Fossil: Reloaded

Full Tcl Integration

Why?

- TH1 provides built-in scripting capabilities for Fossil; however, it is a small subset of “full” Tcl.

- There was a significant demand for advanced features that could be easily scripted if full Tcl was available (e.g. email notifications for check-ins).

Replace TH1?

- No. It was decided very early on that replacing TH1 would be too much of a break with backwards compatibility.

- Instead, a dual-language solution was developed that involves seamless bi-directional “bridging” of TH1 and Tcl.

Requirements

- Works with Tcl 8.4 or higher. Supported on all platforms where dynamic loading is available.

- Typically, the new “private stubs” mechanism should be used (created by Jan Nijtmans).

Compilation (Linux)

- From a Fossil source checkout:
- If automatic detection of Tcl library works:

```
./configure --with-tcl=1 --with-tcl-stubs=1
```

- Otherwise, something like this:

```
./configure --with-tcl=1 --with-tcl-private-stubs=1
```

Private Stubs?

- They allow Fossil to be compiled with Tcl integration support without needing to locate an actual Tcl (or stubs) library at compile-time.

Recommended Compile Options

- This will configure Fossil to use the in-tree zlib, enable OpenSSL and JSON, and enable Tcl integration:

```
./configure --with-zlib=compat/zlib --with-openssl=auto --json=1 --with-tcl=1 --with-tcl-private-stubs=1
```

Compilation (Windows)

- Using MSVC is recommended.
- Using the “buildmsvc.bat” tool from a Fossil source checkout is recommended, e.g.:

```
cd win
buildmsvc.bat fossil.exe FOSSIL_ENABLE_TCL=1
```

Great, it built, now what?

When you have a Fossil binary with Tcl integration enabled, there is [at least] one more step to be able to evaluate Tcl from within Fossil, which is to enable the “tcl” setting using a command similar to:

```
fossil settings tcl 1
```

This can be set per-repository or globally. It is disabled by default.

Can I use Tk?

- Yes.
- However, since Fossil does not currently service the Tcl Event Loop, using **vwait** will almost certainly be required.

Using Tcl from Fossil

- After Tcl integration has been enabled at compile-time and runtime (via the “tcl” setting), it can be used anywhere it is legal to use TH1.
- This is accomplished using several new TH1 commands, namely **tclEval**, **tclExpr**, and **tclInvoke** (all defined in the source file “**th_tcl.c**”).

TH1 ←→ Tcl bridging

- On the TH1 side: **tclEval**, **tclExpr**, **tclInvoke**
- These can be used anywhere TH1 is legal.

- On the Tcl side: **th1Eval**, **th1Expr**
- Useful for accessing Fossil / TH1-specific functionality from a Tcl script.

Simple Example

- From the “Enhanced Default” skin:

```
<th1>
proc getTclVersion {} {
  if {[catch {
    tclEval info patchlevel
  } tclVersion] == 0} {
    return "<a href=\"http://www.tcl.tk/\">Tcl</a> version $tclVersion"
  }
  return ""
}
set tclVersion [getTclVersion]
</th1>
```

“Advanced” Example
(from the test suite)

WARNING: This may be unsafe due to the possibility of multiple copies of the SQLite library in the process (e.g. on Linux). Also, please do NOT write to the Fossil databases.

```
tclInvoke set repository_name [repository 1]

set userCount [tclEval {
  package require sqlite3
  sqlite3 db $repository_name -readonly true
  set x [db eval {SELECT COUNT(*) FROM user;}]
  db close
  return $x
}]
```

Not Dangerous Enough?

- TH1 Hooks: Enables a TH1 (optionally with Tcl) script to be evaluated just prior to [and/or just after] any Fossil command or web page.
- Redefine or log commands.
- Perform pre-processing and/or post-processing for commands.
- Command line arguments to the command can be seen by the script.

How to enable TH1 Hooks

- At compile-time using “./configure”:
  - Add “**--with-th1-hooks=1**” to the command.
- At compile-time using “buildmsvc.bat”:
  - Add “**FOSSIL_ENABLE_TH1_HOOKS=1**” to the command.
- At runtime, enable the “**th1-hooks**” setting.

TH1 Hooks Example

- Define inside “.fossil-settings/th1-setup” file inside of a test repository:

```
proc command_hook {} {
  if {$::cmd_name eq "timeline"} {
    puts $::cmd_args\n
    puts "Tcl [tclEval {info patchlevel}]\n\n"
  }
}
```

Still Need More Danger?

- TH1 Docs: Enables TH1 scripts to be evaluated when a “th1” file is viewed as embedded documentation.
- Disabled by default at compile-time and runtime.
- **WARNING: This option should only be enabled on a repository where all users with check-in privileges are “implicitly trusted”.**

How to enable TH1 Docs

- Be 100% sure that you want this.
- At compile-time using “./configure”:
  - Add “**--with-th1-docs=1**” to the command.
- At compile-time using “buildmsvc.bat”:
  - Add “**FOSSIL_ENABLE_TH1_DOCS=1**” to the command.
- At runtime, enable the “**th1-docs**” setting.

TH1 Docs Example

If this is “**example.th1**” in the root of the repository checkout, it could be viewed using the URI:

`http://localhost:8080/doc/ckout/example.th1`

```
<th1>
reinitialize 5; # enable Tcl, if configured.
html "Tcl 2 ** 2 == [tclEval {expr 2 ** 2}]"
</th1>
```

Show “fileStat.th1” now…

- This is a more complete example of how to make use of TH1 Docs and Tcl integration.

Questions?

Contact

- Joe Mistachkin joe@mistachkin.com