git-jcheck

Description

`git-jcheck` is a Git port of the Mercurial extension jcheck. `git-jcheck` validates that one or more Git commits follow an OpenJDK project's conventions. Example of such a conventions include:

- Tab characters should not be present in the source code
- The commit message should follow the prescribed format
- Executable files and symbolic links should not be present

Different OpenJDK project have different conventions. The conventions are described by the file `.jcheck/conf` in the project's source code repository. As an example, the OpenJDK project Skara has the following in its `.jcheck/conf` file:

```
[checks]
error=author,reviewers,whitespace

[checks "whitespace"]
files=.*\.java$|.*\.yml$|.*\.gradle$|.*.\txt$

[checks "reviewers"]
reviewers=1
```

The above configuration shows that for OpenJDK project Skara `git-jcheck` will verify that commits have an author with full name and e-mail, is reviewed by at least one Reviewer from project Skara and that files with the suffixes `.java`, `.yml`, `.gradle` and `.txt` do not contain tabs, carriage returns or trailing whitespace.

Since the `.jcheck/conf` file is versioned every commit is checked according to the conventions described in the `.jcheck/conf` file *in that commit.* This means checks can be added and/or removed and old commits will still be checked correctly. If no `.jcheck/file` is present in a commit then `git-jcheck` exits with status code 0.

Usage

```
$ git jcheck -h
usage: git jcheck [options]
        -r, --rev REV               Check the specified revision or range (default: HEAD)
            --census FILE           Use the specified census (default: https://openjdk.java.net/census.xml)
            --ignore CHECKS         Ignore errors from checks with the given name
            --setup-pre-push-hook   Set up a pre-push hook that runs jcheck on commits to be pushed
        -m, --mercurial             Deprecated: force use of mercurial
            --verbose               Turn on verbose output
            --debug                 Turn on debugging output
            --lax                   Check comments, tags and whitespace laxly
        -s, --strict                Check everything
        -v, --version               Print the version of this tool
        -h, --help                  Show this help text
                        --conf-staged                       Use staged .jcheck/conf
                        --conf-working-tree               Use .jcheck/conf in current working tree
                        --staged                               Check staged changes as if they were committed
                        --working-tree                       Check changes in working tree as if they were
committed
```

Examples

Run `git-jcheck` on the `HEAD` commit:

```
$ git jcheck
```

Run `git-jcheck` on all commits between master and `HEAD`:

```
$ git jcheck -r master..HEAD
```

Run `git-jcheck` on all commits between master and `HEAD` and ignore errors about branches:

```
$ git jcheck -r master..HEAD --ignore=branches
```

Note: `git-jcheck` always prints the name of the check when showing errors. Use these check names as arguments to `--ignore`.

Download the OpenJDK census database locally and tell `git-jcheck` to use it when checking commits between `master` and `HEAD`:

```
$ curl -O https://openjdk.java.net/census.xml
$ git jcheck -r master..HEAD --census=$PWD/census.xml
```

Note: when explicitly specifying a census file then `git-jcheck` will work offline.

Setup a pre-push hook that will run git-jcheck on commits that are about to be pushed:

```
$ git jcheck --setup-pre-push-hook
```

Note: if `git-jcheck` discovers any errors then the push will be aborted.

Configuration

All options to git-jcheck can be configured via Git's configuration files. For example, the following will always use a local version of census stored in `~/.jcheck/census.xml` when running `git-jcheck`:

```
$ git config --global jcheck.census "$HOME/.jcheck/census.xml"
```

If you want to always ignore the "branches" check then just configure "jcheck.ignore" as in the following example:

```
$ git config --global jcheck.ignore branches
```

Source

See GitJCheck.java.