GammaLib - Feature #559

Change request # 558 (Closed): Gammalib CDC: Coding and Design Conventions

Create Eclipse C++ code formatting style for gammalib

10/12/2012 01:17 PM - Deil Christoph

Status:	In Progress	**Start date:**	10/12/2012
Priority:	Normal	**Due date:**	
Assigned To:	Deil Christoph	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

Eclipse CDT allows the specification of code formatting rules for C++ code and share this for all developers on a project as one XML file.

I'll try to create such a specification for the rules given in the Gammalib CDC document so that one can simply press CMD+Shift+F before saving and committing and doesn't have to remember and manually take care about the code formatting details all the time.

- What should I call this file and where should I put it? (In the HESS software we have HESSStyle.xml in the top-level directory.)

History

#1 - 10/12/2012 04:28 PM - Knödlseder Jürgen

Christoph Deil wrote:

> - What should I call this file and where should I put it? (In the HESS software we have HESSStyle.xml in the top-level directory.)

I guess it should live in the top-level directory.

Is the file Eclipse specific? If yes, I would add eclipse in the filename, maybe simply calling it eclipse.xml?

#2 - 12/05/2012 09:20 PM - Deil Christoph

This style check should also be run on Jenkins (see comment https://cta-redmine.irap.omp.eu/issues/617#note-1)
I'll try to find out if there's a command line tool (or if not if it's possible to make Eclipse do it via a command line call).

#3 - 12/06/2012 09:03 PM - Deil Christoph

Apparently there's two good tools:
http://stackoverflow.com/questions/93260/a-free-tool-to-check-c-c-source-code-against-a-set-of-coding-standards

There's cpplint and the C++ and Python style guide from Google:
https://code.google.com/p/google-styleguide/

And there's:
http://www.inspirel.com/vera/

#4 - 12/06/2012 09:45 PM - Deil Christoph

With Google the file eclipse-cpp-google-style.xml teaches Eclipse the style, that's really useful.

The example code in this style looks like this:

```
/*
 * A sample source file for the code formatter preview
 */
#include <math.h>

class Point {
 public:
  Point(double x, double y)
      : x(x),
        y(y) {
  }
  double distance(const Point& other) const;

  double x;
  double y;
};

double Point::distance(const Point& other) const {
  double dx = x - other.x;
  double dy = y - other.y;
  return sqrt(dx * dx + dy * dy);
}
```

Eclipse CDT http://www.eclipse.org/cdt/ also has the K & R style built in http://en.wikipedia.org/wiki/Indent_style#K.26R_style , which might be another option. It looks like this:

<code class="cpp"> /* * A sample source file for the code formatter preview */ #include <math.h> class Point { public: Point(double x, double y) :   x(x), y(y) {   }   double distance(const Point& other) const;   double x;   double y; }; double Point::distance(const Point& other) const { double dx = x - other.x;   double dy = y - other.y;   return sqrt(dx * dx + dy * dy); } </pre> The @cpplint.py@ tool from Google works like this: <pre> $ ~/code/google-styleguide/cpplint/cpplint.py --counting=toplevel src/sky/GSkyDir.cpp src/sky/GSkyDir.cpp:54: Complex multi-line /*...*/-style comment found. Lint may give bogus warnings. Consider replacing these with //-style comments, with #if 0...#endif, or with more clearly structured multi-line comments. [readability/multiline_comment] [5] src/sky/GSkyDir.cpp:54: At least two spaces is best between code and comments [whitespace/comments] [2] src/sky/GSkyDir.cpp:54: Should have a space between // and comment [whitespace/comments] [4] src/sky/GSkyDir.cpp:55: Weird number of spaces at line-start. Are you using a 2-space indent? [whitespace/indent] [3] src/sky/GSkyDir.cpp:56: Weird number of spaces at line-start. Are you using a 2-space indent? [whitespace/indent] [3] src/sky/GSkyDir.cpp:58: { should almost always be at the end of the previous line [whitespace/braces] [4] src/sky/GSkyDir.cpp:67: Complex multi-line /*...*/-style comment found. Lint may give bogus warnings. Consider replacing these with //-style comments, with #if 0...#endif, or with more clearly structured multi-line comments. [readability/multiline_comment] [5] src/sky/GSkyDir.cpp:67: At least two spaces is best between code and comments [whitespace/comments] [2] src/sky/GSkyDir.cpp:67: Should have a space between // and comment [whitespace/comments] [4] src/sky/GSkyDir.cpp:68: Weird number of spaces at line-start. Are you using a 2-space indent? [whitespace/indent] [3] src/sky/GSkyDir.cpp:69: Weird number of spaces at line-start. Are you using a 2-space indent? [whitespace/indent] [3] src/sky/GSkyDir.cpp:70: Weird number of spaces at line-start. Are you using a 2-space indent? [whitespace/indent] [3] src/sky/GSkyDir.cpp:71: Weird number of spaces at line-start. Are you using a 2-space indent? [whitespace/indent] [3] src/sky/GSkyDir.cpp:73: { should almost always be at the end of the previous line [whitespace/braces] [4] src/sky/GSkyDir.cpp:85: Complex multi-line /*...*/-style comment found. Lint may give bogus warnings. Consider replacing these with //-style comments, with #if 0...#endif, or with more clearly structured multi-line comments. [readability/multiline_comment] [5] src/sky/GSkyDir.cpp:85: At least two spaces is best between code and comments [whitespace/comments] [2] ... src/sky/GSkyDir.cpp:788: Weird number of spaces at line-start. Are you using a 2-space indent? [whitespace/indent] [3] src/sky/GSkyDir.cpp:790: { should almost always be at the end of the previous line [whitespace/braces] [4] src/sky/GSkyDir.cpp:792: Missing spaces around == [whitespace/operators] [3] src/sky/GSkyDir.cpp:575: Add #include <string> for string [build/include_what_you_use] [4] Done processing src/sky/GSkyDir.cpp Category 'build' errors found: 1 Category 'readability' errors found: 41 Category 'whitespace' errors found: 279 Total errors found: 321 </pre> It only works with the Google style @cpplint.py@ only, which conflicts with the current @gammalib@ style all over the place. As far as I can see it, it cannot be customised, actually it doesn't even have a command line option to accept @.hpp@ files as @gammalib@ uses, it only takes @.h@ files as input: <pre> Linted extensions are .cc, .cpp, and .h. Other file types will be ignored. </pre> So looks like the Google style and tool are out of the question. I'll look at Vera++ or the possibility to run Eclipse style check or fix from the command line tomorrow.

#5 - 12/06/2012 09:46 PM - Deil Christoph

- Status changed from New to In Progress

#6 - 12/07/2012 10:56 PM - Knödlseder Jürgen

Thanks for looking into this. The styles here are not fully compliant to the style adopted in GammaLib, which probably explains most of the warnings.

Is it possible to adapt these style files to the GammaLib style? This would be really useful.

#7 - 01/05/2014 11:37 PM - Deil Christoph

- Priority changed from Low to Normal

- Target version set to 2nd coding sprint

I've been doing some C++ learning yesterday and came across this:
http://channel9.msdn.com/Events/GoingNative/2013/The-Care-and-Feeding-of-C-s-Dragons

The Clang guys recently have developed a nice formatter:
http://clang.llvm.org/docs/ClangFormat.html

It comes with a few different default styles and an easy config file option to configure it:
http://clang.llvm.org/docs/ClangFormatStyleOptions.html

I don't have time to try out these tools, but I can do it before or at the coding sprint, and then we can fix the exact code formatting rules for gammalib in person.

#8 - 07/19/2014 02:09 AM - Knödlseder Jürgen

- Target version deleted (2nd coding sprint)