

**Wikicode: A Web Application for Collaborative Code Editing and Visualization**

Alexander Dobner  
Department of Computer Science  
Yale University  
April 30, 2015

**Abstract**

One of the most powerful and versatile forms of content management systems on the internet today is the *wiki*, a collaborative easy-to-edit knowledge base consisting of many interconnected pages. There are several notable examples of websites which use the wiki concept as a means of storing and disseminating human knowledge (e.g. Wikipedia, Wiktionary, and WikiHow). For my project I explored how a similar model could be also be applied to create a knowledge base of source code rather than of encyclopedic information.

Many open source wiki platforms already exist for collaborative editing of text-based knowledge, but they are not designed with source code in mind. **Wikicode** is a web application specifically created to store, view, and edit source code. Just as Wikipedia enables users to navigate encyclopedia pages via hyperlinks, **Wikicode** makes it easy for programmers to traverse codebases by allowing them to click on function or class names which link to the corresponding function or class definitions.

Creating such an application required a merging of several technologies including version control systems, static code analysis tools, and lexers/parsers for syntax highlighting. There are many features that could be added to further expand the platform, but **Wikicode** provides a first look at how the wiki model can be applied to computer programming knowledge bases.
1 Introduction

One of the most powerful and versatile forms of content management systems on the internet today is the wiki, a collaborative easy-to-edit knowledge base consisting of many interconnected pages. There are many existing wiki platforms for storing text-based content (e.g. MediaWiki), but they are not designed with source code in mind. Correspondingly, there are many wiki websites which are meant to be repositories of computer programming knowledge (e.g. codecodex.com, rosettacode.org, and en.literateprograms.org), but these websites are primarily used for instructive purposes; they feature implementations of various algorithms or common programming tasks along with explanations of how the implementations work. Wikicode is a web application specifically created to store, edit, and view source code. It also allows programmers to collaborate on code and navigate the code through a system of hyperlinks just like traditional text-based wikis.

Currently, Wikicode is designed for viewing and navigating code written in the Python programming language. The platform is inherently not programming language agnostic due to the need for programming language specific syntax highlighters and code analysis tools, but in the future it can easily be extended on a language-by-language basis.

2 Features

The Wikicode interface is relatively simple and easy to learn. The details of each page is given below.

2.1 Pages

Home page The homepage of Wikicode gives an index of the files currently stored in the system. All the files and any edits to them are stored on the Wikicode server using the version control system Git. Each of these files represents a python module, and users can search for modules or create new ones by typing the name into a search bar at the top of the page.
View code  Clicking on a module in the index or finding a module via the search bar takes the user to a viewing page. Here a user can look at the syntax-highlighted code of the most recent edit of a module. There are also links to edit the code of the module and to view the history of edits to that module.

Edit code  Upon loading the edit page, the user is presented with a text box containing the most recent edition of the module’s source code. The user can make edits and optionally give a commit message specifying the details of the changes they made. After hitting the ‘Update Source’ button the change is committed to the Git repository and the user is redirected to the viewing page.
Figure 3: Edit code

**Code history**  The history page contains a sequential list of all the commits in the Git repository that modified the given file. For each edit the time and commit message are shown, and the user can click a particular edit to view the code at that point in time.
2.2 Hyperlinking

One of the hallmarks of the wiki content management model is the system of interconnected hyperlinking. Wikipedia for example allows users to easily go from one article to another (or to a specific section of an article) by clicking hyperlinks in the article text. Similarly, Wikicode also allows users to traverse its codebase by clicking function, class, or module names in the source code which link to the definition of the corresponding function, class, or module. The most powerful feature of Wikicode is its automatic hyperlink creation system which uses a static code analysis library called Jedi to deduce these hyperlinks. Note that it is not always possible to know which function, for example, is being called at a given point in the code because the Python interpreter decides this at runtime, but in practice it is frequently possible to make these deductions using static code analysis.

Below, in Figures 5 and 6, is an example of Wikicode’s automatic hyperlinking feature.

Figure 5: When the user hovers their mouse over a variable, function, class, or module name, the name is highlighted in pink. Upon clicking the word, the user is taken to the definition. The definition may be in the current module or in another module stored by Wikicode.
3 Technology

Wikicode is written in Python using the Flask microframework, which handles the url routing and HTML templating. All of the backend data including source code and revisions are stored via Git, a powerful version control system, and interactions with the Git repository are handled using the Git- Python library.

The syntax highlighting for Wikicode is performed by the Pygments library which has been modified to enable the line highlighting and hyperlinking needed for Wikicode.

To perform automatic hyperlinking so that users can traverse the stored codebase easily, Wikicode uses the Jedi library’s static code analysis features.
4 Challenges Encountered

Throughout the process of developing Wikicode I encountered several challenges:

**Using Git as a storage backend** I was initially attracted to the idea of using Git as the storage mechanism for Wikicode because it is a mature version control system specifically built for storing source code and revisions. Unfortunately, because it is not a standard relational databases like MySQL or PostgreSQL, most web frameworks are not designed to handle interactions with Git in a clean fashion. Because of this I decided to use the Flask microframework since it does impose any restrictions on database interaction or use any built-in object-relational mapping. Ultimately, this meant all the backend data handling had to be done manually by writing to files on the disc and by interacting with the git database via GitPython.

**Performing code analysis on Python** Because Python is a highly dynamic language, the static code analysis required to perform a “go to definition” like Wikicode does is complicated. Luckily the Jedi library provided a way for me to do this, but because Jedi is mainly meant to be used as an autocomplete library it did not have all the features that I would have wanted. Specifically I had hoped to add a feature to Wikicode that would enable users to view a graph of the interdependencies between different classes and functions stored in the codebase, but given the time constraint and the limitations of the libraries available I was not able to do so.

**UI and Design** Part of the reason I chose to create Wikicode as a web application was to gain experience with frontend web technology. I had difficulties at times with page formatting issues due to the complexities of CSS, and I am also relatively inexperienced at using Javascript to perform Ajax requests (which is used whenever the user clicks on a name in the source code and the server is queried to find where the definition occurs). Lastly, there are some points in the code where I needed to jump through hoops to produce the desired HTML output; in particular, I used the Pygments library for generating HTML for the syntax highlighted code, which then needed to be modified slightly before ultimately being sent on to the Flask templating engine to generate the whole page.
5 Future Work

There is still much more that can be done to make Wikicode a more fully featured code-collaboration tool. Firstly, because my goal was to focus on the features that made Wikicode unique amongst wiki software, I did not spend as much time on standard wiki features like for instance the ability to view differences between two versions of a file or the ability to block certain users from modifying certain pages. These features would of course still be helpful in a tool for code-collaboration.

Secondly, there are many features that could be added to make editing code on the site easier. For instance the ability to have syntax highlighting and syntax error reporting in the edit view (which would not be difficult to add using existing libraries), and the ability to clone the server’s git repository on a local machine so that users could edit pages in their own text editor and then push those changes would be useful as well. The site would also have better usability if users could add documentation or notes to the code, and if they could view relationships in the code via a graph showing the interdependencies of functions, classes, and modules.

Lastly, looking even further into the future, perhaps it would one day be possible to use a collaboratively edited wiki of source code as a sort of massive crowdsourced code library (just as wikipedia is a crowdsourced library of knowledge) in which users can easily import classes or functions that they need and add new functions to benefit the community.