Let’s Wub

Wub tutorial

jos.decoster@gmail.com

What is Wub?

- HTTP 1.1 Webserver
- Written by Colin McCormack
- Successor of tclhttpd
- 100% Tcl Web application framework
- Heavy user of recent Tcl features, which pushed development of Tcl 8.6 forward:
  - dicts
  - coroutines
  - tclOO
  - Zlib
- Domain based (packages grouping functionality)

Wub in action!

- Tcler's wiki runs on Wub since 2007

  http://wiki.tcl.tk
- Wubchain (web interface to Tcler's chat)

  http://wiki.tcl.tk:30008
- Intranet application to manage tool releases

Requirements?

- Tcl 8.6 (dicts, coroutines, tclOO, zlib)
- Tcllib 1.11
- Unix or Windows

Getting Wub

- Wub's hosted at Google

  http://code.google.com/wub
- Releases at

  http://code.google.com/p/wub/downloads
- Checkout via SVN:

```
svn checkout http://wub.googlecode.com/svn/trunk Wub
```

- Available together with required Tcllib modules as part of WubWikit

  http://code.google.com/p/wubwikit

Documentation

- Wub's wiki page:
  http://wiki.tcl.tk/Wub

- Wub documentation:
  http://wiki.tcl.tk/_wub/

- This Tutorial

Tutorial goals

- Overview different parts of Wub
- Step-by-step introduction of File, Mason, Direct, Nub, Ini, jQuery, … domains
- Wub API: utilities to generate HTML, cache, handle queries, convert response types, forms, …

Tutorial examples

- Get examples at:

  http://code.google.com/p/wubwikit/downloads

- Run as:

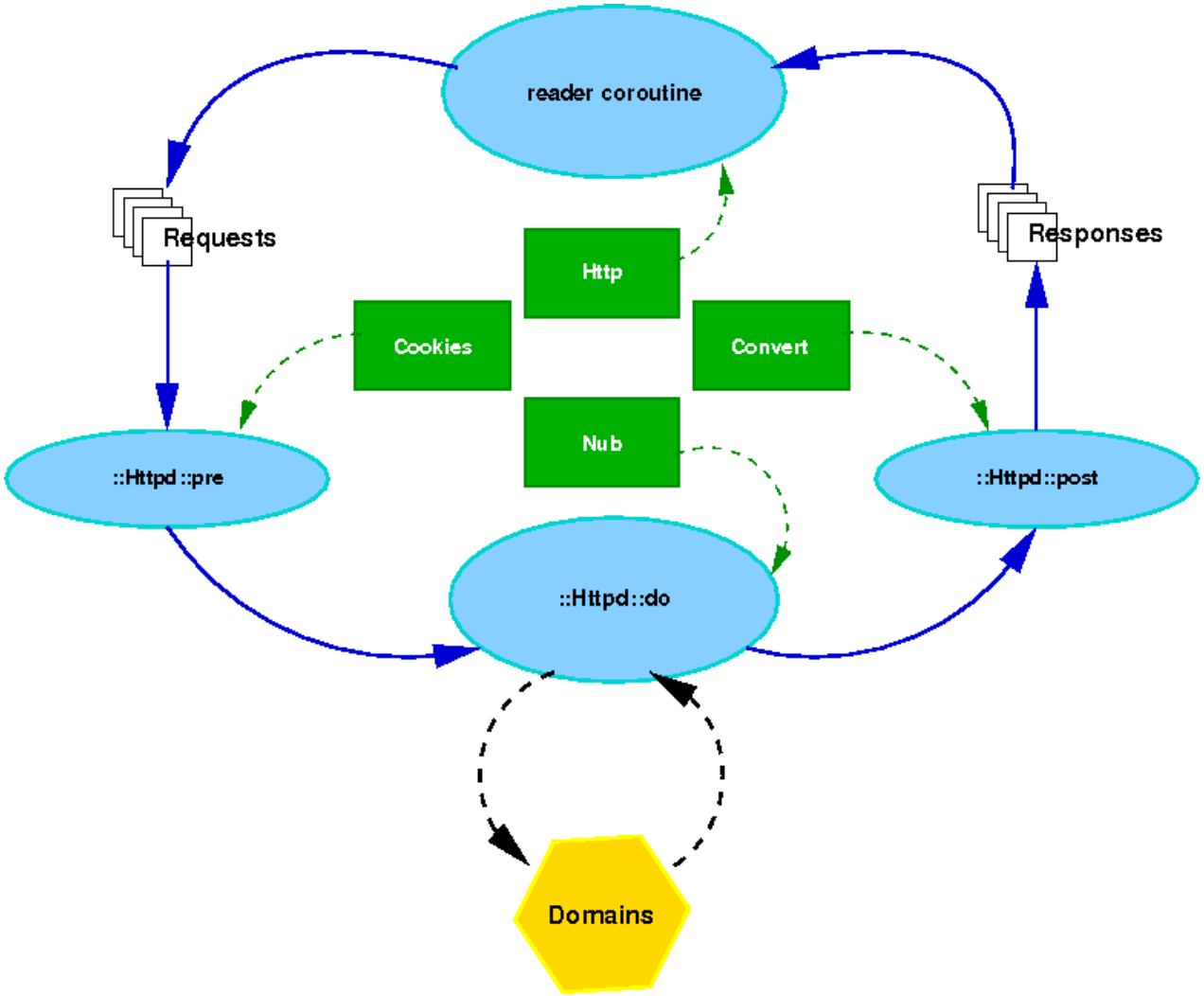
  `tclsh ex.tcl` (make sure it can find tcllib and Wub)

  ../Wub and ../tcllib are added to auto_path by each example

- Point browser to:

  http://localhost:8080

Wub's architecture



Wub's architecture

- Httpd module converts HTTP client request into *request* dicts.
- Caching and blocking based on *request* dict
- *Request* dicts are transformed into *response* dicts by the web application using Wub utilities and domains.
- *Response* dicts are send back to the client by the Httpd module (protocol / non-protocol fields in dict)

Example 1: Wub out-of-the-box

- Running the default setup, start the webserver with these lines:

```
package require Site
Site start home .
```

- What's being served?
  - Files from html, images, css and scripts subdirectories of ./docroot

Example 1 (continued)

- How are these file served:
  - On port 8080
  - Using caching
  - The html, images, css and scripts subdirectories are served as root in the URL

On server	On client	Expires
./docroot/html/file.html	http://myurl:8080/html/file.html	tomorrow
./docroot/images/file.gif	http://myurl:8080/images/file.gif	next week
./docroot/css/file.css	http://myurl:8080/css/file.css	tomorrow
./docroot/scripts/file.js	http://myurl:8080/scripts/file.js	tomorrow

Example 1a: adding some Nub

- Nub is a configuration utility for Wub
- Specify redirect from root URL to ./docroot/html/ex.html in ex.nub:

```
redirect / /html/ex.html
```

- Start Wub with these lines:

```
package require Site
Site start home . nubs ex.nub
```

Example 1a: Nub redirect syntax

```
redirect <from url> <to url>
```

Argument	Description
`from url`	URL for which to send a redirect.
`to url`	URL to redirect to

Example 2: the File domain

- Map URL's to file system hierarchy
- Making File domains explicit in ex.nub:

```
domain /css/     {File css}     root [file join . docroot css]     expires tomorrow    nodir 1
domain /images/  {File images}  root [file join . docroot images]  expires "next week" nodir 1
domain /scripts/ {File scripts} root [file join . docroot scripts] expires tomorrow    nodir 1
domain /html/    {File html}    root [file join . docroot html]    expires tomorrow    nodir 1
```

- Add new File domain:

```
domain /disk/ {File disk} \
   root       / \
   indexfile  index.* \
   hide       {^([.].*)|(.*~)|(\#.*)$} \
   redirdir   1 \
   expires    tomorrow \
   dateformat "%Y %b %d %T" \
   nodir      0
```

- Add Rewrite to ex.nub:

```
rewrite {/[^/]+[.]html} {/html/[file tail [dict get $r -path]]}
```

Example 2: Nub domain syntax

```
domain <url> <list of domain_name and name> <args>
```

Argument	Description
`url`	Url to be processed with the specified domain
`domain_name`	name of domain to use (File, Mason, Direct, jQ, …)
`name`	Name of this domain usage
`args`	Domain specific arguments

Example 2: File domain arguments

Argument name	Description	Default value
`root`	File-system root directory of File domain.	
`indexfile`	Name of the file which stands for a directory, such as `index.html`. The contents of that file will be returned in stead of the directory listing.	`index.*`
`hide`	A regular expression to hide temp and other uninteresting files (default hides .* *~ and #*).	`^([.].*)|(.*~)$`
`redirdir`	Should references to directories be required to have a trailing /?	1
`expires`	A tcl clock expression indicating when contents expires.	0
`dateformat`	A tcl clock format for displaying dates in directory listings.	`%Y %b %d %T`
`nodir`	Don't allow the browsing of directories (default: 0 - browsing allowed).	0

Example 2: Nub rewrite and redirect syntax

```
rewrite <regexp> <script>
```

Argument	Description
`regexp`	Regular expression to select an URL to be transformed
`script`	Script to be called to transform the URL (evaluated in Nub namespace)

Example 3: Mason domain

- A File like domain
- Mapping URL's to file system hierarchy
- Provides templating by applying [subst] on .tml files
- Pre/post filtering of request and responses
- Adding Mason domain to ex.nub

```
domain /mason/ {Mason mason} root [file join . docroot mason]
```

Example 3: Mason arguments

Argumment name	Description	Default value
`root`	File-system root for Mason domain.	
`ctype`	Default content type of returned values.	`x-text/html-fragment`
`hide`	Regular expresion to detect files to hide.	`^([.].*)|(.*~)$`
`indexfile`	Name of the file which stands for a directory, such as `index.html`. The contents of that file will be returned in stead of the directory listing.	`index.html`
`expires`	A tcl clock expression indicating when contents expires.	
`functional`	File extension indicating which files will be evaluated.	`.tml`
`notfound`	Template to be evaluated when requested file can't be found.	`.notfound`

Example 3: Mason arguments (cont.)

Argumment name	Description	
`wrapper`	Template to be evaluated with successful response.	`.wrapper`
`auth`	Template to be evaluated before processing requested file.	`.auth`
`nodir`	Don't allow the browsing of	`0`
`dateformat`	A tcl clock format for displaying dates in directory listings.	`%Y %b %d %T`

Example 3: template file

- Is evaluated using [subst]
- Result of evaluation is returned as content
- Alternatively set –content field of response dict available in variable response
- Use the response dict to access the request information (e.g. query information)
- Other files with same name but different extension are also matched to template file, which allows for one template file to provide multiple formats (e.g. test.html, test.txt and test.tml)

Example 3: Pre/Post/Not found filter

- Pre filter (.auth): return code != 200 or set response dict –code to value != 200 to deny access
- Post filter (.wrapper): transforms responses after they have been processed (e.g. set content type)
- Not found (.notfound): is ran when request can't be resolved
- If no filter file found in requested directory, Mason goes up in file system hierarchy until it finds one or reaches its root directory.

Example 4: Direct domain

- Dispatch a URL request to:
  - Commands within a namespace

  or
  - Methods within an TclOO object
- Proc/method names must start with a /
- Adding direct domain to nub:

```
domain /directns/  Direct namespace MyDirectDomain
domain /directoo/  Direct object    $::oodomain
```

Example 4: Direct arguments

Argument name	Description	Default value
`ctype`	Default content type of returned values	`text/html`
`wildcard`	A proc/method to be used when the request URL doesn't match any of the proc's/methods	`/default`

Example 4: basic Direct proc

```
namespace eval MyDirectDomain {
  proc /test { req } {
    dict set req -content "Test for MyDirectDomain“
    dict set req content-type x-text/html-fragment
    dict set req -title "MyDirectDomain: test with query“
    return $req
  }
}
```

Example 4: basic Direct method

```
oo::class create MyOODomain {
  constructor {args} {}
  method /test {req args} {
    dict set req -content "Test for MyOODomain“
    dict set req content-type x-text/html-fragment
    dict set req -title "MyOODomain: test"
    return $req
  }
}

set oodomain [MyOODomain new]
```

Example 4: Query arguments

- Specify query arguments as proc/method arguments with same name as used in the query
- Wub will decode and assign the query arguments.
- Arguments missing in the request passed as empty string
- Use utilities in `Query` package to handle queries

Example 4: armouring

- HTML special characters (<, ‘, …) need to be armoured.
- Use built-in command `armour`.

Example 4: conversions

- Wub converts all content to something permitted by the accept request field (e.g. test/html, text/plain, …)
- The Convert domain has a table of possible conversions
- Conversion notation:

  `.<from>.<to>`
- Basic built-in conversion:

  `.x-text/html-fragment.text/html`

Example 4: Content type x-text/html-fragment

Response dict key	Description
`-content`	Content to be sent to client
`-title`	Title field in header
`-headers`	List of HTML statements puts verbatim in header
`-script`	List of script source and script arguments, converted into `<script>` statements and placed in the header
`-style`	List of style source and style arguments, converted into `<stylesheet>` statements and placed in the header
`-preload`	List of HTML `<script>` statement to be put in header
`-postscript`	List of script source and script arguments, converted into `<script>` statements and placed at end of body
`-postload`	List of HTML `<script>` statement to be put at end of body

Example 4: custom conversions

- Add proc in `conversions` namespace
- Name contains content type from which conversion starts and content type it generates: .<from>.<to>

```
namespace eval ::conversions {
  proc .x-unarmoured-text/html-fragment.x-text/html-fragment { rsp } {
    set rspcontent [dict get $rsp -content]
    if {[string match "<!DOCTYPE*" $rspcontent]} {
      # the content is already fully HTML
      set content $rspcontent
    } else {
      set content [armour $rspcontent]
    }
    return [Http Ok $rsp $content x-text/html-fragment]
  }
}
```

Example 4: other content types

- Other content types can be returned by setting the `content-type` response dict value

```
dict set req content-type text/plain
```

Example 4: generating html

- Wub offers HTML generation commands in `Html` package:

  `<command> ?key value …? contents`

- Is converted into HTML statement:

  `<command key='value' key2='value2' …>`
  `contents`
  `</command>`

<h1>	<h2>	<h3>	<h4>
<p>	<b>	<i>	<tt>
<table>	<tr>	<td>	
<author>	<description>	<copyright>	<generator>
<keywords>	<meta>	<link>	<script>

Example 5: Http commands

- Found in `Http` package
- Send Http responses to client:

```
Http <error-name> <response dict> ?arguments?
```

- Supported error-names (with codes):
  - `Ok` (200)
  - `Moved` (301), `Redirect` (302), `RedirectReferer` (302), `Found` (302), `SeeOther` (303), `Relocated` (307)
  - `Bad` (400), `Forbidden` (403), `NotFound` (404)
  - `ServerError`

Example 5: Caching

- Use `Http` commands to enable/disable caching
  - `NoCache` : indicate response contents can not be cached
  - `Cache` : indicate response contents may be cached
  - `Dcache` : indicate response contents may be cached buts revalidation is required

Example 5: Caching Age

- The `Cache` and `Dcache` take age argument:
  - Specified as integer: the age at which the contents expires, expressed in seconds
  - Specified as `clock scan` string: the point in time until which the contents remain valid (tomorrow, next week, 7 june 2009, …)

Example 5: Clear cache

- `Cache clear`: Clear the complete cache. All cached contents is removed from the cache
- `Cache delete <url>`: Remove the contents cached for the specified URL from the cache

Example 6: Forms

- Create form with HTML generation commands from `Form` package
- Specify namespace proc or object method as form `action`
- Use post or get as form method
- As with queries, form entries are translated by Wub to proc/method arguments based on entry names

<form>	<input>	<button>	<hidden>
<textarea>	<option>	<optgroup>	<select>

Example 7: jQuery

- jQuery is a JavaScript library

  http://jquery.com/

- Wub makes it easy to use jQuery and some of its plugins by wrapping it into a File like domain.
- To add jQuery to you application add this statements:

```
set req [jQ jquery $req]
```

Example 7: jQuery

- To run a scripts when jQuery is loaded:

```
set req [jQ ready $req <script>]
```

- To use jQuery plugin:

```
set req [jQ <plugin> <selector> <plugin arguments>]
```

- List of wrapped plugins at http://wiki.tcl.tk/_wub/docs/Domains/JQ

jframe	jtemplates	history	datepicker
timeentry	hint	boxtoggle	tablesorter
multifile	containers	tabs	accordion
resizable	draggable	droppable	sortable
selectable	autogrow	autoscale	tooltip
hoverimage	galleria	gallery	editable
form	validate	autofill	confirm
ingrid	map		

Example 8: Ini parameters

- Configure the webserver
- Uses ini-file syntax
- Per ini section, a dict is create in the Site namespace, keys and values of this dict are taken from the ini section, except wub section keys which become variable in Site namespace
- Pre-defined sections and keys
- User-defined sections and keys

Example 8: Ini parameters

Section	Key	Default value	Description
wub	home	[file dirnam [info script]]	Home of application
	ini	site.ini	Ini file
	globaldocroot	0	?????
	cmdport	8082	Console port
	application	“”	Package to require
	local	local.tcl	Post-init script
	vars	vars.tcl	Pre-init script
listener	-port	8080	Wub listener port
cache	maxsize	204800	Maximum size of object in cache
	high	100	High water mark
	low	90	Low water mark
	weight_age	0.02	Age weight for replacement
	weight_hits	-2.0	Hits weight for replacement

Example 8: Ini parameters

Section	Key	Default value	Description
nub	nubs	nub.nub bogus.nub	List of nub files to process
httpd	logfile	“wub.log”	Log file name
	max_conn	20	Maximum number of connections per IP
	no_really	30	How many time to complain about max_conn exceeded
	retry_wait	20	How long to advise client to wait on exhaustion (in seconds)
	timeout	60000	Idle time tolerated (in milli-seconds)

Example 9: Nub

- Configure Wub (interactively)
- Map URL's to domains

Nub command	Description
`domain <url> <domain> <args>`	Process specified URL with given domain
`redirect <from> <to>`	Redirect <from> URL to <to> URL
`rewrite <regexp> <script>`	Transform URL selected by the regexp into the one calculated by the script
`code <url> <script> ?<content-type>?`	Send result of evaluating the script to client requesting specified URL
`literal <url> <contents> ?<content-type>?`	Send literal contents to client accessing specified URL
`block <list of url's>`	Block IP addresses trying to access specified (glob matched) URL's

Example 10: Suspend/Resume

- Suspend request until response is available:

```
return [Httpd Suspend $req]
```

- Resume suspended request when response became available:

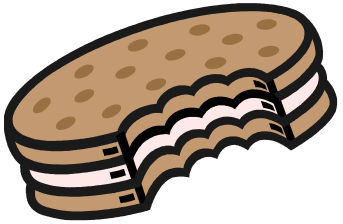
```
Httpd Resume [Http Ok [/test_return_result $req]]
```

- Used as basis for e.g. Wubchain where request for new chat messages from client is suspended until a new chat message arrives from another chatter

Example 11: Cookies

- Cookies are stored in the request dict
- The Cookies package can be used to manipulate the cookies as stored in the request dict:

```
set cdict [dict get $req -cookies]
set vdict [Cookies fetch $cdict -name my_cookie]
set value [dict get $vdict -value]
```



Example 12: Command port

- Make connection to running Wub using telnet from localhost
- Stop, configure or query running server
- Set command port with cmdport ini parameter in wub section

More domains

- CGI
- Coco: co-routine domain
- Commenter: parse Tcl file and show comments
- Dub: database domain (based on metakit)
- Honeypot: catch spiders and link harvesters
- Login: simple login account handling
- RAM: convert contents of an array into a domain
- Repo: file repository (e.g. half bakery)
- Session: session handling
- Tie: mapping of namespace variables to URLs
- Tub: direct domain to store arbitrary data
- Woof: Wub's interface to Woof (Web oriented object framework by Ashok P. Nadkarni)

More utilities

- Auth: authentication (rfc 2617)
- captcha: create captcha's using `convert`
- Color: color manipulation
- Debug: debug info logger
- Report: convert a dict or csv data into an HTML table
- scgi: scgi interface
- Sitemap: create google sitemap from dict
- UA: user-agent detection
- Url: URL manipulation

Reporting bugs, feature request, …

- Bugs and feature request for Wub are best reported at

  http://code.google.com/p/wub/issues

- Bugs and feature request for this tutorial is best reported at

  http://code.google.com/p/wubwikit/issues

Conclusion

- Wub is a new environment for Web application development using Tcl/Tk
- Reliable (see Tcler's wiki)
- Under active development
- Offering lots of useful domains and utilities to make a developer's life easier
- When looking for a Tcl solution and thinking about using tclHTTPd , consider using Wub

Questions?