Rivet: webscripting for Tcl'ers

Application Development with Rivet and
the Apache HTTP Web Server

Rivet Templates (.rvt files)

```
<html>
    <head>
        <?
            set page_title "Rivet Script Environment"
            set page_keywords [list apache web scripting rivet]
        ?>
        <meta name="keywords" content="<?= [join $page_keywords ,] ?>" />
        <title><?= $page_title ?></title>
    </head>
    <body>
        <div>
            <h1><?= $page_title ?></h1>
            <?
                load_env environment
            ?>
            <table>
                <tr><th>Variable</th><th>Value</th></tr>
                <?
                    foreach en [array names environment] {
                        puts "<tr><td>$en</td><td>$environment($en)</td></tr>"
                    }
                ?>
            </table>
        </div>
    </body>
</html>
```

- In a Rivet Template code is placed within <?...?>

Pure Tcl Scripts

```
# file_download.tcl
# Code example for the transmission of a pdf file.

    if {[var exists pdfname]} {
        set pdfname [var get pdfname]

        set pdf_full_path [file join $pdf_repository ${pdfname}.pdf]
        if {[file exists $pdf_full_path]} {

            headers type                    "application/pdf"
            headers add Content-Disposition "attachment; filename=${pdfname}.pdf"
            headers add Content-Description "PDF Document"

        # As a matter of fact the following lines could be any datasource

            set paper       [open $pdf_full_path r]
            fconfigure      $paper -translation binary
            set pdf         [read $paper]
            close $paper

            headers add Content-Length  [string length $pdf]

        # Let's send the actual file content

            puts $pdf

        } else {
            parse pdf_not_found_error.rvt
        }
    } else {
        parse parameter_not_defined_error.rvt
    }
```

Templates vs Pure Tcl Scripts

- Rivet templates are evaluated within the ::request namespace
  - The ::request namespace gets deleted and recreated on every request

- Tcl scripts are evaluated within the global namespace
  - Variables, arrays and object instances are preserved across requests
  - Potential risks

Apache 'prefork' MPM

- Apache has an extremely modular architecture
  - Multi-Processing modules are responsible for binding network port, accepting request and dispatching them to agents for processing
- Rivet currently supports only the 'prefork' MPM.
  - Agents responding requests are child processes forked by the main server process
  - Child processes are created and terminated depending on the server workload
  - Each child process exits after having served MaxRequestsPerChild requests

Rivet Apache Commands

- Rivet introduces 3 new commands (contexts) to Apache configuration
  - RivetServerConf
    - Global directives valid for the whole server.
  - RivetUserConf
    - They appear in .htaccess files
  - RivetDirConf
    - Directives valid for the directory tree for which they are specified. Meaningful when in <Directory ...> sections

  Example:

  ```
  RivetServerConf UploadDirectory /tmp
  ```

Rivet Application Control

- Rivet Directives: 4 categories
  - Child Process Creation and Termination
  - Request Processing
  - Error and Exception Handling
  - Module Parameters

Child Process Initialization/Termination

- GlobalInitScript
- ChildInitScript
- ChildExitScript

Child Process initialization:

- External Packages
- Database Connections
- In-Memory Data

Request Processing

- BeforeScript
- AfterScript

- Request processing is done by concatenating 3 scripts
  - The script configured by the BeforeScript option
  - The script in the .rvt/.tcl file referenced in the URI
  - The script configured by the AfterScript option

Error and Exception Handling

- ErrorScript
- AbortScript

  - ErrorScript is run when a script fails
    - The script is fired when errors go uncaught
    - Avoids useless disclosure of code shown in the backtrace

  - AbortScript is triggered by calling 'abort_page <code>'
    - The code argument passed to abort_page can later be retrieved by calling abort_code

Virtual Hosts

- Virtual Hosts are multiple web sites running within a single instance of a webserver
  - Most of Rivet configuration scripts go within <VirtualHosts ...>...</VirtualHost> declarations
  - Multiple applications require almost certainly separate interpreters:

    RivetServerConf SeparateVirtualInterps On
    - Each Virtual Host will get its slave interpreter
    - GlobalInitScript is ignored: you can initialize interpreters by means of ChildInitScript

Server Initialization

- ServerInitScript

- Available in Rivet 2.1.0 (currently 'trunk' in svn repository)
  - Runs when Apache is still a single process
  - The right place to do tasks that need to avoid concurrency
  - The interpreter created here is cloned by the fork operation into the child processes:
    - No need to load packages in child processes
    - High speed initialization if SeparateVirtualInterps is off

FlightAware.com

- Most successful Rivet based Application
  - Currently the #1063 most popular site on the Internet in the US (Quantcast.com)
  - From May 25th to June 25th 2012, 2.3 million people visited flightaware.com 4.6 million times and viewed about 90 million pages
  - 1.4 million registered users
  - Additional data are delivered to customers using their FlightXML API and MAPI (Mobile API)
- Every single piece of information or web page is delivered using Rivet

FlightAware.com

Getting Involved

- Our website http://tcl.apache.org/rivet/
- Everyone is invited to join us on our mailing list

rivet-dev@tcl.apache.org