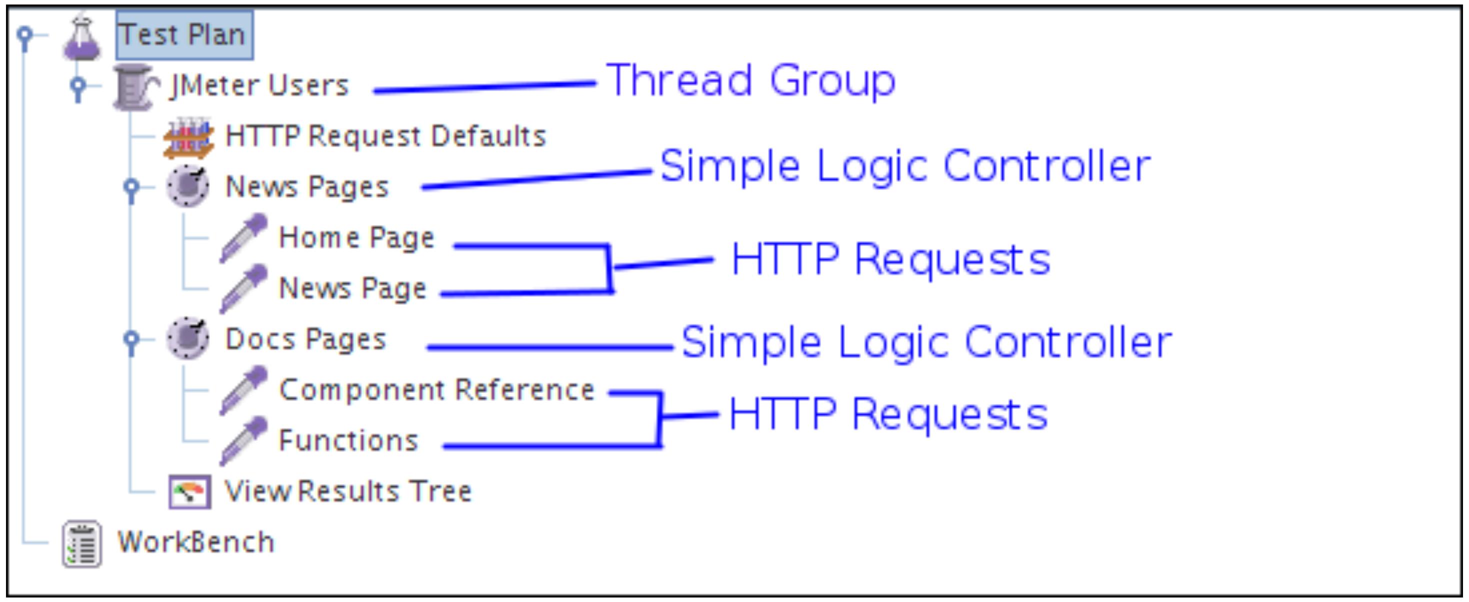
Embedded URLs must match: http://www\.example\.com Source IP address:

Create HTTP/HTTPS requests to a web server.

Possible to specify whether JMeter should parse HTML for resources (e.g. images, frames...)

Logic controllers

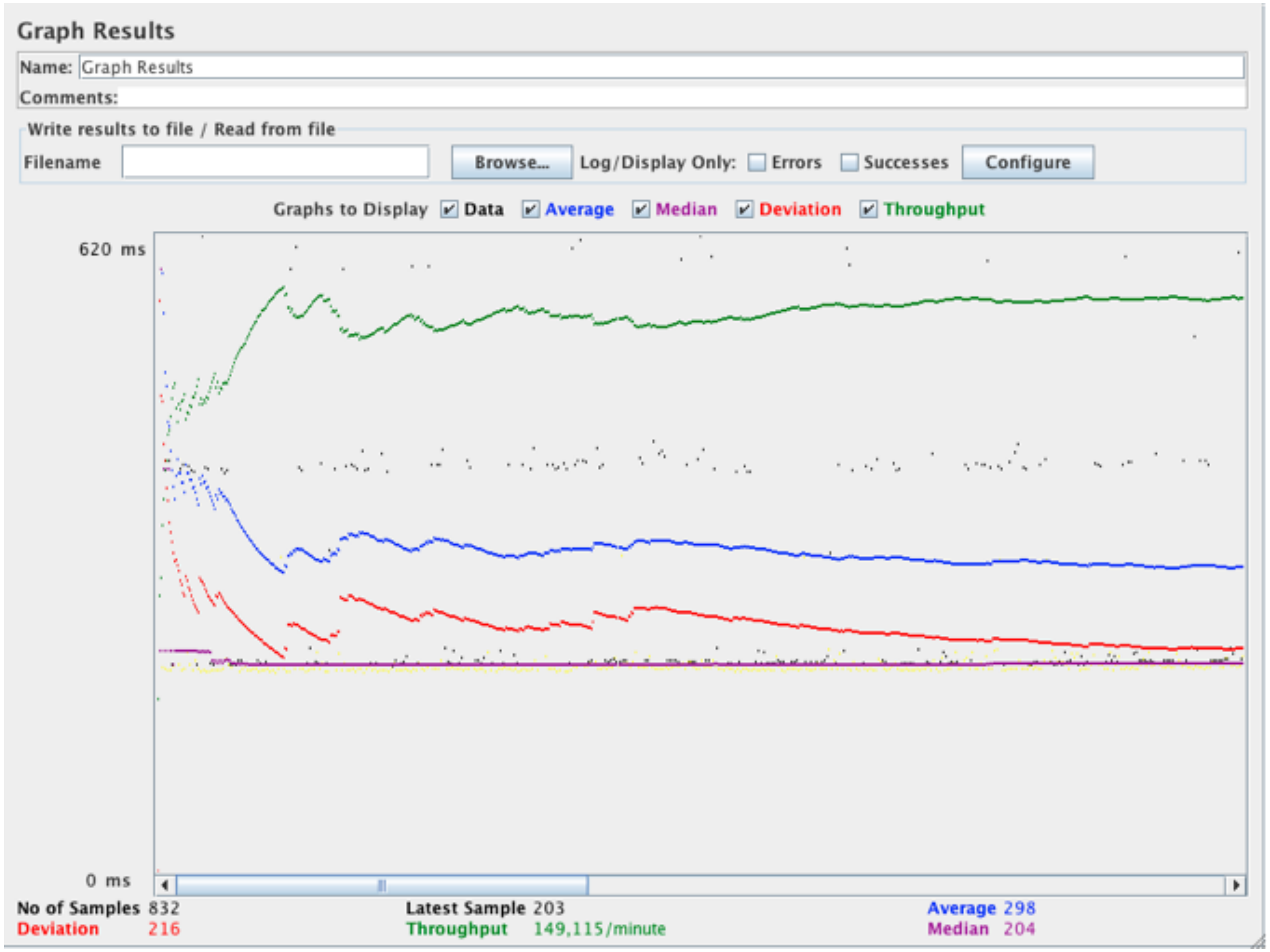
Determine the order in which Samplers are processed.



Execute sequentially
Execute samplers in loops
Alternate among different samplers
Random
Conditionally execute
...

Listeners

Meant to “listen” to the test results. Provide means to view, save and read saved test results.



Graph Results

Name: Graph Results

Comments:

Write results to file / Read from file

Filename

Log/Display Only: Errors Successes

Graphs to Display [✓ Data] [✓ Average] [✓ Median] [✓ Deviation] [✓ Throughput]

No of Samples 832
Deviation 216
Latest Sample 203
Throughput 149,115/minute
Average 298
Median 204

Robustness/Scalability

- Assure that software behaves in acceptable way even in presence of situations that were not mentioned in the requirements.
  - The most important variable for web apps is the workload.
- Scalability: Managing several requests with QOS. Being able to deal with workloads that are heavier than originally considered.

Stress testing

Highest estimated workload: 1000 requests per second

Test with 2000, 5000, 10000 ... requests per second

Security testing

- Goal: Assure that the application protects data and maintains functionality as intended.
- Mostly manual, little automation support

Manual

Security experts try to actually break the system and find vulnerabilities

Automation

Static analysis techniques: scan application code and detect likely security flaws

Fuzz testing techniques: create random/malformed inputs to make app crash or find security hole

Fuzzers

- Generators use combinations of static fuzzing vectors (values that are known to be dangerous), or totally random data.
- A fuzzer would try combinations of attacks on:
  - numbers (signed/unsigned integers/float) : zero, possibly negative or very big numbers
  - chars (URLs, command-line inputs): escaped, interpretable characters/instructions (e.g. quotes and commands for SQL requests)
  - metadata: (user-input text)
  - pure binary sequences
- Usually black-box
  - White-box fuzzers start from well-formed inputs and use dynamic symbolic execution to generate new ones.

Reliability

- Assure that the system performs and maintains its functions in normal circumstances.
- Classic functional, structural and model based techniques can be used for web applications as well.
    - Different automation support to implement and run unit and system tests
- Major challenge: Assure that a web application works correctly in different environments (OS, Browsers ...)
    - Cross browser compatibility

Unit testing - HttpUnit

- **HttpUnit**: JUnit for web
- It models HTML documents, and provides an API that allows to invoke pages, fill out forms, click links...

```
@Test
public void homePage() throws Exception {
    final WebClient webClient = new WebClient();
    final HtmlPage page = webClient.getPage("http://htmlunit.sourceforge.net");
    Assert.assertEquals("HtmlUnit - Welcome to HtmlUnit", page.getTitleText());

    final String pageAsXml = page.asXml();
    Assert.assertTrue(pageAsXml.contains("<body class=\"composite\">"));

    final String pageAsText = page.asText();
    Assert.assertTrue(pageAsText.contains("Support for the HTTP and HTTPS protocols"));

    webClient.closeAllWindows();
}
```

Limited JavaScript support

Unit testing - JSUnit

- JUnit for Javascript

```
<html>
   <script type='text/javascript'>

    function setUp(){
      // perform fixture set up
    }
    function tearDown() {
      // clean up
    }
    function testOneThing(){
      // instantiating a SystemUnderTest, a class in the drw namespace
      var sut = new drw.SystemUnderTest();
      var thing = sut.oneThing();
      assertEquals(1, thing);
    }


    function testAnotherThing(){
      var sut = new drw.SystemUnderTest();
      var thing = sut.anotherThing();
      assertNotEquals(1, thing);
    }
  </script>
 </head>
 <body/>
</html>
```

Unit testing

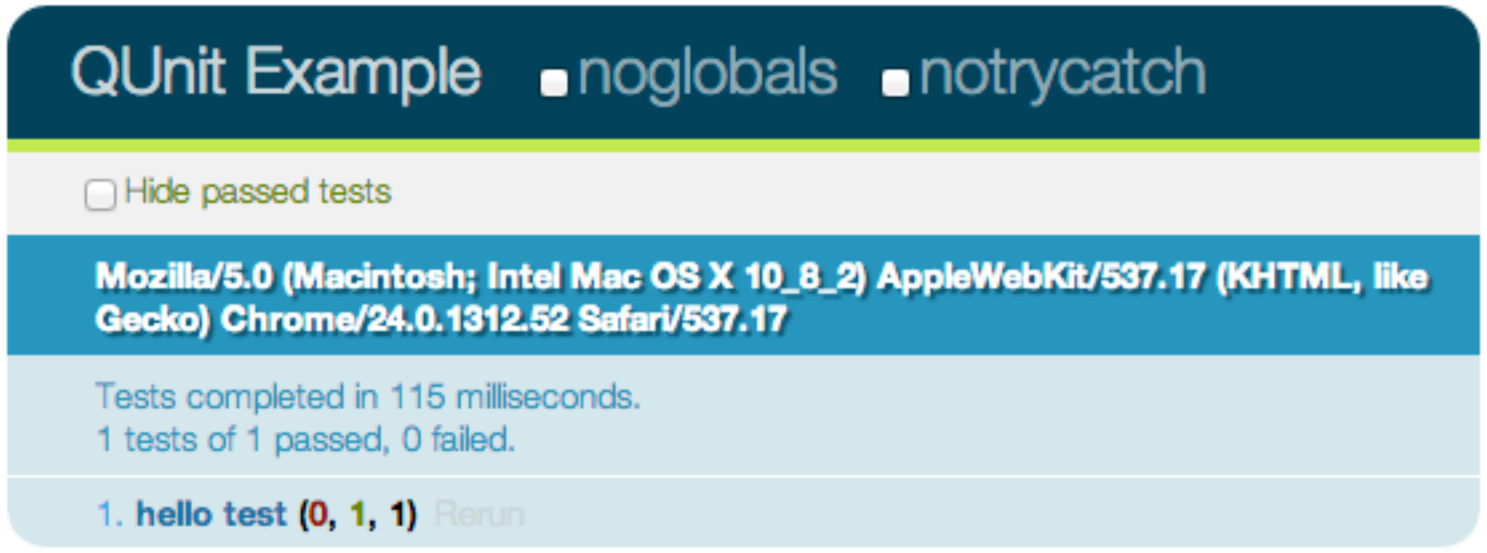
- Other tools:
  - HtmlUnit (same as HttpUnit, but better Javascript support)
  - Canoo Web testing
    - Built on top of HtmlUnit
    - Makes it easier to organize and run test cases
- ...

Unit testing - QUnit

- Advanced Javascript unit testing framework

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>QUnit Example</title>
  <link rel="stylesheet" href="/resources/qunit.css">
</head>
<body>
  <div id="qunit"></div>
  <div id="qunit-fixture"></div>
  <script src="/resources/qunit.js"></script>
  <script src="/resources/tests.js"></script>
</body>
</html>
```

```
test( "hello test", function() {
  ok( 1 == "1", "Passed!" );
});
```

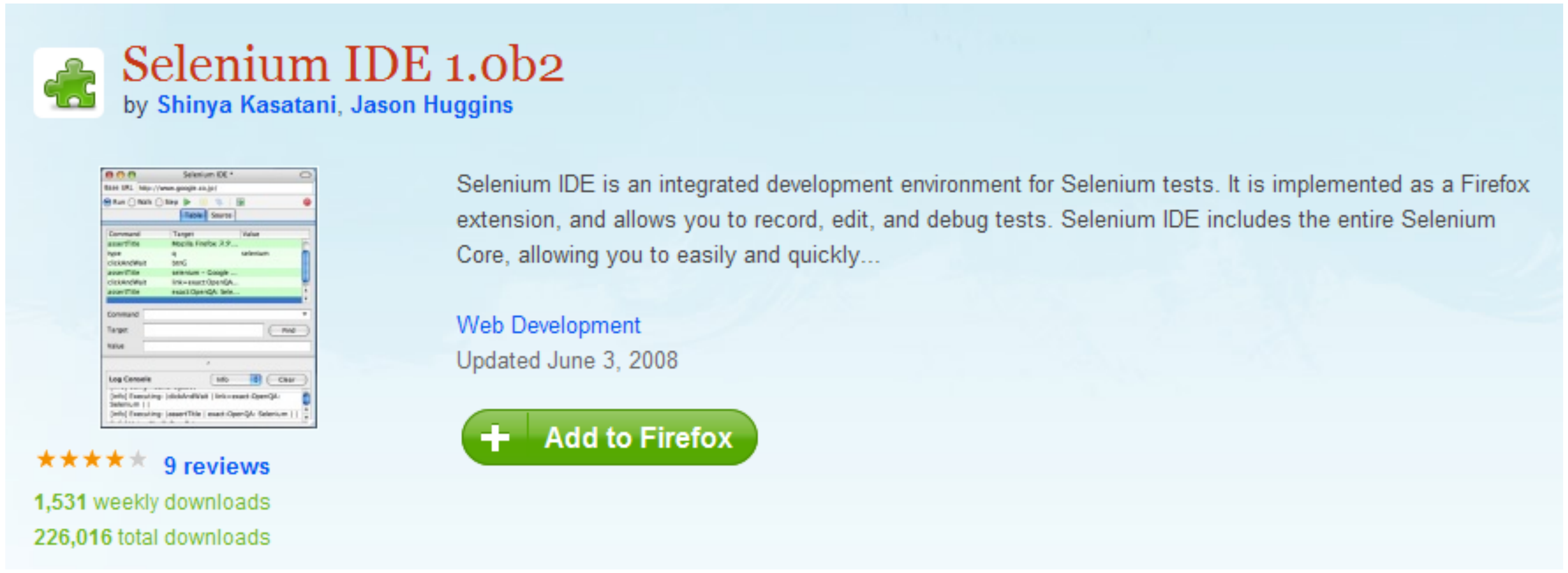


Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_2) AppleWebKit/537.17 (KHTML, like Gecko) Chrome/24.0.1312.52 Safari/537.17

Tests completed in 115 milliseconds.
1 tests of 1 passed, 0 failed.

1. hello test (0, 1, 1)

System test



Selenium IDE 1.0b2

by Shinya Kasatani, Jason Huggins

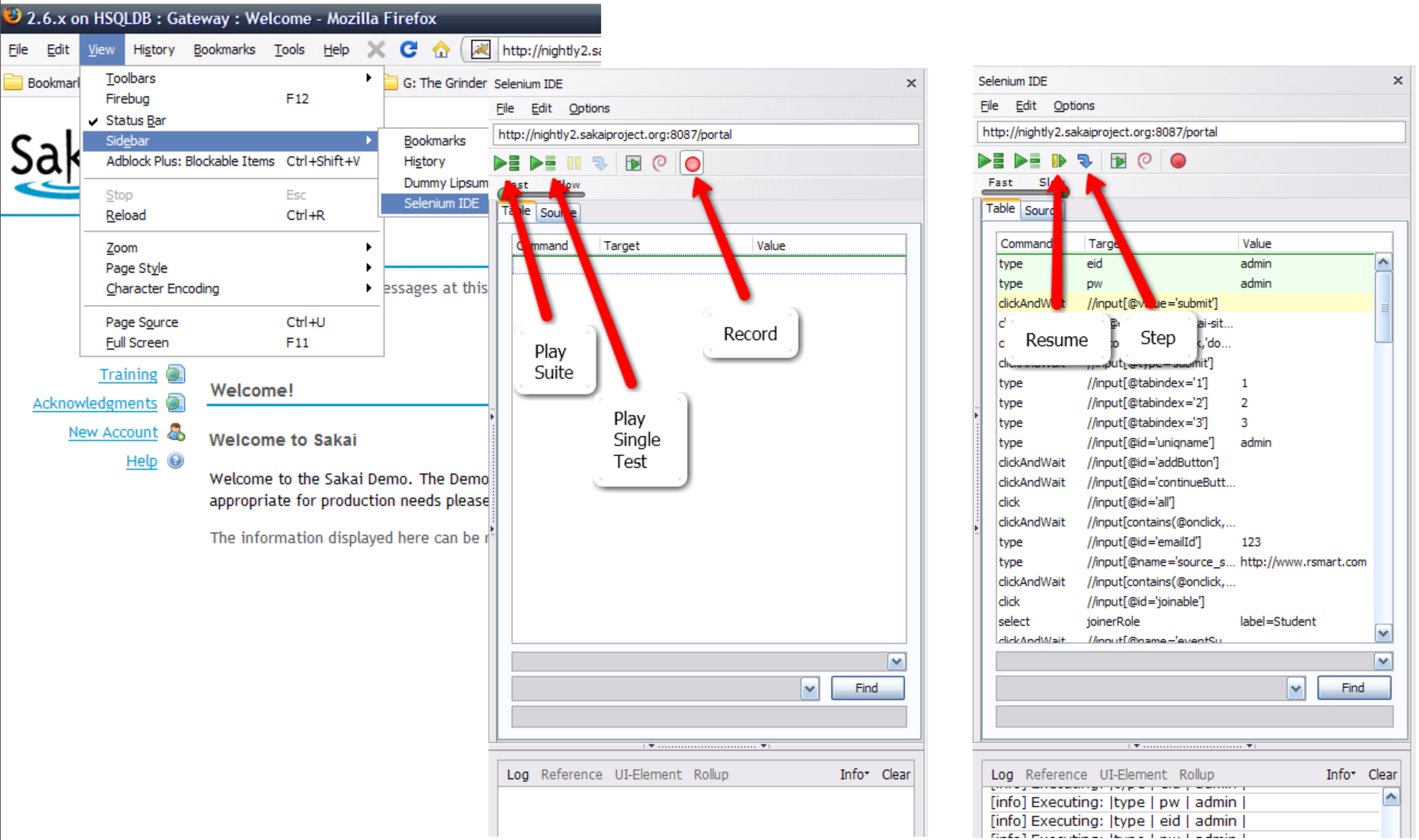
Selenium IDE is an integrated development environment for Selenium tests. It is implemented as a Firefox extension, and allows you to record, edit, and debug tests. Selenium IDE includes the entire Selenium Core, allowing you to easily and quickly...

Web Development

Updated June 3, 2008

Easily capture and replay actions

Selenium IDE



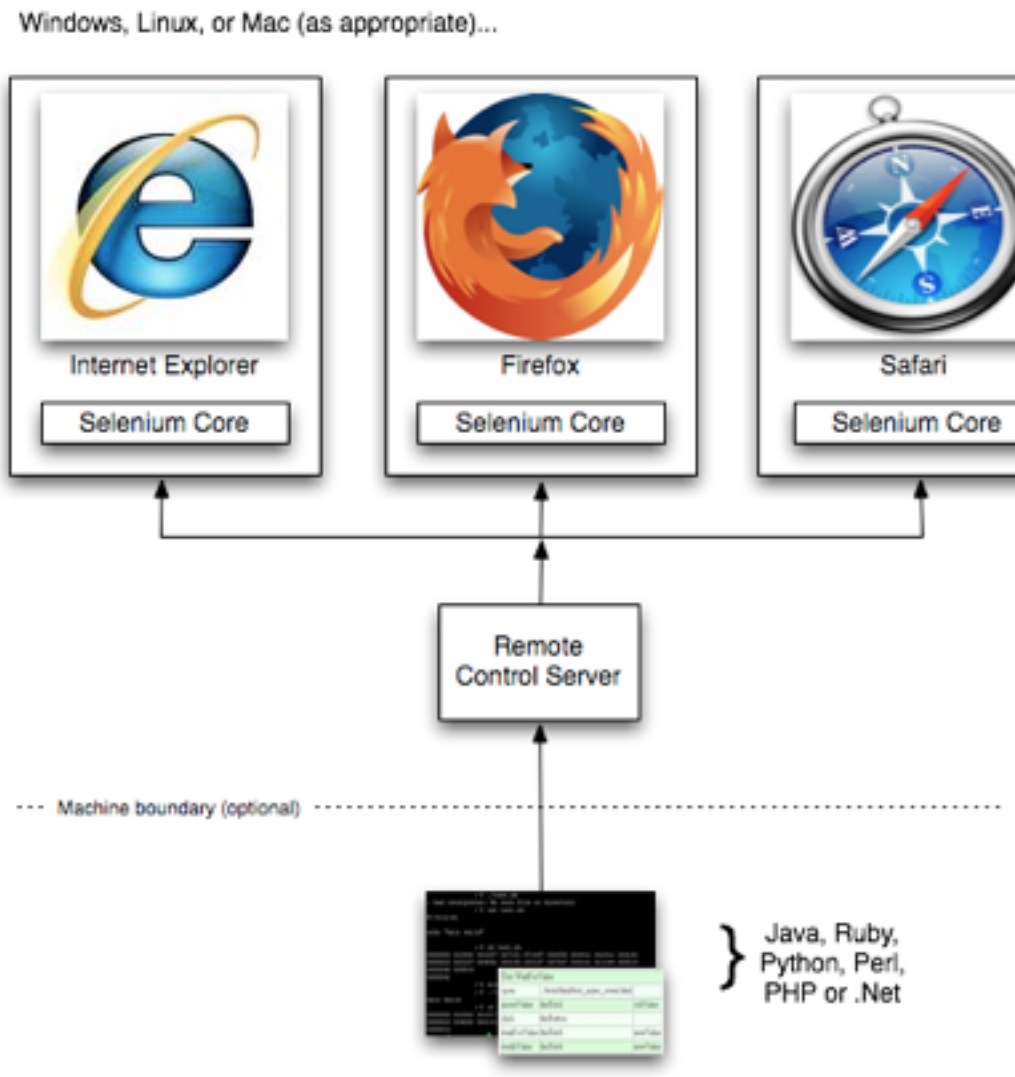
Welcome to the Sakai Demo. The Demo appropriate for production needs please

The information displayed here can be

Command	Target	Value
type	eid	admin
type	pw	admin
clickAndW	//input[@v ue='submit']	
c	ai-sit...	
c	o k,'do...	
cl	mit']	
type	//input[@tabindex='1']	1
type	//input[@tabindex='2']	2
type	//input[@tabindex='3']	3
type	//input[@id='uniqname']	admin
clickAndWait	//input[@id='addButton']	
clickAndWait	//input[@id='continueButt...	
click	//input[@id='all']	
clickAndWait	//input[contains(@onclick,...	
type	//input[@id='emailId']	123
type	//input[@name='source_s...	http://www.rsmart.com
clickAndWait	//input[contains(@onclick,...	
click	//input[@id='joinable']	
select	joinerRole	label=Student

```
[info] Executing: |type | pw | admin |
[info] Executing: |type | eid | admin |
```

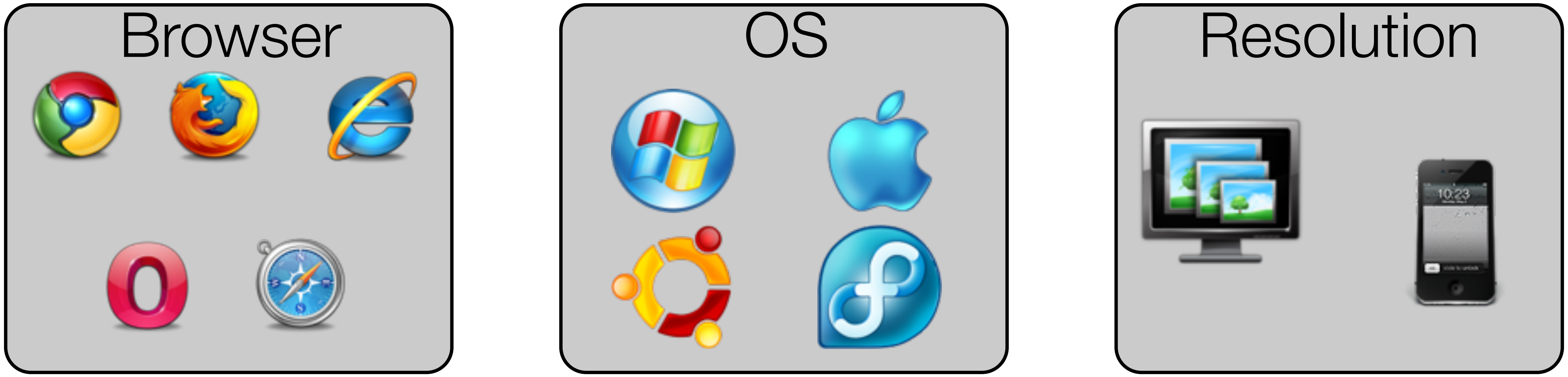
Selenium RC


Windows, Linux, or Mac (as appropriate)...
Internet Explorer
Selenium Core
Firefox
Selenium Core
Safari
Selenium Core
Remote Control Server
Machine boundary (optional)
Java, Ruby, Python, Perl, PHP or .Net

```
// You may use any WebDriver implementation. Firefox is used
here as an example
WebDriver driver = new FirefoxDriver();

// A "base url", used by selenium to resolve relative URLs
 String baseUrl = "http://www.google.com";

// Create the Selenium implementation
Selenium selenium = new WebDriverBackedSelenium(driver,
baseUrl);

// Perform actions with selenium

selenium.open("http://www.google.com");
selenium.type("name=q", "cheese");
selenium.click("name=btnG");

// Get the underlying WebDriver implementation back. This will
refer to the
// same WebDriver instance as the "driver" variable above.
WebDriver driverInstance = ((WebDriverBackedSelenium)
selenium).getWrappedDriver();

//Finally, close the browser. Call stop on the
WebDriverBackedSelenium instance
//instead of calling driver.quit(). Otherwise, the JVM will
continue running after
//the browser has been closed.
selenium.stop();
```

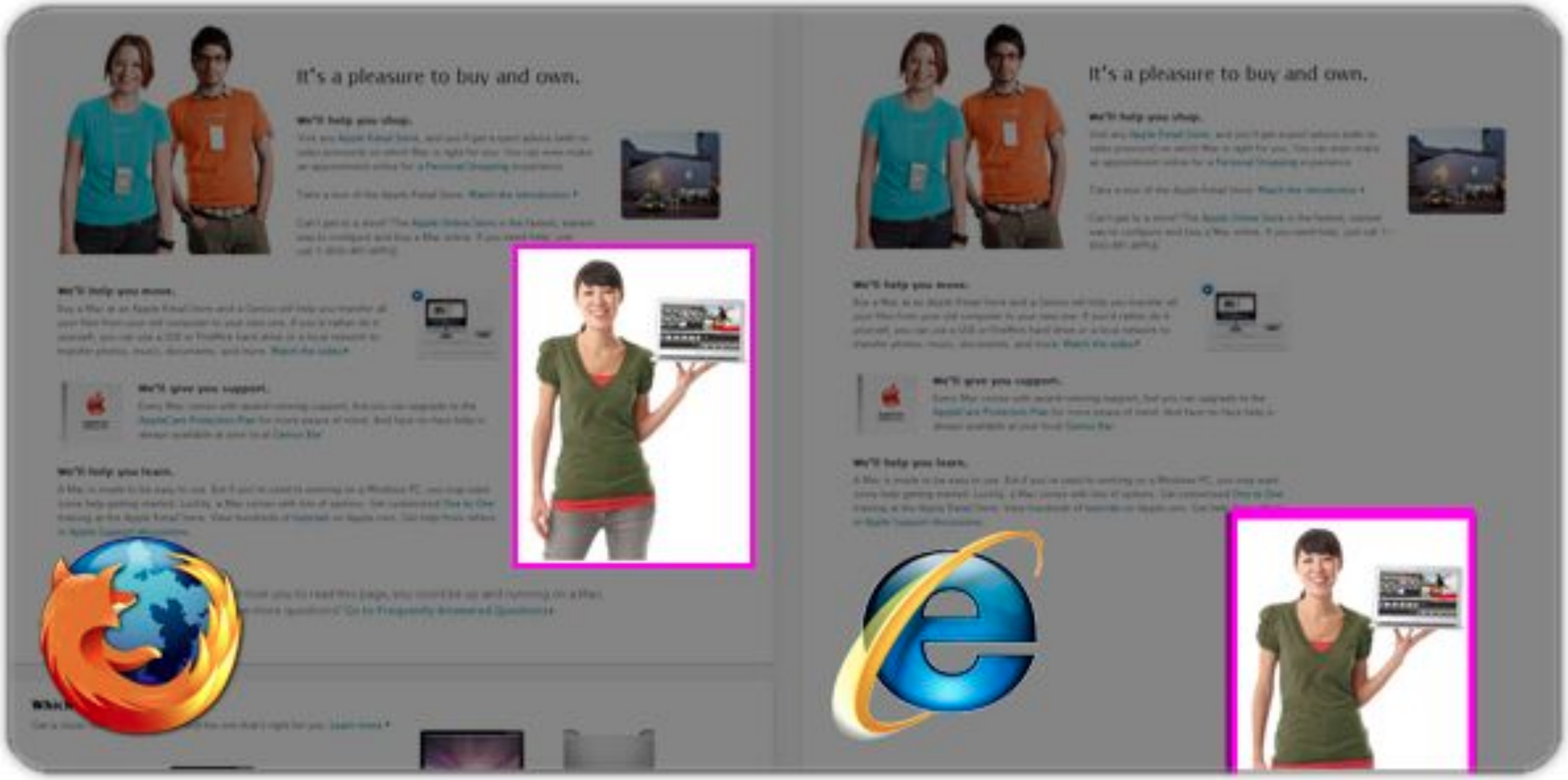
APIs in Java, Python, Ruby,
Php, JavaScript ...

Cross browser compatibility



- Elements in Web pages may appear differently on different combinations
- Functionality may work on some browsers and not on others
- Most of the issues are related to CSS and JavaScript

Example of layout issue

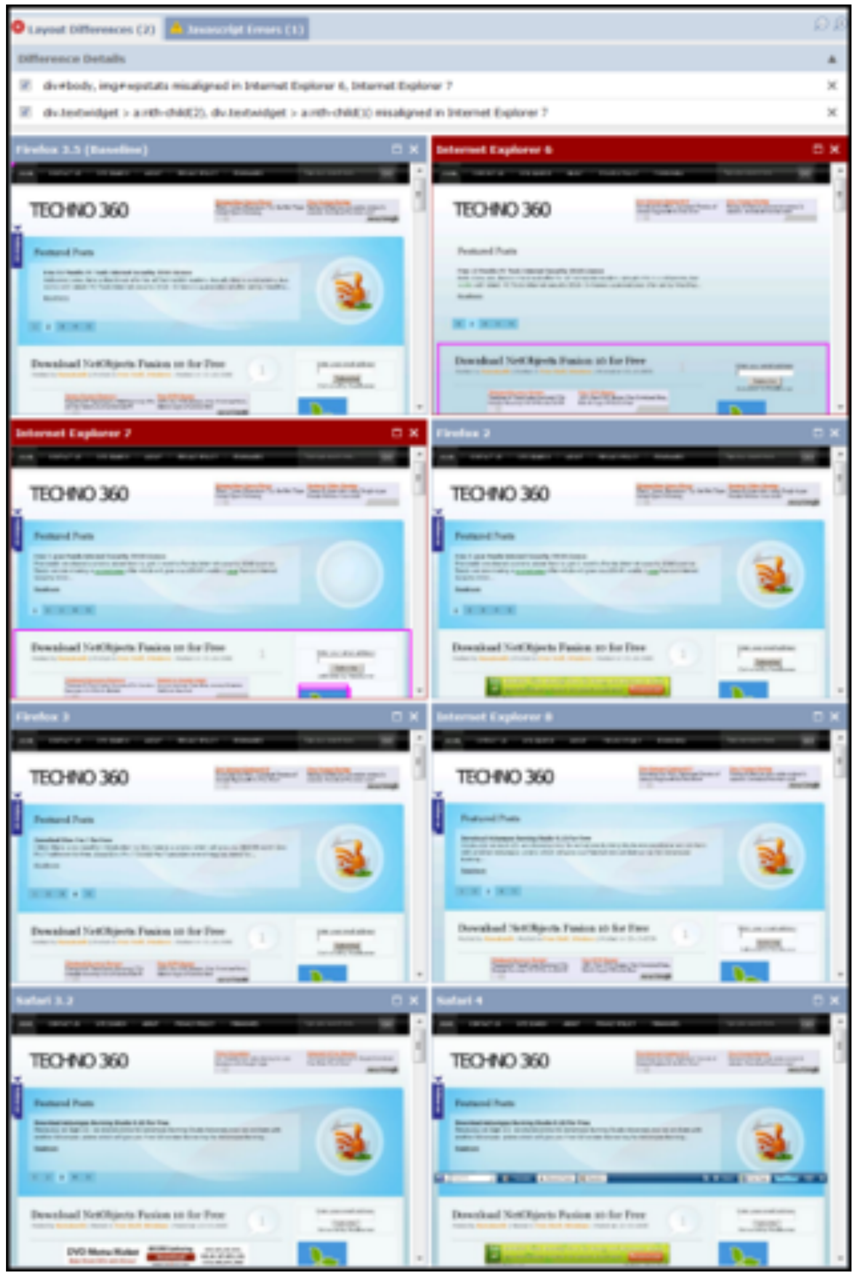


Cross browser testing

- Load specific web page on relevant combinations of browsers, OS and resolutions and compare the output.
- Provide URL to a service, and have your web page loaded in the requested combinations.
- A set of “screenshots” as a result.

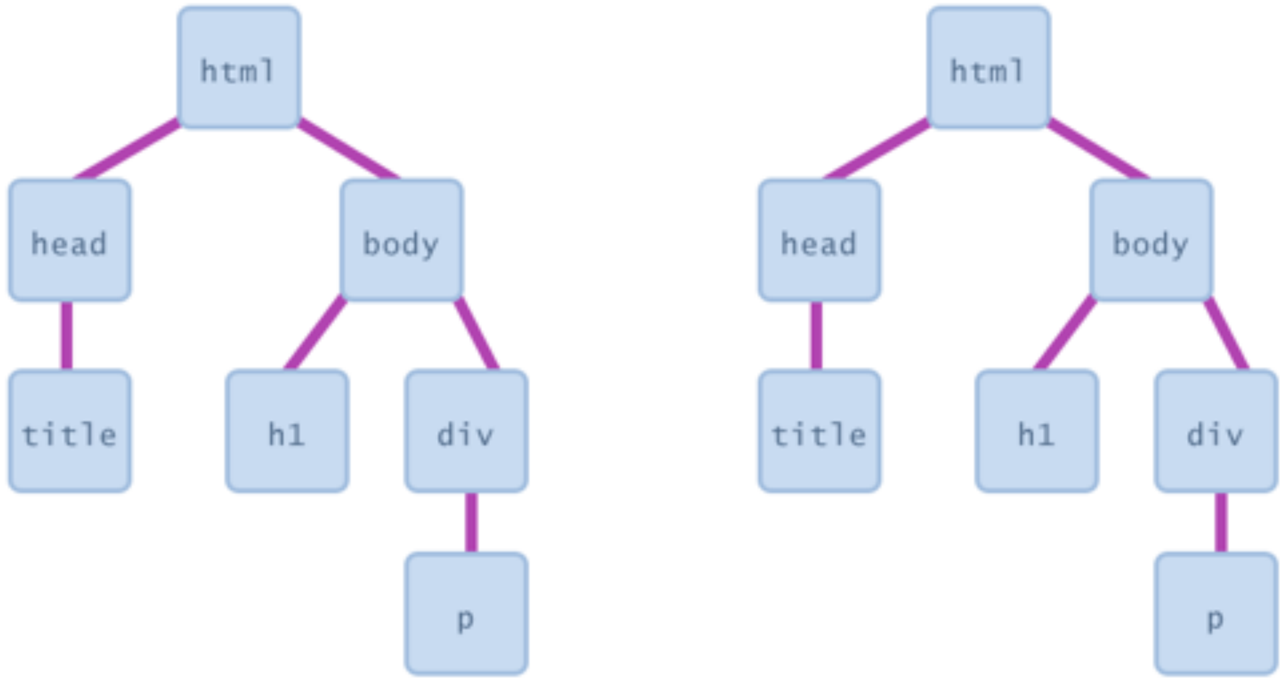
Comparing results

Screenshots



mostly manual

Document Object Models



Can be automated. Tricky, DOM APIs are different across browsers

Webmate

http://www.st.cs.uni-saarland.de/webmate/