Technical Manual of the `US-Common` Layer

Organisation: Copyright (C) 2019-2020 Olivier Boudeville

Contact: about (dash) us-common (at) esperide (dot) com

Creation date: Saturday, May 2, 2020

Lastly updated: Saturday, May 2, 2020

Status: Work in progress

Version: 0.0.1

Dedication: Users and maintainers of the `US-Common` layer.

Abstract: The role of the US-Common layer (part of the Universal Server project) is to provide base elements on which the various *Universal Services* are built, notably:

- the Universal Server itself: see us-main
- the Universal Webserver: see us-web

We present here a short overview of these services, to introduce them to newcomers. The next level of information is to read the corresponding source files, which are intensely commented and generally straightforward.

Table of Contents

Overview

The US-Common layer is the basis (lowest-level) of the Universal Server project.

Its purpose is to provide base elements on which the various *Universal Services* are built, notably:

- the Universal Server itself: see us-main

- the Universal Webserver: see us-web

We present here a short overview of these services, to introduce them to newcomers.

The next level of information is to read the corresponding source files, which are intensely commented and generally straightforward.

Layer Stack

From the highest level to the lowest, a software stack involving US-Common usually is like:

- applicative layer such as us-main, us-web, etc.

- US-Common (this layer)

- Ceylan-Traces (for advanced runtime traces)

- Ceylan-WOOPER (for OOP)

- Ceylan-Myriad (as an Erlang toolbox)

- Erlang (for the compiler and runtime)

- GNU/Linux

The shorthand for `US-Common` is `uc`.

Licence

`US-Common` is licensed by its author (Olivier Boudeville) under the GNU Affero General Public License as published by the Free Software Foundation, either version 3 of this license, or (at your option) any later version.

This allows the use of the US-Common code in a wide a variety of software projects, while still maintaining copyleft on this code, ensuring improvements are shared.

We hope indeed that enhancements will be back-contributed (ex: thanks to merge requests), so that everyone will be able to benefit from them.

Current Stable Version & Download

As mentioned, the single, direct prerequisite of US-Common is Ceylan-Traces, which implies in turn Ceylan-WOOPER, then Ceylan-Myriad and Erlang, version 22.1 or more recent[1].

Using Cutting-Edge GIT

This is the installation method that we use and recommend; the US-Common `master` branch is meant to stick to the latest stable version: we try to ensure that this main line always stays functional (sorry for the pun). Evolutions are to take place in feature branches and to be merged only when ready.

Once Erlang is available, it should be just a matter of executing:

```
$ git clone https://github.com/Olivier-Boudeville/Ceylan-Myriad myriad
$ cd myriad && make all && cd ..

$ git clone https://github.com/Olivier-Boudeville/Ceylan-WOOPER wooper
$ cd wooper && make all && cd ..

$ git clone https://github.com/Olivier-Boudeville/Ceylan-Traces traces
$ cd traces && make all && cd ..

$ git clone https://github.com/Olivier-Boudeville/us-common
$ cd us-common && make all
```

Running a corresponding test just then boils down to:

```
$ cd test && make class_USScheduler_run CMD_LINE_OPT="--batch"
```

Should LogMX be installed and available in the PATH, the test may simply become:

```
$ make class_USScheduler_run
```

---

[1]Note that, in the Ceylan-Myriad repository, we have a script to streamline the installation of Erlang, see install-erlang.sh; use `install-erlang.sh --help` for guidance.

Using OTP-Related Build/Runtime Conventions

As discussed in these sections of Myriad, WOOPER and Traces, we added the (optional) possibility of generating a US-Common *OTP application* out of the build tree, ready to be integrated into an *(OTP) release*. For that we rely on rebar3, relx and hex.

Unlike Myriad (which is an OTP *library* application), US-Common is (like WOOPER and Traces) an OTP *active* application, meaning the reliance on an application that can be started/stopped (`us_common_app`) and a root supervisor (`us_common_sup`).

As for Myriad, WOOPER and Traces, most versions of US-Common are also published as Hex packages.

For more details, one may have a look at:

- rebar.config.template, the general rebar configuration file used when generating the US-Common OTP application and release (implying the automatic management of Myriad and WOOPER)

- rebar-for-hex.config.template, to generate a corresponding Hex package for US-Common (whose structure and conventions is quite different from the previous OTP elements)

- rebar-for-testing.config.template, the simplest test of the previous Hex package: an empty rebar project having for sole dependency that Hex package

Support

Bugs, questions, remarks, patches, requests for enhancements, etc. are to be reported to the project interface (typically issues) or directly at the email address mentioned at the beginning of this document.

Please React!

If you have information more detailed or more recent than those presented in this document, if you noticed errors, neglects or points insufficiently discussed, drop us a line! (for that, follow the Support guidelines).

Ending Word

Have fun with US-Common!