



# Patterns for Continuous Test Automation with Canoo WebTest.

**Nate Oster, Number Six Software**



**webtest**



## Who's this Nate Oster guy?

- **Player-Coach with Number Six Software**
- **Emphasis on helping teams adopt Agile testing practices**
- **Agile Alliance Member**
- **RUP-certified Consultant**
  
- **Committer, Eclipse Process Framework**
- **Content Lead, OpenUP Test Discipline**



[noster@numbersix.com](mailto:noster@numbersix.com)

**703.930.4100 (m)**



## **6 Testing is about Feedback.**

- **Two levels of Testing**
- **Developer Tests**
  - *Unit and Integration level*
  - *Isolate and drive smallest pieces*
  - *Design focused*
- **Acceptance Tests**
  - *System level*
  - *Requirements focused*



## **6 Testing is about Feedback.**

- **Testers specify Test Cases as logical “conditions of satisfaction,” in parallel with requirements.**
  - Tests are “Intent.”
  - They are *specifications* of what the system has to do to satisfy a requirement.
  - “Test First.”



## **6 Testing is about Feedback.**

- **Test Scripts are executable.**
  - Automation? Yes, please.
  - They might be written before or after the solution is implemented.
  - TDD is ideal, but this decision is often technology dependent.



## **6 Why extensive functional test automation?**

- **Enables more frequent feedback**
- **Provides a built-in safety net (Stop-the-Line mentality)**
- **Better velocity: The power to test is the power to *change***
- **Provide the executable definition of “correctness”**
- ***Critical* to agile testing, or test cycles become too long.**

**What other benefits can you think of?**



## Just-in-Time Test Specification.

- **There is no “Test Phase”**
- **Test Scripts are fully specified just in time for immediate goals.**
- **Follows the Lean principle of “defer commitment to the last responsible moment.”**
- **Why wait?**

**Less time for tests to get stale.**

**Throw out your “test phase” and start building working, tested software! Fly! Be free!**



## What is Canoo WebTest?

- **A popular open-source testing tool for web applications**
- **An extension to Ant, the Java build tool.**
- **You write Test Scripts directly in XML (using any editor)**

```
<webtest name="check that WebTest is Google's top 'WebTest' result">  
  
<invoke url="http://www.google.com/ncr" description="Go to Google (in English)"/>  
<verifyTitle text="Google" />  
<setTextField name="q" value="WebTest" />  
<clickButton label="I'm Feeling Lucky" />  
<verifyTitle text="Canoo WebTest Homepage" />  
  
</webtest>
```





# It's a powerful tool, but easy to start with.

- Under the hood, it uses HTMLUnit to simulate a browser.
- Extensive support for Javascript with Rhino.
- Helps resolve the mismatch between OO and what is essentially a procedural programming task.
- Clean reporting and results browsing.

## webtest

WebTest Test Report Tests started at Thu Sep 06 15:48:04 CEST 2007

### Result Summary

WebTests	#	%	Graph	Secs	#	%	Histogram
✓	1	100		0 - 1	1	50	
✗	0	0		1 - 3	1	50	
<b>Sum</b>	<b>1</b>	<b>100</b>		3 - 5	0	0	
Steps	#	%	Graph	5 - 10	0	0	
✓	5	100		10 - 30	0	0	
✗	0	0		> 30	0	0	
⚠	0	0		<b>Sum</b>	<b>2</b>	<b>100</b>	
<b>Sum</b>	<b>5</b>	<b>100</b>		<b>Avg</b>		<b>711 ms</b>	

### Server Roundtrip Timing Profile

✓ check that WebTest is Google's top 'WebTest' result

Test started at Thu Sep 06 15:48:04 2007, lasting 1431 ms.

#	Result	Name	Parameter	Duration
1	✓	invoke	url http://www.google.com/ncr	1130
			Go to Google (In English)	
			Resulting page	
2	✓	verifyTitle	text Google	3
3	✓	setInputField	name q value WebTest	6
4	✓	clickButton	label I'm Feeling Lucky	292
			Resulting page	
5	✓	verifyTitle	text Canoo WebTest Homepage	0

[Back to Test Report Overview](#)



## Yeah Yeah Yeah! Let's take a look!

- **Run the demo suite from a command line**
- **View the results in any browser**
  - Easy inspection
  - “Movie playback”



## WebTest uses a “declarative-procedural” style.

- **Declarative**
  - Expect specific outcome from specific input.
  - Example:  $2+5=7$  Not:  $x+y=z$
  - Don't put business logic in the tests!
  - Keep tests as thin as possible.
  
- **Among agilists, considered a Good Thing.**



## WebTest uses a “declarative-procedural” style.

- **Procedural**
  - A sequence of steps
  - Similar to traditional, manual test scripts
  - Follows the pattern of how an end-user actually executes the feature in a step-by-step way.
  
- **More controversial among testing thought leaders.**
  - Tests as executable requirements specifications
  - Procedural tests can be too verbose and lack context



## Challenges with this approach:

- **Procedural tests can be long**
  - It can be unclear what is unique about each script
  - Not effective as executable requirements *by themselves*
- **Thought leaders see move away from “imperative” to a terse “declarative” style of executable requirements specifications**
  - Use Domain-Specific Languages (DSLs)
  - Enable Functional Test Driven Development (FTDD)
  - See Jennitta Andrea’s article, “Envisioning the Next Generation of Functional Testing Tools” (IEEE Software, May/June 2007).
  - Brian Marick, <http://www.testing.com/writings.html>
- **Currently, there appears to be weak tooling around these ideas.**



**6** Though imperfect, the procedural style has good real-world results.

- **FTDD is a promising leading edge technique, but few teams have these skills at present.**
- **Procedural test scripts are familiar:**
  - Directly convert majority of manual scripts
  - WebTests are XML, so you can transform them to any format for enhanced readability
  - Retire the manual script for good!
- **On a recent project we had:**
  - 1200 WebTest Scripts, 4 Test Engineers
  - 92% functional test automation
  - Ran in about 45 minutes
  - We were sneaky and said it took 4 hours. And it did. Sort of.



## Other Styles of Test Scripts?

- **Script-driven** ← **boo! hiss!**
  - “Intelligent” scripts interpret GUI
  - Hallmark is lots of business logic and flow of control
- **Data-driven**
- **Table-driven**
  - Fit/FitNesse
  - Some Selenium approaches
  - Purely declarative DSLs
- **Exploratory** ← **woo!**
- **And a cast of thousands!**



## Data-driven scripts in WebTest are simple.

- **Data-driven tests just run one script repeatedly with different data**
- **Use the `<dataDriven>` task.**
- **The script captures the *structure* of the test condition**
- **Perfect for:**
  - **Business rules**
  - **Field validations**
  - **Fault injection**
  - **Boundary and exception handling**
  - **Wreaking havoc on your hard disk**

**Demo!**

<http://opensource.basehaus.com/webtest/screenscasts/data-driven-webtest.htm>





## Why not use record-playback?

- **Record-Playback is a great way to quickly record many scripts**
- **A horrible way to *maintain* them**
- **Tends toward lots of duplication**
  - Removing duplication means factoring out common code
  - Focus on what's unique about each test
  - Reduces need to record long sequences in the first place
  - Experienced users write code directly.
- **Eventually, recorded “objects” just get in the way.**

**Pick your metaphor: Fool's Gold? Quicksand? Snake Oil?**



**Maintainability is the key to extensive functional automation.**

- **Modularize scripts and ruthlessly drive out duplication:**
  - Follow the rule: “two strikes and you’re out!”
  - Refactor scripts as you work. Avoid planning paralysis.
- **Couple to the GUI as loosely as possible:**
  - In some cases, avoid it altogether (Fit/FitNesse)
  - Specify strict expectations, not unrelated UI details.
- **Use XPath for flexible selection of HTML elements**
  - Always use relative XPath expressions
  - Get free tools: Firefox, XPather, View Formatted Source, View Source Chart.
  - Good: `//input[@name='as_rq']`
  - Bad and Ugly:  
`/html/body/form/font/div[@id='__vpps_16']/table/tbody/tr/td/div[@id='__vpps_17']/table/tbody/tr[1]/td/div[@id='__vpps_18']/table/tbody/tr/td[3]/div[@id='__vpps_19']/table/tbody/tr[1]/td/input`



## Next Steps...

- **Connect your Test Suites to your CI system**
  - Use at least an overnight build
  - WebTests are just Ant files, so integration is generally simple
  - Add source control update and commit results
  - Email results to team members
- **Extend with Groovy scripting**
  - You can write extensions and macros where procedural doesn't make sense (like sorting)
  - You have full access to the HTMLUnit object model
  - You can write test scripts in pure Groovy





## Next Steps...

- **Download WebTest at <http://webtest.canoo.com>**
- **Try out the demos**
- **See the screencasts**  
<http://opensource.basehaus.com/webtest/screencasts/>
- **Join the mailing list**



# Questions? Anyone? Llama?

Nate Oster

[noster@numbersix.com](mailto:noster@numbersix.com)

703.930.4100 (m)

