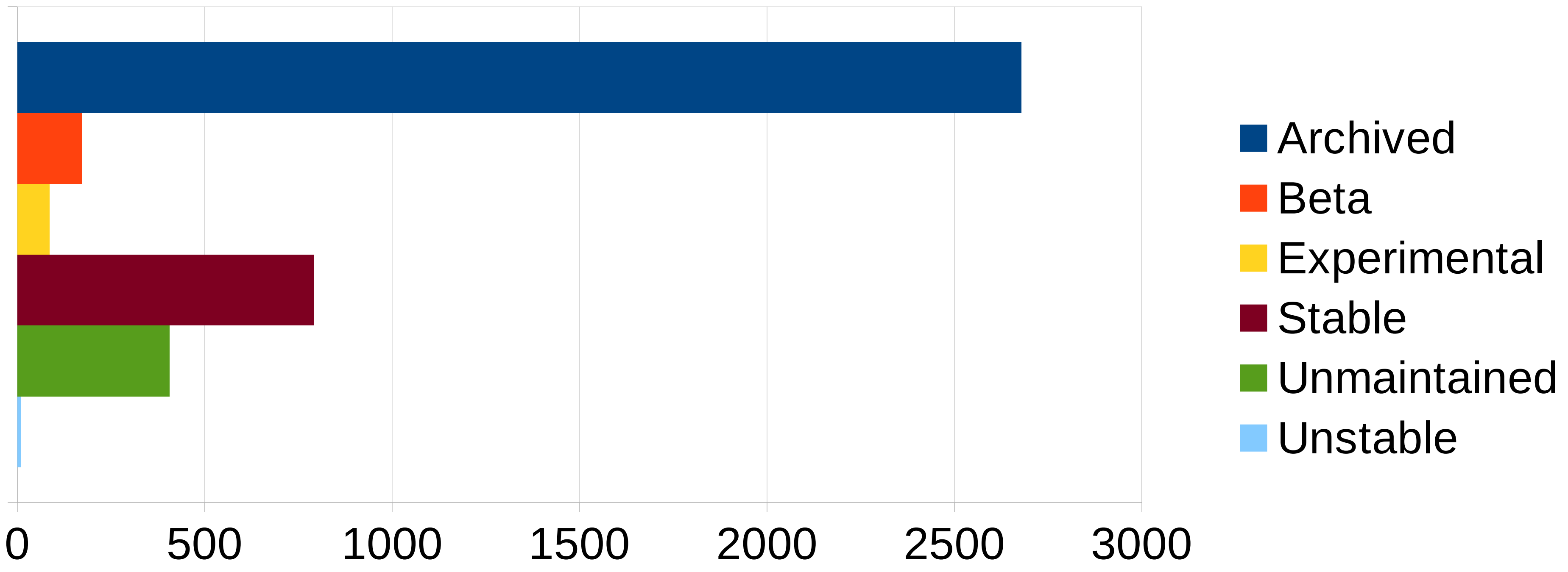
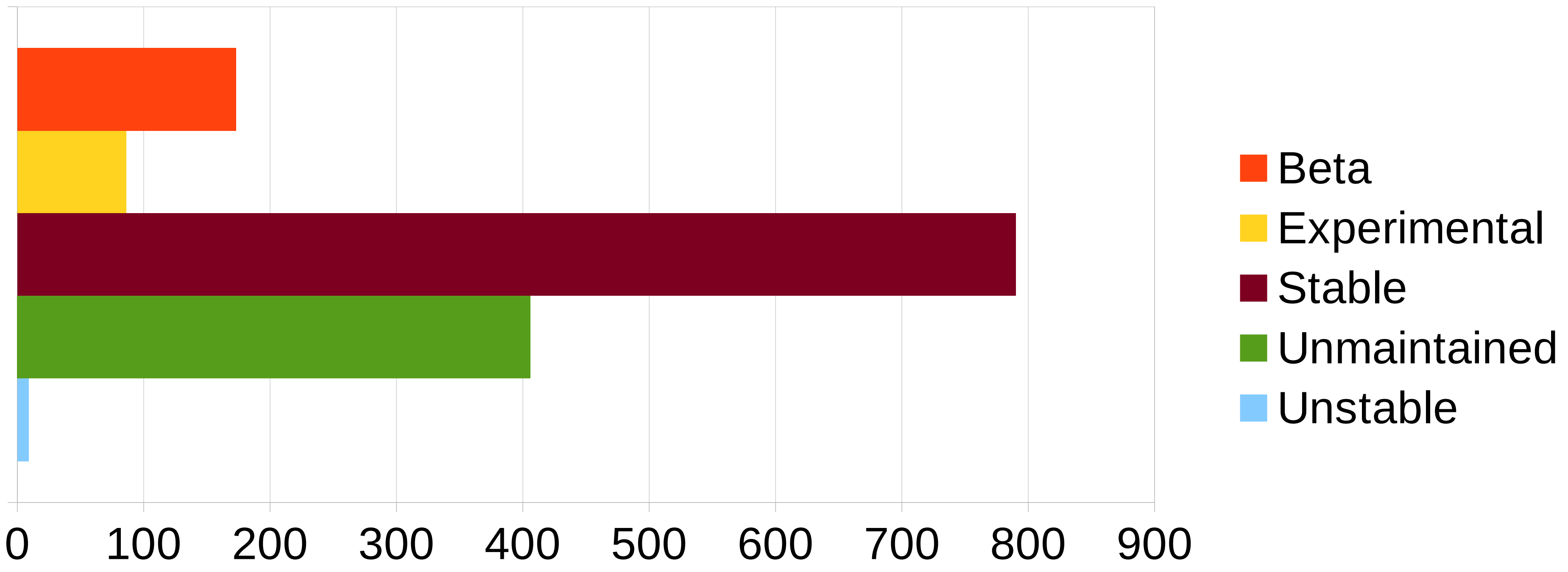
Creating a directory of extension and skin usability

Yaron Koren
MediaWiki Users and Developers Conference
Portland, Oregon, USA
April 17, 2024

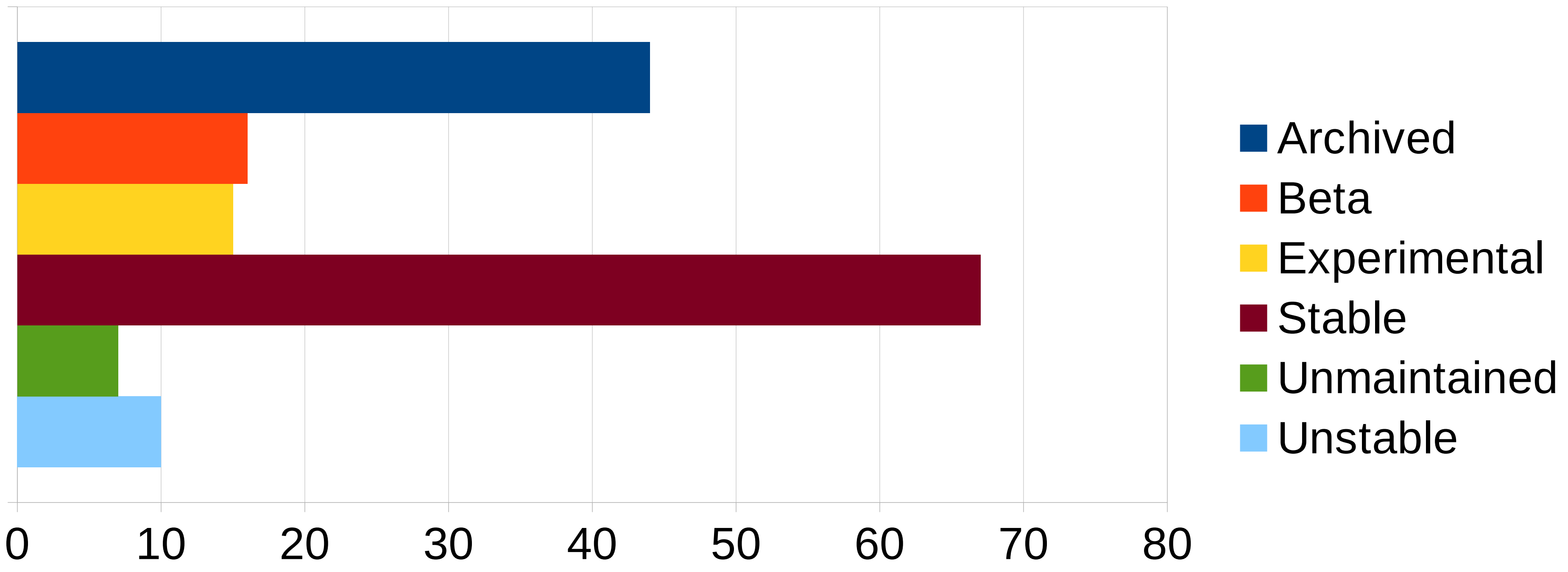
Extensions on mediawiki.org, by status



Non-archived extensions on mediawiki.org, by status

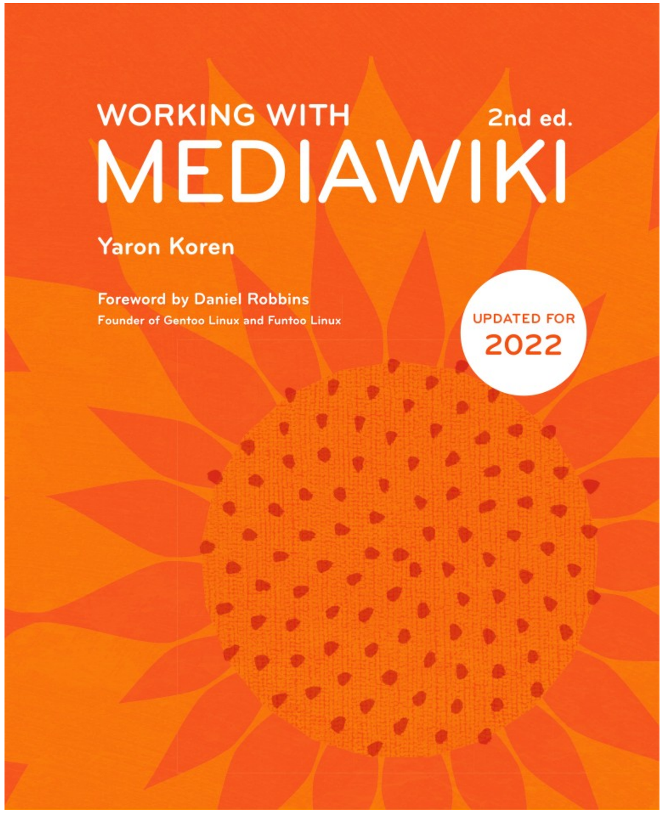


Skins on mediawiki.org, by status



How can MediaWiki administrators know which of these (hundreds of) extensions and (dozens of) skins to install?

One option


WORKING WITH
2nd ed.
MEDIAWIKI
Yaron Koren
Foreword by Daniel Robbins
Founder of Gentoo Linux and Funtoo Linux
UPDATED FOR 2022

Needed: a dedicated resource that will help people make this decision.

"Complete" implementation

“I have MediaWiki version ___ installed, on operating system ____ version _, with web server ____ version _, database system ___ version _, and PHP version _. I would like the wiki to do _______. What is the set of extensions and skins I can install, and the commit ID and required libraries for each, that I can use for this purpose?”

Minimum implementation

"For the **current LTS** (long-term support) version of MediaWiki, running on the **recommended PHP version** and reasonable versions of **MySQL**, **Apache** and **Linux** (**Debian** or **Ubuntu**), what are the set of recommended skins and extensions, and the recommended commit ID and required libraries for each?"

The "OnWikimedia" template

Denotes which extensions and skins are used on one or more Wikimedia sites.

The "Used by" template

In use on mediawiki.org since 2022. Currently used for information on the following packages:

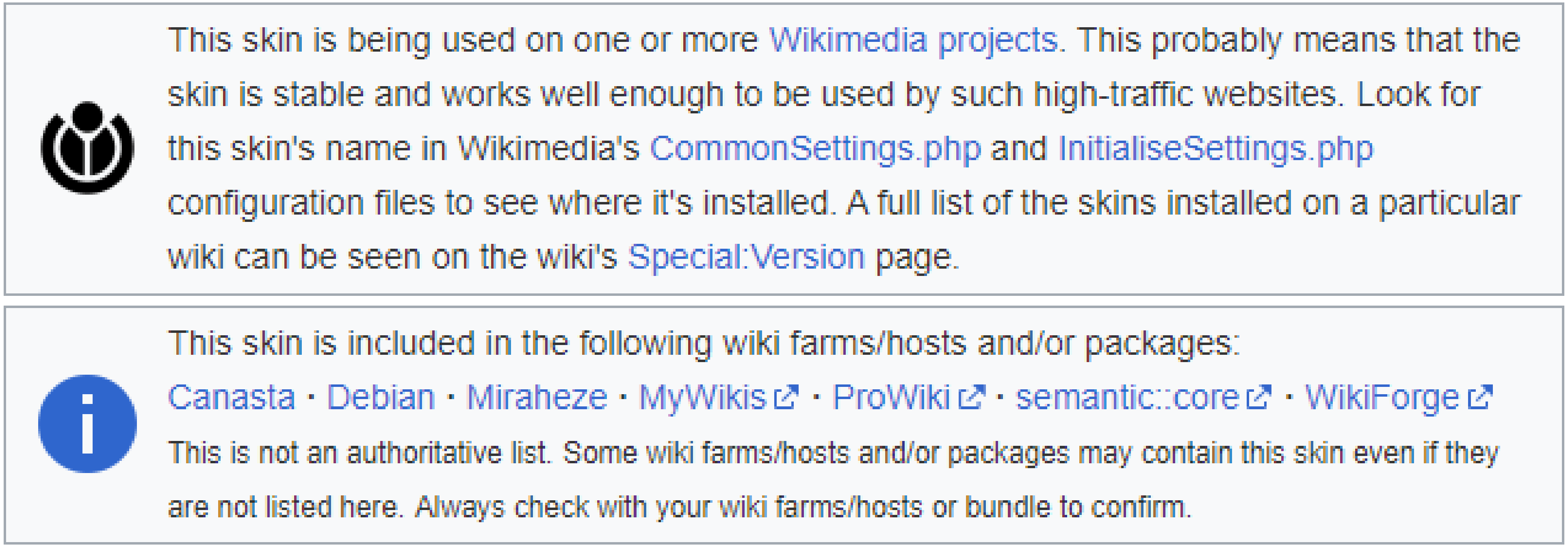
- BlueSpice
- Canasta
- Debian
- Open CSP
- semantic::core

The "Used by" template

...and the following wiki farms:

- Fandom
- Miraheze
- MyWikis
- ProWiki
- ShoutWiki
- Telepedia
- wiki.gg
- WikiForge

From bottom of "Timeless" skin page

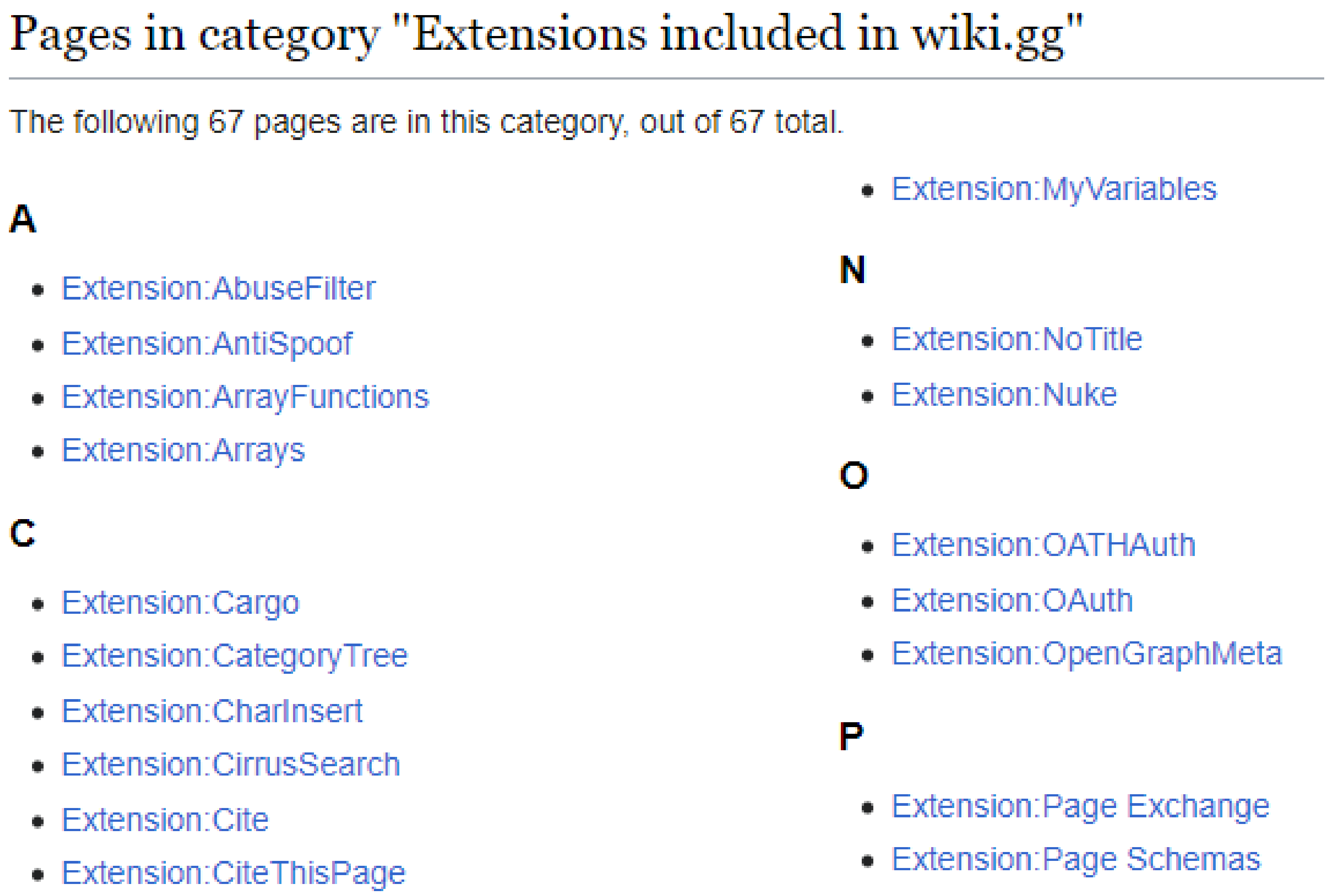


This skin is being used on one or more Wikimedia projects. This probably means that the skin is stable and works well enough to be used by such high-traffic websites. Look for this skin's name in Wikimedia's CommonSettings.php and InitialiseSettings.php configuration files to see where it's installed. A full list of the skins installed on a particular wiki can be seen on the wiki's Special:Version page.

This skin is included in the following wiki farms/hosts and/or packages:

Canasta · Debian · Miraheze · MyWikis · ProWiki · semantic::core · WikiForge

This is not an authoritative list. Some wiki farms/hosts and/or packages may contain this skin even if they are not listed here. Always check with your wiki farms/hosts or bundle to confirm.



Pages in category "Extensions included in wiki.gg"

The following 67 pages are in this category, out of 67 total.

A

- Extension:AbuseFilter
- Extension:AntiSpoof
- Extension:ArrayFunctions
- Extension:Arrays

C

- Extension:Cargo
- Extension:CategoryTree
- Extension:CharInsert
- Extension:CirrusSearch
- Extension:Cite
- Extension:CiteThisPage

- Extension:MyVariables

N

- Extension:NoTitle
- Extension:Nuke

O

- Extension:OATHAuth
- Extension:OAuth
- Extension:OpenGraphMeta

P

- Extension:Page Exchange
- Extension:Page Schemas

Example: CodeMirror extension

- Written by Pavel Astakhov
- Allows for IDE-style editing of wiki pages.
- Used by 8 of the 13 packages and wiki farms, plus Wikimedia.
- In my opinion, the strongest current candidate for bundling within MediaWiki!

Example 2: CookieWarning extension

- Written by Florian Schmidt et al.
- Displays the now-ubiquitous “This site uses cookies” popup warning
- Used by 8 of the 13 packages and wiki farms
- Another strong bundling (or maybe even integration) contender.

Other unexpectedly popular extensions, discovered via "Used by":

- Arrays (8 of 13) 🙁
- Loops (8 of 13)
- Maps (8 of 13)
- Page Forms (8 of 13)
- Semantic MediaWiki (9 of 13)
- Variables (10 of 13) 🙁

Example 3: DynamicPageList extensions

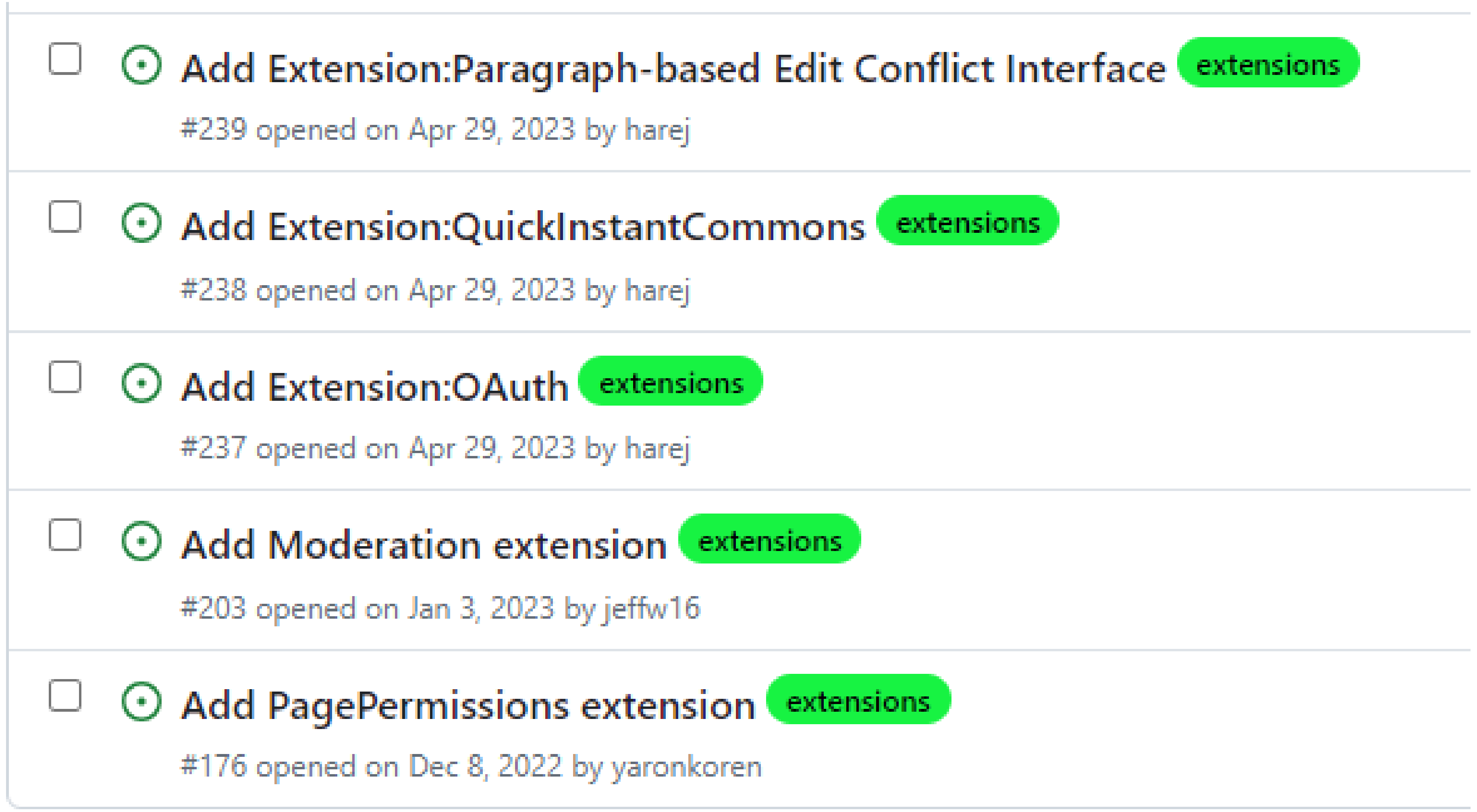
- DynamicPageList (Wikimedia): **2** “Used by”, + Wikimedia
- DynamicPageList (third-party) (*AKA DPL2*): **0** “Used by”
- DynamicPageList3: **6** “Used by”

Should there be a public listing of extensions and their popularity?

- e.g., "**Category:Extensions used in 6 of the cataloged MediaWiki packages and wiki farms**".
- Pros: Lets people see popularity at a glance
- Cons: The exact popularity is kind of random; could lead to "groupthink", i.e. extensions gaining popularity as a result of current popularity

Special case: Canasta

- In my (biased) opinion, the best MediaWiki package to use
- Complete, Docker-based solution
- Fully open source
- Consensus-based methodology for determing extension and skin inclusion



Add Extension:Paragraph-based Edit Conflict Interface extensions

#239 opened on Apr 29, 2023 by harej

Add Extension:QuickInstantCommons extensions

#238 opened on Apr 29, 2023 by harej

Add Extension:OAuth extensions

#237 opened on Apr 29, 2023 by harej

Add Moderation extension extensions

#203 opened on Jan 3, 2023 by jeffw16

Add PagePermissions extension extensions

#176 opened on Dec 8, 2022 by yaronkoren

How to choose the best version (down to the commit ID) of an extension, or skin?

The Canasta approach

- For extensions bundled in with MediaWiki, just go with that exact code
- For WMF-maintained extensions, go with the branch for the current MW version (e.g., REL1_39), and the latest commit in that branch
- For other extensions that have version releases, go with the latest released version
- For all other extensions, go with just the latest commit

Okay, but what if this version of the extension contains bugs?

A word about bugs

- The sad truth of software: every piece of software that does something useful contains bugs
- Bugs in a MW extension can be inherent to the code, or appear in interaction with all the other "moving parts" (MediaWiki, LAMP stack, Docker, libraries, other extensions and skins)

So, what to do about bugs?

- Bugs can range in severity from "Add deprecation warnings in logs" to "Delete the entire database"
- Major bugs of course must be dealt with
- In the ideal case, a major bug has already been fixed in the extension - and upgrading to that later code solves the problem

If the bugs are still there…

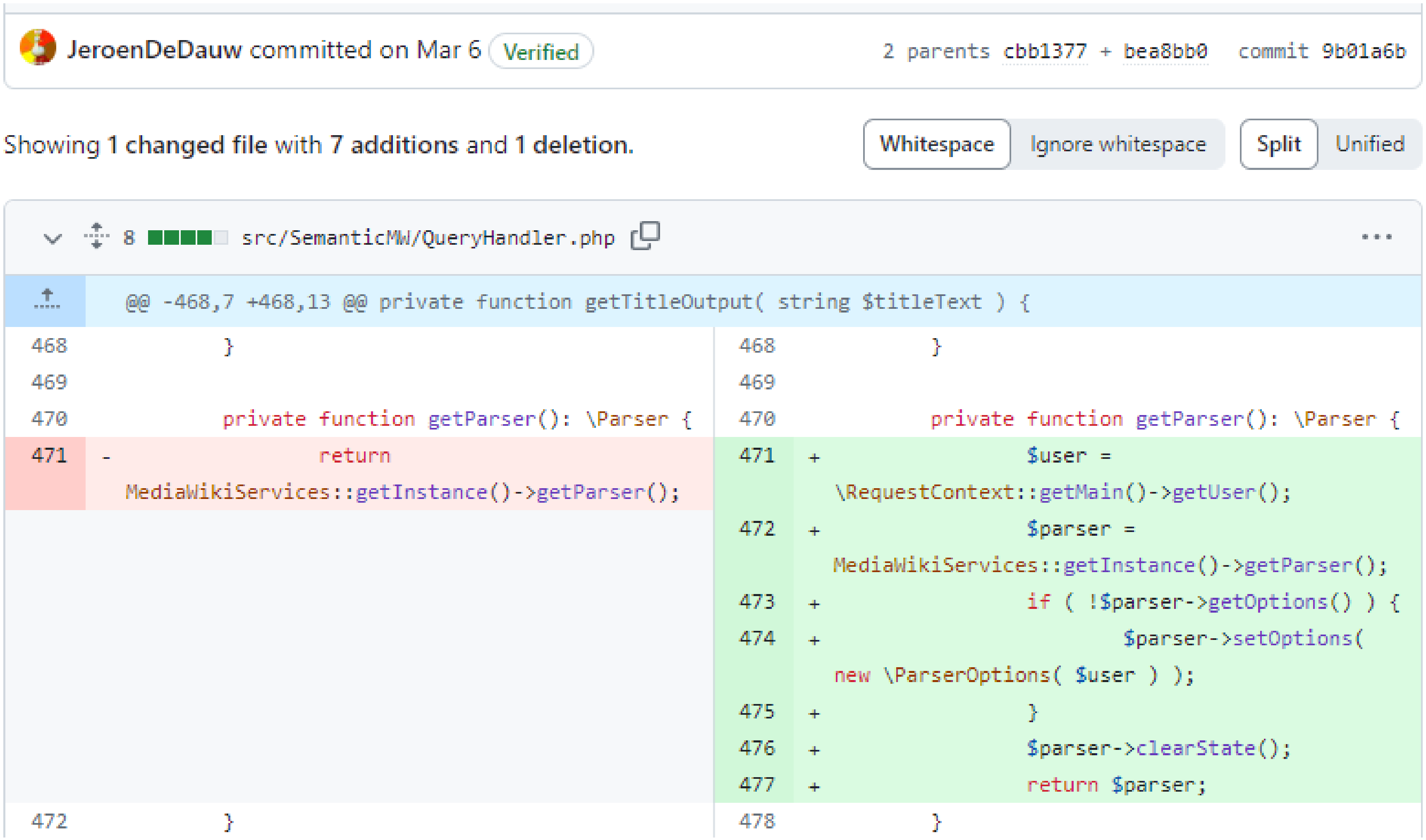
…and they’re severe enough, there are three main options:

- Get the bug fixed in the extension code
- Apply a local patch
- Get rid of that extension

Case study: Maps not working with VisualEditor

Bug discovered by Pavel Astakhov, while running a Canasta-based Docker image

Submitted to the Maps extension GitHub repository, and committed by Jeroen De Dauw soon afterwards



JeroenDeDauw committed on Mar 6 Verified   2 parents cbb1377 + bea8bb0   commit 9b01a6b

Showing 1 changed file with 7 additions and 1 deletion.

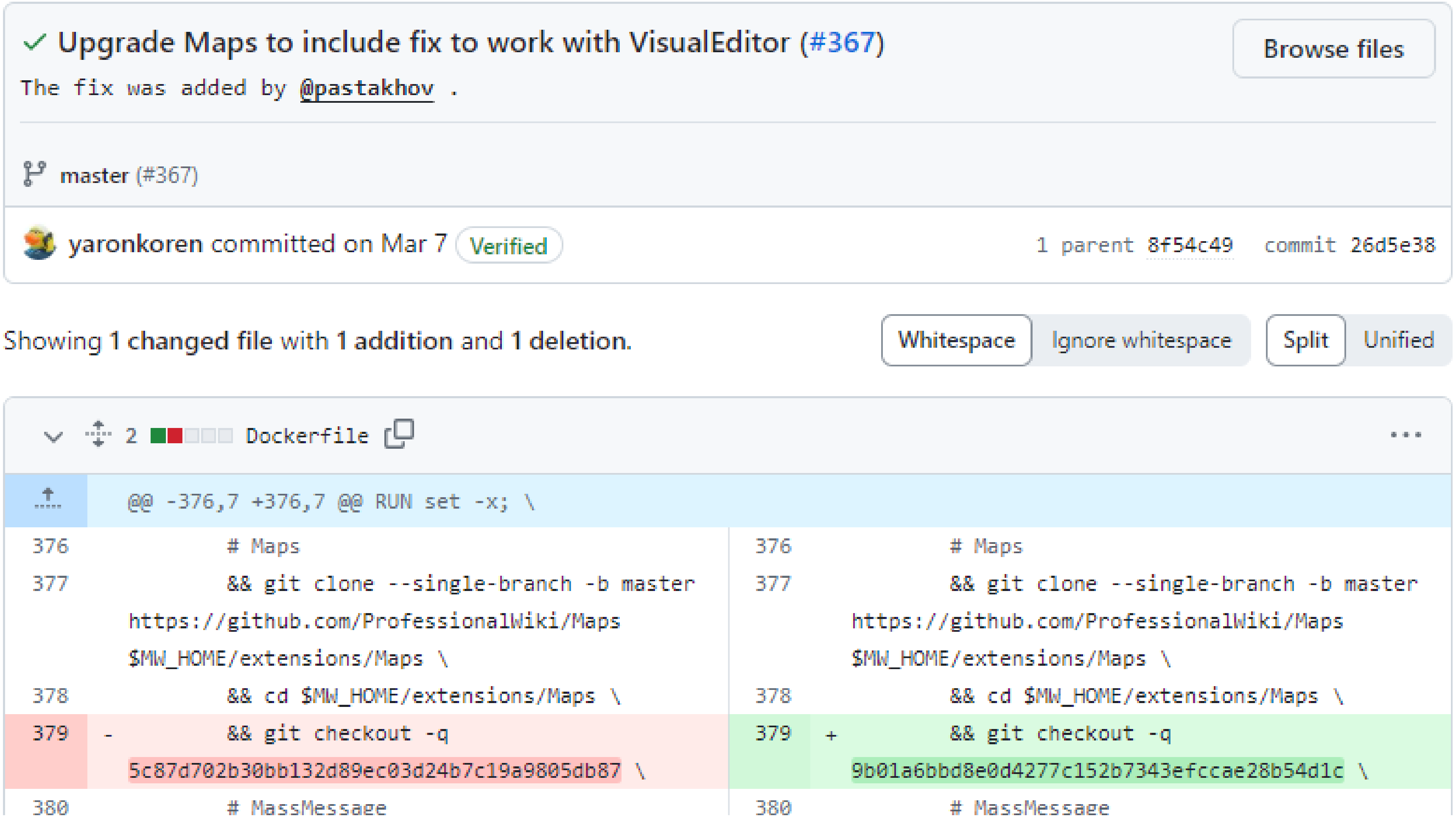
Whitespace | Ignore whitespace | Split | Unified

8 src/SemanticMW/QueryHandler.php

```
@@ -468,7 +468,13 @@ private function getTitleOutput( string $titleText ) {
 	}

 	private function getParser(): \Parser {
-		return MediaWikiServices::getInstance()->getParser();
+		$user = \RequestContext::getMain()->getUser();
+		$parser = MediaWikiServices::getInstance()->getParser();
+		if ( !$parser->getOptions() ) {
+			$parser->setOptions( new \ParserOptions( $user ) );
+		}
+		$parser->clearState();
+		return $parser;
 	}
```

Version of Maps was then updated in Canasta.


✓ Upgrade Maps to include fix to work with VisualEditor (#367)

The fix was added by @pastakhov .

master (#367)

yaronkoren committed on Mar 7 Verified 1 parent 8f54c49 commit 26d5e38

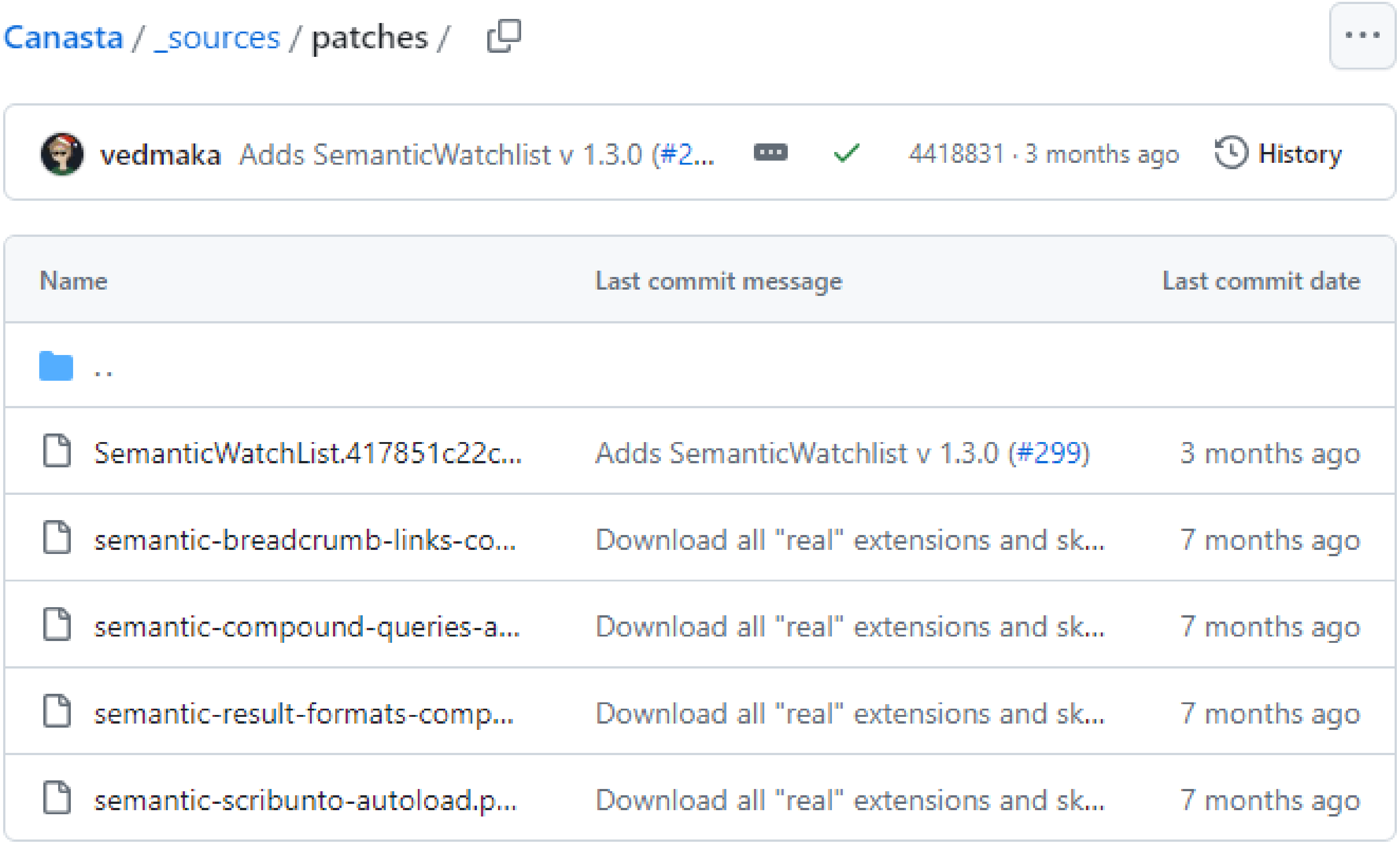
Showing 1 changed file with 1 addition and 1 deletion.

2 Dockerfile

```
@@ -376,7 +376,7 @@ RUN set -x; \
376          # Maps                                                376          # Maps
377          && git clone --single-branch -b master                377          && git clone --single-branch -b master
     https://github.com/ProfessionalWiki/Maps                           https://github.com/ProfessionalWiki/Maps
     $MW_HOME/extensions/Maps \                                         $MW_HOME/extensions/Maps \
378          && cd $MW_HOME/extensions/Maps \                      378          && cd $MW_HOME/extensions/Maps \
379 -        && git checkout -q                                    379 +        && git checkout -q
     5c87d702b30bb132d89ec03d24b7c19a9805db87 \                         9b01a6bbd8e0d4277c152b7343efccae28b54d1c \
380          # MassMessage                                          380          # MassMessage
```

Current patches in Canasta

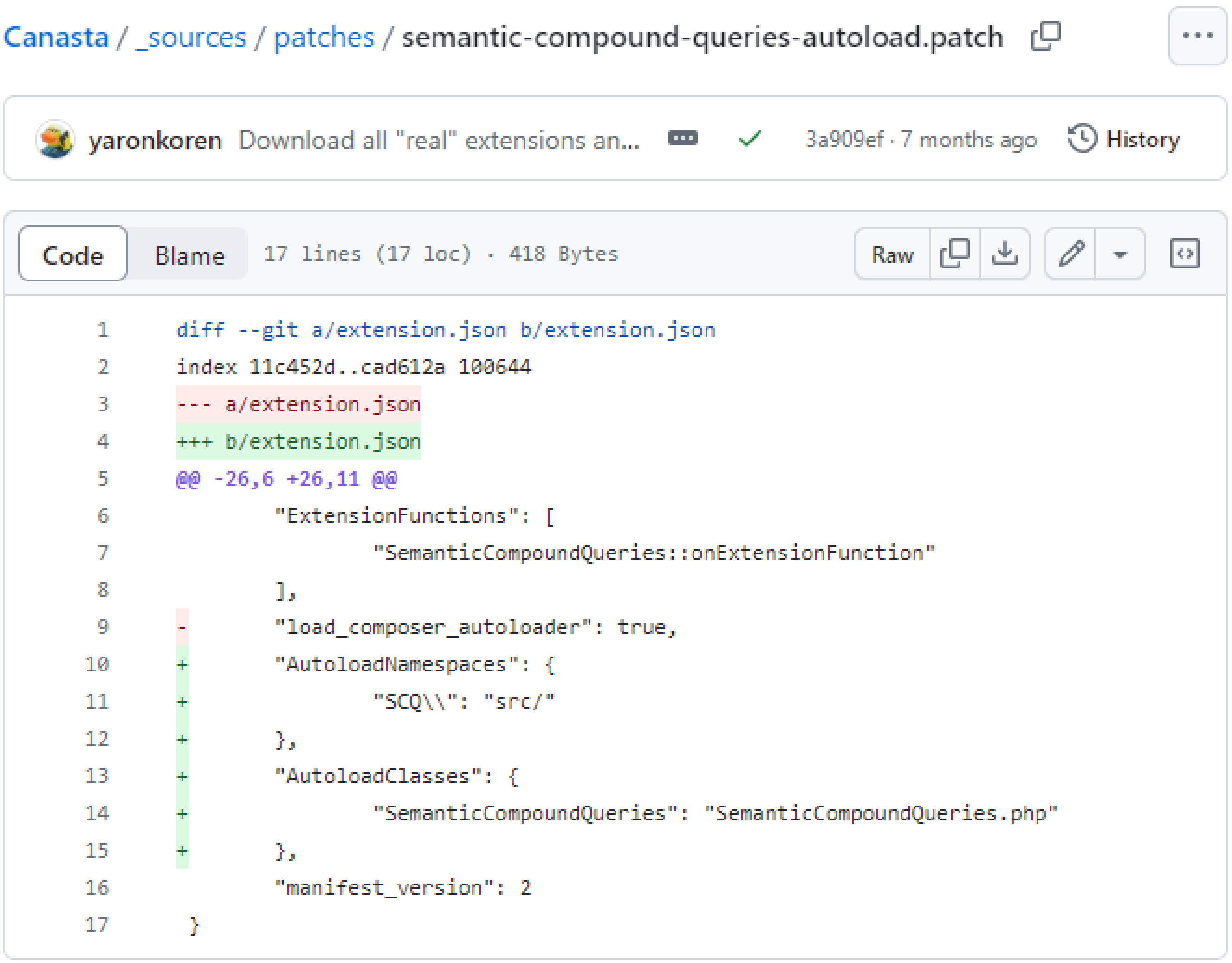
Currently, Canasta contains five patches, all related to Composer-based autoloading of SMW-related extensions.



Canasta / _sources / patches /

vedmaka Adds SemanticWatchlist v 1.3.0 (#2... ✓ 4418831 · 3 months ago History

Name	Last commit message	Last commit date
..		
SemanticWatchList.417851c22c...	Adds SemanticWatchlist v 1.3.0 (#299)	3 months ago
semantic-breadcrumb-links-co...	Download all "real" extensions and sk...	7 months ago
semantic-compound-queries-a...	Download all "real" extensions and sk...	7 months ago
semantic-result-formats-comp...	Download all "real" extensions and sk...	7 months ago
semantic-scribunto-autoload.p...	Download all "real" extensions and sk...	7 months ago


Canasta / _sources / patches / semantic-compound-queries-autoload.patch

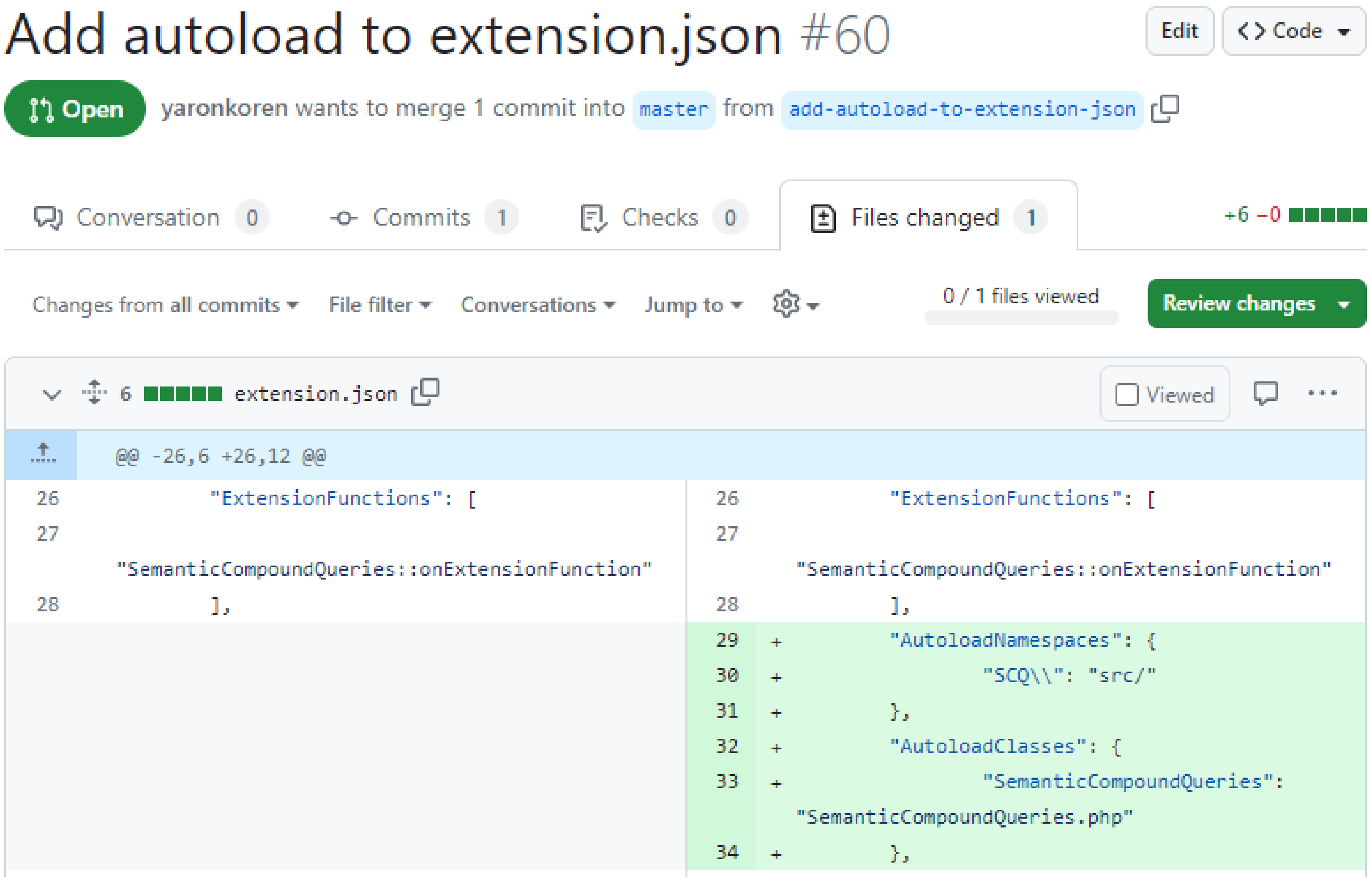
yaronkoren Download all "real" extensions an... 3a909ef · 7 months ago History

Code Blame 17 lines (17 loc) · 418 Bytes Raw

```
diff --git a/extension.json b/extension.json
index 11c452d..cad612a 100644
--- a/extension.json
+++ b/extension.json
@@ -26,6 +26,11 @@
 	"ExtensionFunctions": [
 		"SemanticCompoundQueries::onExtensionFunction"
 	],
-	"load_composer_autoloader": true,
+	"AutoloadNamespaces": {
+		"SCQ\\": "src/"
+	},
+	"AutoloadClasses": {
+		"SemanticCompoundQueries": "SemanticCompoundQueries.php"
+	},
 	"manifest_version": 2
 }
```

Why have these changes not been added to the original extensions?

Still waiting. ☹



Add autoload to extension.json #60

Open yaronkoren wants to merge 1 commit into master from add-autoload-to-extension-json

```
@@ -26,6 +26,12 @@
26          "ExtensionFunctions": [                     26          "ExtensionFunctions": [
27                                                       27
    "SemanticCompoundQueries::onExtensionFunction"          "SemanticCompoundQueries::onExtensionFunction"
28          ],                                          28          ],
                                                        29 +        "AutoloadNamespaces": {
                                                        30 +                "SCQ\\": "src/"
                                                        31 +        },
                                                        32 +        "AutoloadClasses": {
                                                        33 +                "SemanticCompoundQueries":
                                                               "SemanticCompoundQueries.php"
                                                        34 +        },
```

What about non-major bugs?

There is a strong case for **not** trying to fix these in the standard release!

Every bug fix is an opportunity for additional bugs - both in the fix itself, and in any intermediate code changes that may have happened

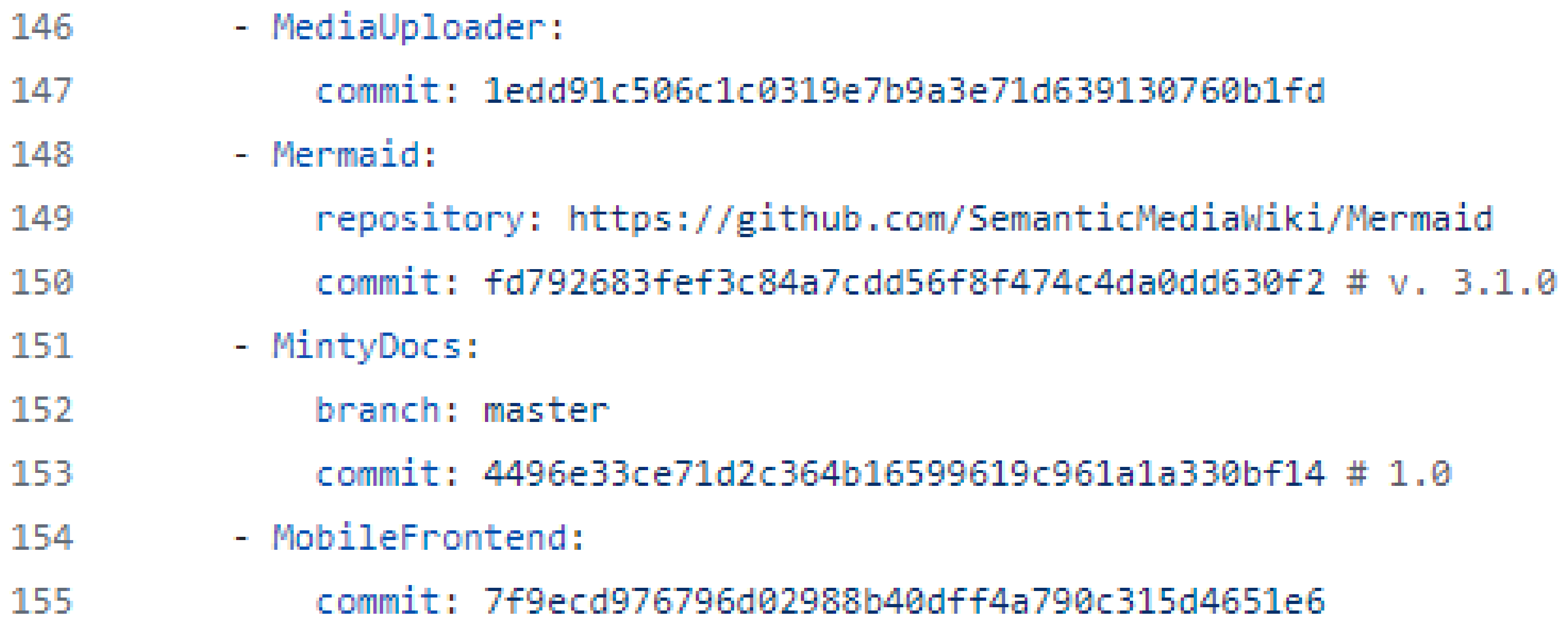
Consistency is an important feature of packages

The fewer code changes are made, the more confidence users can have that it works as intended - since the code they are running will have been run by many others.

Any change to the code - including bug fixes - removes some of that confidence.

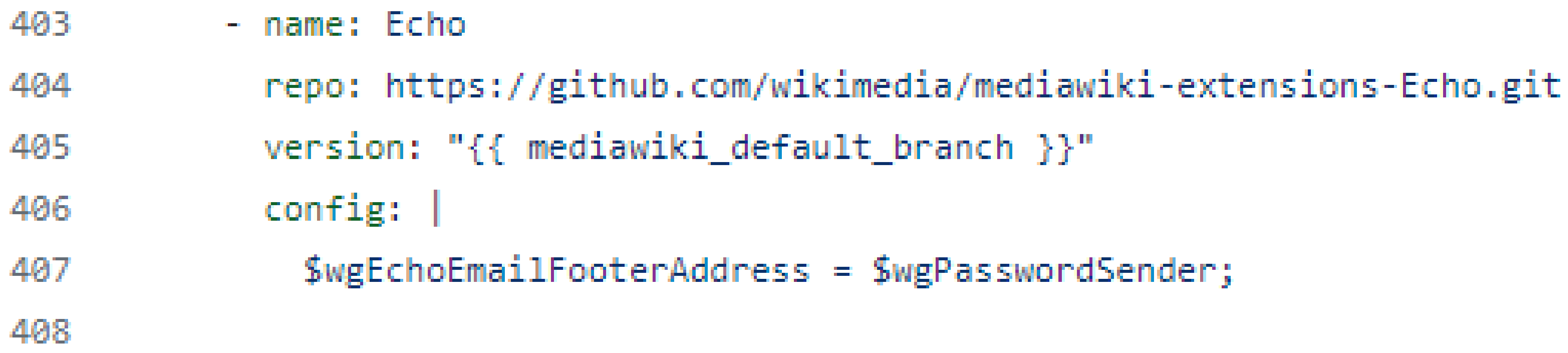
Thus, many bugs should simply wait for the next major release to be fixed.

Canasta YAML file



```
  - MediaUploader:
      commit: 1edd91c506c1c0319e7b9a3e71d639130760b1fd
  - Mermaid:
      repository: https://github.com/SemanticMediaWiki/Mermaid
      commit: fd792683fef3c84a7cdd56f8f474c4da0dd630f2 # v. 3.1.0
  - MintyDocs:
      branch: master
      commit: 4496e33ce71d2c364b16599619c961a1a330bf14 # 1.0
  - MobileFrontend:
      commit: 7f9ecd976796d02988b40dff4a790c315d4651e6
```

Meza has a similar approach (also YAML)



```
- name: Echo
  repo: https://github.com/wikimedia/mediawiki-extensions-Echo.git
  version: "{{ mediawiki_default_branch }}"
  config: |
    $wgEchoEmailFooterAddress = $wgPasswordSender;

```

Most useful parts (“Phase 2”)

The most useful enhancements, beyond “minimum”, would be listing extensions and their versions for:

- Non-LTS MediaWiki versions
- Non-MySQL DB systems (i.e. PostgreSQL and SQLite)
- Specific feature needs (e.g. content publishing)

Questions / comments