GitHub
GIT CHEAT SHEET

V 1.1.1

Git is the open source distributed version control system that facilitates GitHub activities on your laptop or desktop. This cheat sheet summarizes commonly used Git command line instructions for quick reference.

INSTALL GIT

GitHub provides desktop clients that include a graphical user interface for the most common repository actions and an automatically updating command line edition of Git for advanced scenarios.

GitHub for Windows
https://windows.github.com

GitHub for Mac
https://mac.github.com

Git distributions for Linux and POSIX systems are available on the official Git SCM web site.

Git for All Platforms
http://git-scm.com

CONFIGURE TOOLING

Configure user information for all local repositories

`$ git config --global user.name "[name]"`
Sets the name you want attached to your commit transactions
`$ git config --global user.email "[email address]"`
Sets the email you want attached to your commit transactions
`$ git config --global color.ui auto`
Enables helpful colorization of command line output

CREATE REPOSITORIES

Start a new repository or obtain one from an existing URL

`$ git init [project-name]`
Creates a new local repository with the specified name
`$ git clone [url]`
Downloads a project and its entire version history

MAKE CHANGES

Review edits and craft a commit transaction

`$ git status`
Lists all new or modified files to be committed
`$ git diff`
Shows file differences not yet staged
`$ git add [file]`
Snapshots the file in preparation for versioning
`$ git diff --staged`
Shows file differences between staging and the last file version
`$ git reset [file]`
Unstages the file, but preserve its contents
`$ git commit -m "[descriptive message]"`
Records file snapshots permanently in version history

GROUP CHANGES

Name a series of commits and combine completed efforts

`$ git branch`
Lists all local branches in the current repository
`$ git branch [branch-name]`
Creates a new branch
`$ git checkout [branch-name]`
Switches to the specified branch and updates the working directory
`$ git merge [branch]`
Combines the specified branch's history into the current branch
`$ git branch -d [branch-name]`
Deletes the specified branch

GIT CHEAT SHEET

REFACTOR FILENAMES
Relocate and remove versioned files

```
$ git rm [file]
```
Deletes the file from the working directory and stages the deletion

```
$ git rm --cached [file]
```
Removes the file from version control but preserves the file locally

```
$ git mv [file-original] [file-renamed]
```
Changes the file name and prepares it for commit

SUPPRESS TRACKING
Exclude temporary files and paths

```
*.log
build/
temp-*
```
A text file named `.gitignore` suppresses accidental versioning of files and paths matching the specified patterns

```
$ git ls-files --other --ignored --exclude-standard
```
Lists all ignored files in this project

SAVE FRAGMENTS
Shelve and restore incomplete changes

```
$ git stash
```
Temporarily stores all modified tracked files

```
$ git stash pop
```
Restores the most recently stashed files

```
$ git stash list
```
Lists all stashed changesets

```
$ git stash drop
```
Discards the most recently stashed changeset

REVIEW HISTORY
Browse and inspect the evolution of project files

```
$ git log
```
Lists version history for the current branch

```
$ git log --follow [file]
```
Lists version history for a file, including renames

```
$ git diff [first-branch]...[second-branch]
```
Shows content differences between two branches

```
$ git show [commit]
```
Outputs metadata and content changes of the specified commit

REDO COMMITS
Erase mistakes and craft replacement history

```
$ git reset [commit]
```
Undoes all commits after `[commit]`, preserving changes locally

```
$ git reset --hard [commit]
```
Discards all history and changes back to the specified commit

SYNCHRONIZE CHANGES
Register a repository bookmark and exchange version history

```
$ git fetch [bookmark]
```
Downloads all history from the repository bookmark

```
$ git merge [bookmark]/[branch]
```
Combines bookmark's branch into current local branch

```
$ git push [alias] [branch]
```
Uploads all local branch commits to GitHub

```
$ git pull
```
Downloads bookmark history and incorporates changes

PDFBox - PDF Text Extraction

Java PDF Library, pdftotext, PDF to text, java pdf text extraction

Table of contents

1. Extracting Text

See class:org.pdfbox.util.PDFTextStripper
See class:org.pdfbox.searchengine.lucene.LucenePDFDocument
See command line app:ExtractText

One of the main features of PDFBox is its ability to quickly and accurately extract text from a variety of PDF documents. This functionality is encapsulated in the org.pdfbox.util.PDFTextStripper and can be easily executed on the command line with org.pdfbox.ExtractText.

1.1. Lucene Integration

Lucene is an open source text search library from the Apache Jakarta Project. In order for Lucene to be able to index a PDF document it must first be converted to text. PDFBox provides a simple approach for adding PDF documents into a Lucene index.

```
Document luceneDocument = LucenePDFDocument.getDocument( ... );
```

Now that you hava a Lucene Document object, you can add it to the Lucene index just like you would if it had been created from a text or HTML file. The LucenePDFDocument automatically extracts a variety of metadata fields from the PDF to be added to the index, the javadoc shows details on those fields. This approach is very simple and should be sufficient for most users, if not then you can use some of the advanced text extraction techniques described in the next section.

1.2. Advanced Text Extraction

Some applications will have complex text extraction requiments and neither the command line application nor the LucenePDFDocument will be able to fulfill those requirements. It is possible for users to utilize or extend the PDFTextStripper class to meet some of these requirements.

1.2.1. Limiting The Extracted Text

There are several ways that we can limit the text that is extracted during the extraction process. The simplest is to specify the range of pages that you want to be extracted. For example, to only extract text from the second and third pages of the PDF document you could do this:

```
PDFTextStripper stripper = new PDFTextStripper();
```

```
            stripper.setStartPage( 2 );
            stripper.setEndPage( 3 );
            stripper.writeText( ... );
```

Note:
The startPage and endPage properties of PDFTextStripper are 1 based and inclusive.

If you wanted to start on page 2 and extract to the end of the document then you would just set the startPage property. By default all pages in the pdf document are extracted.

It is also possible to limit the extracted text to be between two bookmarks in the page. If you are not familiar with how to use bookmarks in PDFBox then you should review the Bookmarks page. Similar to the startPage/endPage properties, PDFTextStripper also has startBookmark/endBookmark properties. There are some caveats to be aware of when using this feature of the PDFTextStripper. Not all bookmarks point to a page in the current PDF document. The possible states of a bookmark are:

- null - The property was not set, this is the default.
- Points to page in the PDF - The property was set and points to a valid page in the PDF
- Bookmark does not point to anything - The property was set but the bookmark does not point to any page
- Bookmark points to external action - The property was set, but it points to a page in a different PDF or performs an action when activated

The table below will describe how PDFBox behaves in the various scenarios:

Start Bookmark	End Bookmark	Result
null	null	This is the default, the properties have no effect on the text extraction.
Points page in the PDF	null	Text extraction will begin on the page that this bookmark points to and go until the end of the document.
null	Points page in the PDF	Text extraction will begin on the first page and stop at the end of the page that this bookmark points to.
Bookmark does not point to anything	null	Because the PDFTextStripper cannot determine a start page based on the bookmark, it will start on the first page and go

		until the end of the document.
null	Bookmark does not point to anything	Because the PDFTextStripper cannot determine a end page based on the bookmark, it will start on the first page and go until the end of the document.
Bookmark does not point to anything	Bookmark does not point to anything	This is a special case! If the startBookmark and endBookmark are exactly the same then no text will be extracted. If they are different then it is not possible for the PDFTextStripper to determine that pages so it will include the entire document.
Bookmark points to external action	Bookmark points to external action	If either the startBookmark or the endBookmark refer to an external page or execute an action then an OutlineNotLocalException will be thrown to indicate to the user that the bookmark is not valid.

Note:

PDFTextStripper will check both the startPage/endPage and the startBookmark/endBookmark to determine if text should be extracted from the current page.