The Implementation

Sven van der Meer

Table of Contents

for SKB-Framework v0.0.1, November 15, 2018

1. skb-framework

The `skb-framework` is the main entry point to the framework. It realizes two things in one. First, it is an application itself. Second, it is a start script that can be used by other applications to start the framework.

At the start, it checks the setting `__FW_LOADER_FLAVOR` (line 1 in the source block below). If this variable is set, then another application wants to start the framework. Otherwise, the `skb-framework` is the application. In the later case, the script sets required variables for the loader (lines 3-5 below):

- `__FW_LOADER_FLAVOR` - the flavor of the application, here `SF`
- `__FW_LOADER_SCRIPTNAME` - the name of the script (application)
- `__FW_LOADER_APPNAME` - the application name

The next step is to find the framework installation. The script tries the variable `SF_HOME` first, `readlink` first (lines 8-13), if that fails `dirname` (lines 15-19). If all attempts fail, the script terminates with an error (lines 20-24). Otherwise it set `FW_HOME`.

```
if [[ -z ${__FW_LOADER_FLAVOR:-} ]]; then
    ## we should load the framework itself, so SF
    export __FW_LOADER_FLAVOR="SF"
    export __FW_LOADER_SCRIPTNAME="$0"
    export __FW_LOADER_APPNAME="SKB Framework"

    ## try readline to find where we are
    if [[ -z ${SF_HOME:-} ]]; then
        SF_HOME=$(readlink -f $0)
        SF_HOME=${SF_HOME%/*}
        SF_HOME=${SF_HOME%/*}
        export SF_HOME
    fi
    ## try dirname to find where we are
    if [[ -z ${SF_HOME:-} ]]; then
        SF_HOME=$(dirname $0)
        SF_HOME=$(cd $SF_HOME/..; pwd)
        export SF_HOME
    fi
    if [[ -z ${SF_HOME:-} ]]; then
        printf "  unable to set home \$SF_HOME (tried environment, readlink, and
dirname \$0)\n"
        printf "  please set SF_HOME\n\n"
        exit 10
    fi
    export FW_HOME=$SF_HOME
```

If `skb-framework` is used by another application to start the framework, the script only tries to find the framework installation. The mechanism here is the same as explained above: try `FW_HOME` first, then `readlink`, then `dirname`. IF all fails, exit with an error.

```
else
    ## try readline to find where we are
    if [[ -z ${FW_HOME:-} ]]; then
        FW_HOME=$(readlink -f $0)
        FW_HOME=${FW_HOME%/*}
        FW_HOME=${FW_HOME%/*}
        export FW_HOME
    fi
    ## try dirname to find where we are
    if [[ -z ${FW_HOME:-} ]]; then
        FW_HOME=$(dirname $0)
        FW_HOME=$(cd $FW_HOME/..; pwd)
        export FW_HOME
    fi
    if [[ -z ${FW_HOME:-} ]]; then
        printf "  unable to set framework home \$FW_HOME (tried environment, readlink,
and dirname \$0)\n"
        printf "  please set FW_HOME\n\n"
        exit 10
    fi
```

One the framework installation has been found, the script tests if the loader exists.

```
if [[ ! -x $FW_HOME/bin/loader/loader.sh ]]; then
    printf "  did find/set \$FW_HOME, but did not find loader\n"
    printf "  tried $FW_HOME/bin/loader/loader.sh\n\n"
    exit 11
fi
```

When all conditions are satisfied, the script executes the loader handing over all arguments unprocessed.

```
$FW_HOME/bin/loader/loader.sh $*
exit $?
```

2. The Loader

The loader is the script `$FW_HOME/bin/loader/loader.sh`. It is responsible for all initial configuration, loading elements, testing settings and dependencies, processing command line arguments (options), and execute task, scenarios, or the shell. The load process has multiple steps, starting with some initial settings and finished with a cleanup.

The initial settings are shown in the source block below. Line 1 restricts *bash*, providing a safer execution environment. Line 2 allows for extended globbing (finding files recursively with wildecards such as `*/.adoc`). Line 3 takes the current time. This information is later used to calculate how long the load process did take. Line 4 removes an environment setting that prints rather annoying messages when running Java.

```
set -o errexit -o pipefail -o noclobber -o nounset
shopt -s globstar
_ts=$(date +%s.%N)
unset JAVA_TOOL_OPTIONS
```

2.1. Dependencies

The first real action in the load process is the test for core dependencies. The source block below shows how the loader tests for:

- *BASH* version 4 - needed to use associative arrays (or maps)
- GNU *Getopt* - extensively used in the loader and tasks to parse command lines
- *bc* - a calculator used for calculating execution time of tasks and scenarios
- *mktemp* - used to create names for temporary directories, required to store runtime configuration information

```
if [[ "${BASH_VERSION:0:1}" -lt 4 ]]; then
    printf " ==> no bash version >4, required for associative arrays\n\n"
    exit 12
fi
! getopt --test > /dev/null
if [[ ${PIPESTATUS[0]} -ne 4 ]]; then
    printf " ==> getopt failed, require for command line parsing\n\n"
    exit 13
fi
if [[ ! $(command -v bc) ]]; then
    printf " ==> did not find bc, require for calculations\n\n"
    exit 14
fi
if [[ ! $(command -v mktemp) ]]; then
    printf " ==> did not find mktemp, require to create temporary files and
directories\n\n"
    exit 15
fi
```

2.2. Core and default Settings

Once all dependencies are satisfied, the loader realizes core settings:

- If not set, then set `$FW_HOME` (lines 1-4)
- Source the loaders declaration files (line 6). This will create the main configuration map called `CONFIG_MAP` along with the map `CONFIG_SRC`.
- Set core variables in the configuration map (lines 7-25):
  - *FW_HOME* - the home directory of the framework
  - *RUNNING_IN* - set to loader, this will later be changed the `shell` or `task` by the shell and tasks
  - *SYSTEM* - to know in which system the framework is running
  - *CONFIG_FILE* - the SKB configuration file
  - *STRICT* - set strict mode to `off`, at least initially
  - *APP_MODE* - set the default application mode to `use`
  - *PRINT_MODE* - set the default print mode to `ansi` (for ANSI formatted text with colors and effects)
  - Levels - set the levels for loader, shell, and tasks initially to `error`
  - Quiet - set quiet mode for loader, shell, and tasks to `off`, i.e. they are *not* quiet
  - *SCENARIO_PATH* - create an empty path for scenarios
  - *SHELL_SNP* - activate shell prompt

```
if [[ -z ${FW_HOME:-} ]]; then
    FW_HOME=$(dirname $0)
    FW_HOME=$(cd $FW_HOME/../.. && pwd)
fi

source $FW_HOME/bin/loader/declare/_include
CONFIG_MAP["FW_HOME"]=$FW_HOME                      # home of the framework
export FW_HOME
CONFIG_MAP["RUNNING_IN"]="loader"                   # we are in the loader,
shell/tasks will change this to "shell" or "task"
CONFIG_MAP["SYSTEM"]=$(uname -s | cut -c1-6)        # set system, e.g. for Cygwin path
conversions
CONFIG_MAP["CONFIG_FILE"]="$HOME/.skb"              # config file, in user's home
directory
CONFIG_MAP["STRICT"]=off                            # not strict, yet (change with
--strict)
CONFIG_MAP["APP_MODE"]=use                          # default application mode is use,
change with --all-mode, --build-mode, --dev-mode
CONFIG_MAP["PRINT_MODE"]=ansi                       # default print mode is ansi,
change with --print-mode

CONFIG_MAP["LOADER_LEVEL"]="error"                  # output level for loader, change
with --loader-level, set to "debug" for early code debugging
CONFIG_MAP["SHELL_LEVEL"]="error"                   # output level for shell, change
with --shell-level
CONFIG_MAP["TASK_LEVEL"]="error"                    # output level for tasks, change
with --task-level

CONFIG_MAP["LOADER_QUIET"]="off"                    # message level for loader, change
with --lq
CONFIG_MAP["SHELL_QUIET"]="off"                     # message level for shell, change
with --sq
CONFIG_MAP["TASK_QUIET"]="off"                      # message level for tasks, change
with --tq

CONFIG_MAP["SCENARIO_PATH"]=""                      # empty scenario path, set from
ENV or file (parameter)
CONFIG_MAP["SHELL_SNP"]="off"                       # shell shows prompt, change with
--snp
```

2.3. Core Includes

```
source $FW_HOME/bin/api/_include
ConsoleResetErrors
ConsoleResetWarnings

source $FW_HOME/bin/api/describe/_include
source $FW_HOME/bin/loader/init/parse-cli.sh
```

2.4. Application Flavor and Name

```
if [[ -z ${__FW_LOADER_FLAVOR:-} ]]; then
    ConsoleFatal " ->" "interal error: no flavor set"
    printf "\n"
    exit 16
else
    CONFIG_MAP["FLAVOR"]=$__FW_LOADER_FLAVOR
    CONFIG_SRC["FLAVOR"]="E"
    if [[ -z ${CONFIG_MAP["FLAVOR"]} ]]; then
        ## did not find FLAVOR
        ConsoleFatal " ->" "internal error: did not find setting for flavor"
        printf "\n"
        exit 16
    fi

    FLAVOR_HOME="${CONFIG_MAP["FLAVOR"]}_HOME"
    CONFIG_MAP["APP_HOME"]=${!FLAVOR_HOME:-}
    CONFIG_SRC["APP_HOME"]="E"
    if [[ -z ${CONFIG_MAP["APP_HOME"]:-} ]]; then
        ConsoleFatal " ->" "did not find environment setting for application home,
tried \$${CONFIG_MAP["FLAVOR"]}_HOME"
        printf "\n"
        exit 17
    elif [[ ! -d ${CONFIG_MAP["APP_HOME"]} ]]; then
        ## found home, but is no directory
        ConsoleFatal " ->" "\$${CONFIG_MAP["FLAVOR"]}_HOME set as
${CONFIG_MAP["APP_HOME"]} does not point to a directory"
        printf "\n"
        exit 18
    fi
fi

if [[ -z ${__FW_LOADER_SCRIPTNAME:-} ]]; then
    ConsoleFatal " ->" "interal error: no application script name set"
    printf "\n"
    exit 20
else
    CONFIG_MAP["APP_SCRIPT"]=${__FW_LOADER_SCRIPTNAME##*/}
fi
if [[ -z "${__FW_LOADER_APPNAME:-}" ]]; then
```

```
    ConsoleFatal " ->" "interal error: no application name set"
    printf "\n"
    exit 21
else
    CONFIG_MAP["APP_NAME"]=$__FW_LOADER_APPNAME
fi
source $FW_HOME/bin/loader/declare/app-maps.sh
if [[ -f ${CONFIG_MAP["APP_HOME"]}/etc/version.txt ]]; then
    CONFIG_MAP["VERSION"]=$(cat ${CONFIG_MAP["APP_HOME"]}/etc/version.txt)
else
    ConsoleFatal " ->" "no application version found, tried
\$APP_HOME/etc/version.txt"
    printf "\n"
    exit 22
fi
```

2.5. Temporary Directory

```
if [[ ! -z ${TMP:-} ]]; then
    TMP_DIRECTORY=${TMP}/${CONFIG_MAP["APP_SCRIPT"]}
else
    TMP_DIRECTORY=${TMPDIR:-/tmp}/${CONFIG_MAP["APP_SCRIPT"]}
fi
if [[ ! -d $TMP_DIRECTORY ]]; then
    mkdir $TMP_DIRECTORY 2> /dev/null
    __errno=$?
    if [[ $__errno != 0 ]]; then
        ConsoleFatal " ->" "could not create temporary directory $TMP_DIRECTORY,
please check owner and permissions"
        printf "\n"
        exit 23
    fi
fi
if [[ ! -w $TMP_DIRECTORY ]]; then
    ConsoleFatal " ->" "cannot write to temporary directory $TMP_DIRECTORY, please
check owner and permissions"
    printf "\n"
    exit 24
fi
```

2.6. Sneak Preview of CLI Arguments

```
case "$@" in
    *"-D"* | *"--dev-mode"*)
        CONFIG_MAP["APP_MODE"]="dev"
        CONFIG_SRC["APP_MODE"]="O"
        ;;
    *"-B"* | *"--build-mode"*)
        CONFIG_MAP["APP_MODE"]="build"
        CONFIG_SRC["APP_MODE"]="O"
        ;;
    *"-A"* | *"--all-mode"*)
        CONFIG_MAP["APP_MODE"]="all"
        CONFIG_SRC["APP_MODE"]="O"
        ;;
esac
```

2.7. Parameter Declarations

```
DeclareParameters
if ConsoleHasErrors; then printf "\n"; exit 25; fi
source $FW_HOME/bin/loader/init/process-settings.sh
ProcessSettings
```

2.8. Option Declarations

```
if [[ -f ${CONFIG_MAP["CACHE_DIR"]}/opt-decl.map ]]; then
    ConsoleInfo "-->" "declaring options from cache"
    source ${CONFIG_MAP["CACHE_DIR"]}/opt-decl.map
else
    DeclareOptions
    if ConsoleHasErrors; then printf "\n"; exit 26; fi
fi
declare -A OPT_CLI_MAP
for ID in ${!DMAP_OPT_ORIGIN[@]}; do
    OPT_CLI_MAP[$ID]=false
done
```

2.9. Parse Command Line Arguments

```
ParseCli $@
if ConsoleHasErrors; then printf "\n"; exit 27; fi
case "${CONFIG_MAP["PRINT_MODE"]:-}" in
    ansi | text | adoc)
        ConsoleInfo "-->" "found print mode '${CONFIG_MAP["PRINT_MODE"]}'"
        ;;
    *)
        CONFIG_MAP["PRINT_MODE"]=ansi
        CONFIG_SRC["PRINT_MODE"]=
        ConsoleWarn "-->" "unknown print mode '${CONFIG_MAP["PRINT_MODE"]}', assuming
'ansi'"
        ;;
esac
```

2.10. Realize early Exit Options

```
if [[ ${OPT_CLI_MAP["clean-cache"]} != false ]]; then
    ConsoleInfo "-->" "cleaning cache and exit"
    source ${CONFIG_MAP["FW_HOME"]}/bin/loader/options/clean-cache.sh
    exit 0
fi
if [[ ${OPT_CLI_MAP["help"]} != false ]]; then
    source ${CONFIG_MAP["FW_HOME"]}/bin/loader/options/help.sh
    exit 0
fi
if [[ ${OPT_CLI_MAP["version"]} != false ]]; then
    source ${CONFIG_MAP["FW_HOME"]}/bin/loader/options/version.sh
    exit 0
fi
```

2.11. Declarations for Commands and Exit Status Codes

```
if [[ -f ${CONFIG_MAP["CACHE_DIR"]}/cmd-decl.map ]]; then
    ConsoleInfo "-->" "declaring commands from cache"
    source ${CONFIG_MAP["CACHE_DIR"]}/cmd-decl.map
else
    DeclareCommands
    if ConsoleHasErrors; then printf "\n"; exit 28; fi
fi
if [[ -f ${CONFIG_MAP["CACHE_DIR"]}/es-decl.map ]]; then
    ConsoleInfo "-->" "declaring exit-status from cache"
    source ${CONFIG_MAP["CACHE_DIR"]}/es-decl.map
else
    DeclareExitStatus
    if ConsoleHasErrors; then printf "\n"; exit 29; fi
fi
```

2.12. Dependency Declarations

```
if [[ -f ${CONFIG_MAP["CACHE_DIR"]}/dep-decl.map ]]; then
    ConsoleInfo "-->" "declaring dependencies from cache"
    source ${CONFIG_MAP["CACHE_DIR"]}/dep-decl.map
else
    ConsoleInfo "-->" "declaring dependencies from source"
    DeclareDependencies
    if ConsoleHasErrors; then printf "\n"; exit 30; fi
fi
```

2.13. Task Declarations

```
if [[ -f ${CONFIG_MAP["CACHE_DIR"]}/task-decl.map ]]; then
    ConsoleInfo "-->" "declaring tasks from cache"
    source ${CONFIG_MAP["CACHE_DIR"]}/task-decl.map
else
    ConsoleInfo "-->" "declaring tasks from source"
    DeclareTasks
    if ConsoleHasErrors; then printf "\n"; exit 31; fi
fi
source $FW_HOME/bin/loader/init/process-tasks.sh
ProcessTasks
if ConsoleHasErrors; then printf "\n"; exit 32; fi
```

2.14. Scenario Declarations

```
ConsoleInfo "-->" "declaring scenarios from source"
DeclareScenarios
if ConsoleHasErrors; then printf "\n"; exit 33; fi
source $FW_HOME/bin/loader/init/process-scenarios.sh
ProcessScenarios
if ConsoleHasErrors; then printf "\n"; exit 34; fi
```

2.15. Set Levels

```
case "${CONFIG_MAP["LOADER_LEVEL"]}" in
    off | all | fatal | error | warn-strict | warn | info | debug | trace)
        ;;
    *)
        ConsoleError "-->" "unknown loader-level: ${CONFIG_MAP["LOADER_LEVEL"]}"
        printf "    use: off, all, fatal, error, warn-strict, warn, info, debug,
trace\n\n"
        exit 35
        ;;
esac
case "${CONFIG_MAP["SHELL_LEVEL"]}" in
    off | all | fatal | error | warn-strict | warn | info | debug | trace)
        ;;
    *)
        ConsoleError "-->" "unknown shell-level: ${CONFIG_MAP["SHELL_LEVEL"]}"
        printf "    use: off, all, fatal, error, warn-strict, warn, info, debug,
trace\n\n"
        exit 36
        ;;
esac
case "${CONFIG_MAP["TASK_LEVEL"]}" in
    off | all | fatal | error | warn-strict | warn | info | debug | trace)
        ;;
    *)
        ConsoleError "-->" "unknown task-level: ${CONFIG_MAP["TASK_LEVEL"]}"
        printf "    use: off, all, fatal, error, warn-strict, warn, info, debug,
trace\n\n"
        exit 37
        ;;
esac
```

2.16. Do (Exit) Options

```
source $FW_HOME/bin/loader/init/do-options.sh
DoOptions
if ConsoleHasErrors; then printf "\n"; exit 38; fi

if [[ $DO_EXIT == true ]]; then
    _te=$(date +%s.%N)
    _exec_time=$_te-$_ts
    ConsoleInfo "-->" "execution time: $(echo $_exec_time | bc -l) sec"
    ConsoleInfo "-->" "done"
    exit 0
fi
```

2.17. Create Runtime Configuration File

```
CONFIG_MAP["FW_L1_CONFIG"]=$(mktemp "$TMP_DIRECTORY/$(date +"%H-%M-%S")-
${CONFIG_MAP["APP_MODE"]}-XXX")
export FW_L1_CONFIG=${CONFIG_MAP["FW_L1_CONFIG"]}
WriteRuntimeConfig
```

2.18. Execute Task or Scenario

```
__errno=0
if [[ "${OPT_CLI_MAP["execute-task"]}" != false ]]; then
    echo ${OPT_CLI_MAP["execute-task"]} | $FW_HOME/bin/shell/shell.sh
    __et=$?
    __errno=$((__errno + __et))
fi
if [[ "${OPT_CLI_MAP["run-scenario"]}" != false ]]; then
    echo "execute-scenario ${OPT_CLI_MAP["run-scenario"]}" |
$FW_HOME/bin/shell/shell.sh
    __et=$?
    __errno=$((__errno + __et))
    DO_EXIT_2=true
fi
```

2.19. Start Shell

```
if [[ ${DO_EXIT_2} == false ]]; then
    $FW_HOME/bin/shell/shell.sh
    __errno=$?
fi
```

2.20. Clean Up

```
if [[ -f $FW_L1_CONFIG ]]; then
    rm $FW_L1_CONFIG >& /dev/null
fi
if [[ -d $TMP_DIRECTORY && $(ls $TMP_DIRECTORY | wc -l) == 0 ]]; then
    rmdir $TMP_DIRECTORY
fi
```

2.21. Done

```
ConsoleMessage "\n\nhave a nice day\n\n\n"
exit $__errno
```

3. The Shell