Table of Contents

The OpenRIF Guide

This is the OpenRIF guide. Based on GitBook: https://github.com/GitbookIO/gitbook

Current site

Current PDF version

Introduction

The introduction.

Introduction Subsection

The introduction subsection. The words OWL, Protege, etc. for search test. TODO: clean this up.

Contributing to OpenRIF

Contributing to OpenRIF.

Contributing to ontologies

Contributing to the ontologies.

Contributing to the OpenRIF Guide

Contributing should be easy. If it is not, please open a bug report describing what can be fixed or improved. There are few different ways for editing the guide.

Introduction

The OpenRIF Guide is currently generated with the GitBook tool. The current format of the source files is the AsciiDoc/Asciidoctor format. This format is similar to the MarkDown idea but it has many additional features. This guide will soon include a separate page that has a brief summary of the main Asciidoctor features to use in contributing to this guide. The above Asciidoctor link will take you to their documentation page that contains many short and detailed guides for writing Asciidoc documents, and using related tools.

The source files are then built with GitBook. The build is done on the master branch and the results are in the `_book` directory. As changes are made in the source files, the `_book` folder will be updated with new builds. This folder can be treated as a draft of the next publication of the guide. If you have a local checkout of the master branch, you can open the `_book/index.html` file in a browser to see the latest build as it would appear at http://www.openrif.org/guide/ when it is published.

You can publish the guide if you have the needed tools installed by running the `./publish` script that is available in the master branch. This script will:

1. Commit and push any changes on the master branch. The push might fail if you don't have the latest changes. Please do a pull and resolve any conflicts before trying this.
2. Build, commit, and push the master branch. This will update the `_book` folder to match the latest source files.
3. checkout the gh-pages branch
4. Copy the `_book` folder from master to the root of the `gh-pages` branch, commit, and push it. This will make the latest build available at http://www.openrif.org/guide/
5. checkout master

There should be enough console output to see the above process and to check if it went well. Nothing will be lost or damaged if there are any errors.

Editing directly on GitHub

You can edit existing source files in GitHub and preview your changes before committing them. Go to the master source folder and find the file you would like to edit and click on it. Then, click on the pencil icon to edit the file. Edit as needed and preview the changes in the preview tab. When you would like to save the changes, enter a description of your changes in the 1Commit changes` section at the bottom and commit. This will make your changes permanent in the repository, and available for the next build of the guide.

Editing in a local checkout

The editing process is the same as above. The only difference is that you can edit many files and commit all of them together as a single change. This might work better when your changes involve multiple files.

When you are happy with all your changes and commits, you can push them to the `openrif/guide` repository if you have write permissions. Otherwise, please create a pull request against it.

While you are editing locally, you can still see a preview of your changes by using one of the options described here. The Firefox plugin has been working well for one editor so far, and in combination with a text editor that has an auto-saving feature (when a typing pause is detected) previewing changes becomes very easy.

Tools

The tools.

The Protege Editor

Download `Protege Desktop` (not WebProtege) from: http://protege.stanford.edu/products.php

The larger download button will download the latest Protege version 5. Older versions are available from the link under the main download button.

Downloading the `platform independent version` might be a better option but it depends on your use-case. The independent version is a zip file that you can unpack anywhere and run the program from that folder. This avoids any installation steps, allows you to easily have multiple versions of Protege without the installations affecting each other, etc.

The latest Protege 5 beta 17 can be downloaded directly from: https://github.com/protegeproject/protege/releases

Older versions are here: http://protegewiki.stanford.edu/wiki/Protege_Desktop_Old_Versions