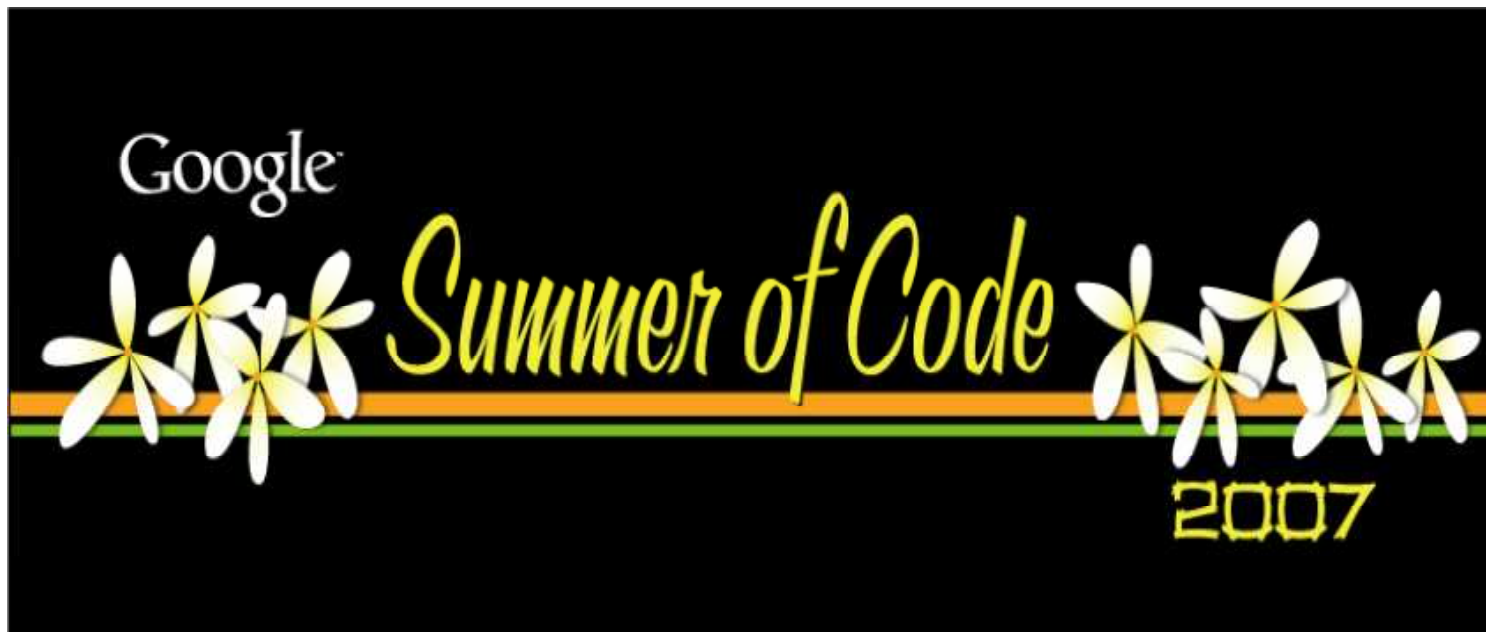




**KDevelop 4: Beyond C++  
Extending the language base.**





**KDevelop 4: Beyond C++  
Extending the language base.**



## **KDevelop4: Beyond C++...extending the Language Base**

KDevelop Unleashed.

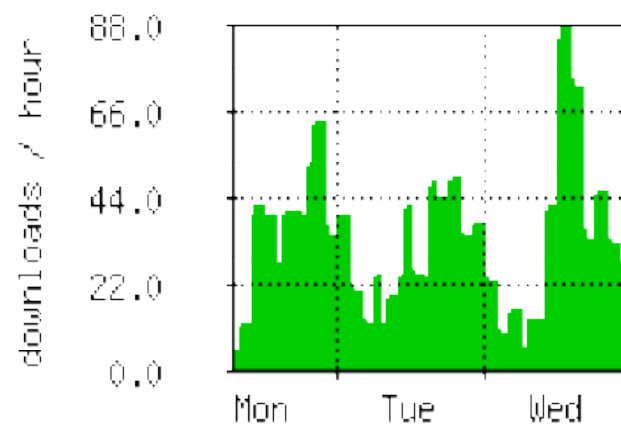


## KDevelop 4: Beyond C++ Extending the language base.



### KDevelop ... A Superb IDE in the making.

- History and the road map of Kdevelop.
- Good Code Completion
- Great Code Navigation and features powered by Definition Use Chain
- Efficient memory management.
- The stats say it:





## KDevelop 4: Beyond C++ Extending the language base.



### The Current Languages Support Status

- C/C++
  - Class Wizard
  - Attribute/Method Wizard
  - Language Parser
  - Code Completion
  - Problem Reporter
  - Debugger
  - Source Code Formatter
  
- Java
  - Class Wizard
  - Attribute/Method Wizard
  - ANTLR Language parser
  - Source Code Formatter
  
- Python
  - Language Parser
  - Basic Extended Code Highlighting



## KDevelop 4: Beyond C++ Extending the language base.



### Face-Off : Kick-Starting the Pre-Requisites.

- A working checkout of KDE4 and Kdevelop4
- The Lexer needs to be implemented/written
- What should the grammar look like?
- Parser Generators could be used Or a hand written parser.
- Kdev-Pg is a handy tool.
  - All about Kdev-Pg



## KDevelop 4: Beyond C++ Extending the language base.



### Diving into the language support.

- Done with the parser for the language; what next?
  - Test, Test & Test again.
- The Kdevelop API.
- How to make language support a Plug-in
- Implementing languagesupport, parseJob; why do we need them?
- Iplugin and ILanguagesupport



## KDevelop 4: Beyond C++ Extending the language base.



### **DU aka Definition Use Chain: the key to glory.**

- What is the Fuss all about?
- How does it work?
- What does it produce?
- What use it is?
- Extending the use beyond C++.
- Viewing the outputs for it.



## KDevelop 4: Beyond C++ Extending the language base.



### Extensions required in DUchain.

- Implementing editorIntegrator ; why do we need it?
- Parse Session implementations.
- Contextbuilder implementation.
- Declarationbuilder implementation.
- Usebuilder implementation



## KDevelop 4: Beyond C++ Extending the language base.



### Putting the DUChain output in Use.

- The DUChain has been built.
- Use it for:
  - Code Highlighting
  - Advanced Syntax Highlighting
  - Code Completion
  - And many other features

**PS: Only Syntax highlighting has been implemented for python by me.**

- Advanced Syntax Highlighting
- Adding custom coloring to declarations/definitions & uses.
- Current DU Chain architecture that might make the use challenging.



## KDevelop 4: Beyond C++ Extending the language base.



### Show .... Don't Tell

#### Working demonstrations, from my work on adding Python to Kdevelop4:-

- Executing a working copy of Kdevelop4.
- See the Python Plug-in in action, when a file is loaded.
  - On the terminal debug messages.
- The DUChain works and shows the contexts created when a file is loaded.
  - As a Kdevelop4 Widget.
- The Extended Code Highlighting after a \*.py has been loaded.



## KDevelop 4: Beyond C++ Extending the language base.



### Acknowledgments & References:

- KDE 4 API reference available at <http://api.kde.org>
- Python plug-in code for KDevelop4 available at <http://code.google.com/p/google-summer-of-code-2007-kde/downloads/list>
- The C# & Java plug-ins for KDevelop4
- Kdevelop 4 API reference available at <http://kdevelop.org/HEAD/doc/api/html/>
- Thanks to: