Static site generator

Generated by Doxygen 1.9.3

Chapter 1

LICENSE

GNU GENERAL PUBLIC LICENSE Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. https://fsf.org/ Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

1. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

1. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

1. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

1. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

```
a) The work must carry prominent notices stating that you modified
it, and giving a relevant date.

b) The work must carry prominent notices stating that it is
released under this License and any conditions added under section
7.  This requirement modifies the requirement in section 4 to
"keep intact all notices".

c) You must license the entire work, as a whole, under this
License to anyone who comes into possession of a copy.  This
License will therefore apply, along with any applicable section 7
additional terms, to the whole of the work, and all its parts,
regardless of how they are packaged.  This License gives no
permission to license the work in any other way, but it does not
invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display
Appropriate Legal Notices; however, if the Program has interactive
interfaces that do not display Appropriate Legal Notices, your
work need not make them do so.
```

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

1. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

```
a) Convey the object code in, or embodied in, a physical product
(including a physical distribution medium), accompanied by the
Corresponding Source fixed on a durable physical medium
customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product
(including a physical distribution medium), accompanied by a
written offer, valid for at least three years and valid for as
long as you offer spare parts or customer support for that product
model, to give anyone who possesses the object code either (1) a
copy of the Corresponding Source for all the software in the
product that is covered by this License, on a durable physical
medium customarily used for software interchange, for a price no
more than your reasonable cost of physically performing this
conveying of source, or (2) access to copy the
Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the
written offer to provide the Corresponding Source.  This
alternative is allowed only occasionally and noncommercially, and
only if you received the object code with such an offer, in accord
with subsection 6b.

d) Convey the object code by offering access from a designated
place (gratis or for a charge), and offer equivalent access to the
Corresponding Source in the same way through the same place at no
further charge.  You need not require recipients to copy the
Corresponding Source along with the object code.  If the place to
copy the object code is a network server, the Corresponding Source
may be on a different server (operated by you or a third party)
that supports equivalent copying facilities, provided you maintain
clear directions next to the object code saying where to find the
Corresponding Source.  Regardless of what server hosts the
Corresponding Source, you remain obligated to ensure that it is
available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided
you inform other peers where the object code and Corresponding
Source of the work are being offered to the general public at no
charge under subsection 6d.
```

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

1. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

```
a) Disclaiming warranty or limiting liability differently from the
terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or
author attributions in that material or in the Appropriate Legal
Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or
requiring that modified versions of such material be marked in
reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or
authors of the material; or

e) Declining to grant rights under trademark law for use of some
trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that
material by anyone who conveys the material (or modified versions of
it) with contractual assumptions of liability to the recipient, for
any liability that these contractual assumptions directly impose on
those licensors and authors.
```

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

1. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

1. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

1. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

1. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

1. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

1. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

1. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

1. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

1. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

1. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year>  <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
<program>  Copyright (C) <year>  <name of author>
This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type 'show c' for details.
```

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see https://www.gnu.org/licenses/.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read https://www.gnu.org/licenses/why-not-lgpl.html.

Chapter 2

StaticSiteGenerator

Chapter 3

Static site generator

This program is created as a semestral project for the NPRG035 and NPRG038 courses.

3.1 Introduction

Static site generator is a program which generates static websites. Users create *Markdown* files which are then converted into *HTML*. Because of that, users don't need to know *HTML* to create websites.

3.2 Functional description

After running the generator without any arguments a project directory will be created containing all the required directories. Users will then provide the *Markdown* files and images into their respective directories. After that, running the generator with the argument "compile" will compile everything and will create the output directory. Static site generator will be able to:

- read and parse the provided *Markdown* files - these are called posts
- convert them into *HTML*
- create an Index page in *HTML*
- add a header and a footer to all *HTML* pages

3.3 Functional requirements

3.3.1 Parsing Markdown files

The markdown files must contain the tags "Author" and "Name" right at the top of the page. The static site generator will be able to parse and translate into *HTML* the following subset of *Markdown*:

Markdown	HTML
# Heading level 1	**Heading level 1**
## Heading level 2	**Heading level 2**
### Heading level 3	**Heading level 3**
#### Heading level 4	**Heading level 4**
##### Heading level 5	**Heading level 5**
###### Heading level 6	**Heading level 6**
bold text	**bold text**
bold text	**bold text**
italic text	*italic text*
italic text	*italic text*
`word`	`word`
Unorderered list with "+", "∗"	
+ First item	• First item
+ Second item	• Second item
! [] (image.png)	will display the provided image
Link `Guide`	clickable Guide

3.3.2 Creating the Index page

The generated Index page will contain a list of all posts. The generator will use tags located inside of each *Markdown* file.

3.3.3 Templating

The generator will only create the *HTML* files. Styling needs to be done by the user using CSS.

3.4 Data inputs

The only data inputs are the *Markdown* files provided by the user along with images inside a special directory.

Chapter 4

Namespace Index

4.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 6

Class Index

6.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 7

Namespace Documentation

7.1 MarkdownCompiler Namespace Reference

Classes

- class BodyNode

  A node representing the body of the file. Contains all paragraphs.
- class BoldParser

  A paprser specialised to recognise a token patter representing bold text.
- class CodeParser

  A paprser specialised to recognise a token patter representing coded text.
- class Compiler
- class ConfigContent
- class ConfigLoader
- class Generator
- class HeaderParser

  A paprser specialised to recognise a token patter representing a header.
- class ImageParser

  A paprser specialised to recognise a token patter representing an image link.
- class IndexFileCreator
- interface INode

  Represents a node of an Abstract Syntax Tree
- interface ISentenceParser

  Represents a parser which looks for a specific set of tags which form a specific sentence.
- interface ISentencesParser

  Represents a parser which parses sentences of tokens. Sentence can be for example a bold text, normal text, link, etc.
- class ItalicParser

  A paprser specialised to recognise a token patter representing text in italic.
- interface IToken

  An implementation must represent a token with a type and a value created by the Tokenizer while reading the correct char from the file.
- interface ITokenizer

  Must represent a Tokenizer. Must be accompanied by a creation of a specific parser.
- class LinkParser

  A paprser specialised to recognise a token patter representing a link.

- class ListParser

  A paprser specialised to recognise a token patter representing a unordered list.
- class Node

  A node representing a sentence in the grammar.
- class ParagraphNode

  A node representing a paragraph. Contains all sentences inside of the paragraph.
- class ParagraphParser

  A paprser specialised to recognise a paragraph node from the list of tokens.
- class Parser
- class SentenceParser

  A paprser specialised to recognise a sentence node from the list of tokens.
- class SentencesEOFParser

  A paprser specialised to recognise a sentences followed by the End of File character.
- class SentencesNewlineParser

  A paprser specialised to recognise a sentences followed by an empty line.
- class SpecialCharScanner
- class TextParser

  A paprser specialised to recognise a token patter representing plain text.
- class TextScanner
- class Token
- class Tokenizer

Enumerations

- enum NodeType {
  TEXT , **H1** , **H2** , **H3** ,
  H4 , **H5** , **H6** , **BOLD** ,
  ITALIC , **CODE** , **LIST** , **IMAGE** ,
  LINK }

  Different types of nodes.
- enum TokenType {
  TEXT , **HASHTAG** , **UNDERSCORE** , **STAR** ,
  PLUS , **EXCLAMATIONMARK** , **NEWLINE** , **LEFTSQUAREBRACKET** ,
  RIGHTSQUAREBRACKET , **LEFTBRACKET** , **RIGHTBRACKET** , **BACKTICK** ,
  EOF }

  Different types of tokens.

7.2 StaticSiteGenerator Namespace Reference

Classes

- class Creator
- class DirectoryCreator
- class Program

  Class containing the Main method.

7.3 StaticSiteGeneratorTests Namespace Reference

Classes

- class TokenizerTests

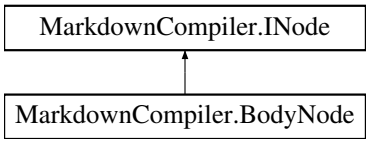
Chapter 8

Class Documentation

8.1 MarkdownCompiler.BodyNode Class Reference

A node representing the body of the file. Contains all paragraphs.

Inheritance diagram for MarkdownCompiler.BodyNode:



Static Public Member Functions

- static BodyNode **BodyNodeFactory** (List< ParagraphNode > paragraphs, int consumed)

Properties

- List< ParagraphNode > **Paragraphs** `[get]`
- int **Consumed** `[get]`

Private Member Functions

- **BodyNode** (List< ParagraphNode > paragraphs, int consumed)

8.1.1 Detailed Description

A node representing the body of the file. Contains all paragraphs.

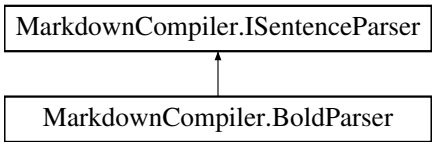
The documentation for this class was generated from the following file:

- BodyNode.cs

8.2 MarkdownCompiler.BoldParser Class Reference

A paprser specialised to recognise a token patter representing bold text.

Inheritance diagram for MarkdownCompiler.BoldParser:



Public Member Functions

- Node Parse (IToken[ ] tokens)

  Parses tokens and looks for a pattern representing bold text.

Private Attributes

- List< TokenType > **UnderscoreTemplate** = new List<TokenType>() { TokenType.UNDERSCORE, Token↩ Type.UNDERSCORE, TokenType.TEXT, TokenType.UNDERSCORE, TokenType.UNDERSCORE }
- List< TokenType > **StarTemplate** = new List<TokenType>() { TokenType.STAR, TokenType.STAR, Token↩ Type.TEXT, TokenType.STAR, TokenType.STAR }

8.2.1 Detailed Description

A paprser specialised to recognise a token patter representing bold text.

8.2.2 Member Function Documentation

8.2.2.1 Parse()

```
Node MarkdownCompiler.BoldParser.Parse (
            IToken[] tokens )
```

Parses tokens and looks for a pattern representing bold text.

Parameters

tokens	A list of tokens.

Returns

Null if no pattern was found. A node containing the text which should be bold.

Implements MarkdownCompiler.ISentenceParser.

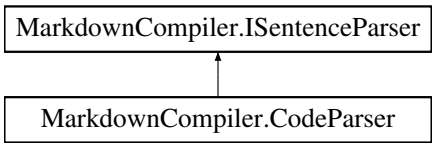
The documentation for this class was generated from the following file:

- BoldParser.cs

8.3 MarkdownCompiler.CodeParser Class Reference

A paprser specialised to recognise a token patter representing coded text.

Inheritance diagram for MarkdownCompiler.CodeParser:



Public Member Functions

- Node Parse (IToken[ ] tokens)
  Parses tokens and looks for a pattern representing coded text.

Private Attributes

- List< TokenType > **BacktickTemplate** = new List<TokenType>() { TokenType.BACKTICK, TokenType.↩ TEXT, TokenType.BACKTICK }

8.3.1 Detailed Description

A paprser specialised to recognise a token patter representing coded text.

8.3.2 Member Function Documentation

8.3.2.1 Parse()

```
Node MarkdownCompiler.CodeParser.Parse (
            IToken[] tokens )
```

Parses tokens and looks for a pattern representing coded text.

Parameters

tokens	A list of tokens.

Returns

Null if no sequence was found or a node containing the text which should be coded.

Implements MarkdownCompiler.ISentenceParser.

The documentation for this class was generated from the following file:

- CodeParser.cs

8.4 MarkdownCompiler.Compiler Class Reference

Public Member Functions

- **Compiler** (bool isInsideWebDirectory, string configFileName)
- async Task CompileFileAsync (string path, string outputDirectoryName)

  Asynchronously reads, tokenizes and parses a markdown file and converts its contents into HTML and writes the result asynchronously into a file with the extension ".html" at the specified output directory. Will let the user know if an error during the compilation occured via the console.
- async Task CreateIndexFileAsync (string outputDirectory, string configFileName)

  Asynchronously creates the index html file in the specified output directory. Uses the contents of the config.json file. All posts are displayed on the index page along with their links and titles.

Properties

- bool **IsInsideWebDirectory** `[get]`
- string **ConfigFileName** `[get, set]`

Private Member Functions

- async Task CompileAndOutputFileAsync (List< IToken > resultTokens, string fileName, string title, string outputDirectoryName)

  Parses the tokens paragraph by paragraph, then generates the HTML of the paragraph and asynchronously writes to a file at the specified location.
- async Task< string > TryGetTitleAsync (StreamReader reader)

  Asynchronously reads the first three lines of the specified file and returns the title of the page specified by the "Name" tag.
- async Task<(List< IToken >, string title)> GetFileTokens (string name)

  Asynchronously reads the specified file line by line and returns the title of the page (specified at the top of the page) and the list of tokens representing the contents of the page.

Private Attributes

- Dictionary< string, string > **FileTitles**

8.4.1 Member Function Documentation

8.4.1.1 CompileAndOutputFileAsync()

```
async Task MarkdownCompiler.Compiler.CompileAndOutputFileAsync (
            List< IToken > resultTokens,
            string fileName,
            string title,
            string outputDirectoryName )  [private]
```

Parses the tokens paragraph by paragraph, then generates the HTML of the paragraph and asynchronously writes to a file at the specified location.

Parameters

resultTokens	A list of tokens.
fileName	Name of the file we want to put our generated HTML.
title	The extracted title of the page.
outputDirectoryName	The output directory where we want to put our newly generated file.

Returns

A task.

8.4.1.2 CompileFileAsync()

```
async Task MarkdownCompiler.Compiler.CompileFileAsync (
            string path,
            string outputDirectoryName )
```

Asynchronously reads, tokenizes and parses a markdown file and converts its contents into HTML and writes the result asynchronously into a file with the extension ".html" at the specified output directory. Will let the user know if an error during the compilation occured via the console.

Parameters

path	The path to a file we want to compile.
outputDirectoryName	The output directory where we want to put our newly generated HTML file.

Returns

A task.

8.4.1.3 CreateIndexFileAsync()

```
async Task MarkdownCompiler.Compiler.CreateIndexFileAsync (
            string outputDirectory,
            string configFileName )
```

Asynchronously creates the index html file in the specified output directory. Uses the contents of the config.json file. All posts are displayed on the index page along with their links and titles.

Parameters

outputDirectory	Where to put the index file.
configFileName	The name of the config file containing the information about the website name and the name of the author.

Returns

8.4.1.4 GetFileTokens()

```
async Task<(List< IToken >, string title)> MarkdownCompiler.Compiler.GetFileTokens (
            string name )  [private]
```

Asynchronously reads the specified file line by line and returns the title of the page (specified at the top of the page) and the list of tokens representing the contents of the page.

Parameters

name	The path to a file which we want to tokenize.

Returns

A tuple consisting of a List of Itokens and a string representing the title of the page. Returns (null,null) if an error occured during extraction of the title of tokenizing

8.4.1.5 TryGetTitleAsync()

```
async Task< string > MarkdownCompiler.Compiler.TryGetTitleAsync (
            StreamReader reader )  [private]
```

Asynchronously reads the first three lines of the specified file and returns the title of the page specified by the "Name" tag.

Parameters

reader	A stream reader of the file from which we want to extract the title.

Returns

Task where the string represents the title or null if there was an error.

The documentation for this class was generated from the following file:

- Compiler.cs

8.5 MarkdownCompiler.ConfigContent Class Reference

Properties

- string **Author** `[get, set]`
- string **WebsiteName** `[get, set]`

The documentation for this class was generated from the following file:

- ConfigLoader.cs

8.6 MarkdownCompiler.ConfigLoader Class Reference

Static Public Member Functions

- static async Task< ConfigContent > **GetConfigContentAsync** (bool isInsideWebDirectory, string config↩ Name)

The documentation for this class was generated from the following file:

- ConfigLoader.cs

8.7 StaticSiteGenerator.Creator Class Reference

Public Member Functions

- **Creator** (Compiler compiler)
- async Task **GenerateHTMLAsync** ()

  Creator will compile all markdown files inside the posts directory into html and put the inside the output directory.
- void CopyPicturesToOutput (string inputDirectoryName, string outputDirectoryName)

  Copies files from the input directory into the ouput directory.

Static Public Member Functions

- static void **TemplateConfigFile** ()

  Creates the configuration json file and adds the template config structure containing variables 'Name' and 'Author'.

Properties

- Compiler **Compiler** `[get, set]`

Private Member Functions

- string[ ] GetPostNames ()

  Searches the posts directory and returns an array of names of all markdown files.

Static Private Member Functions

- static bool **GenerateConfigFile** ()

  Creates the configuration json file.

Static Private Attributes

- static readonly string **TemplateConfigContent** = "{\"Author\":\"John Doe\", \"WebsiteName\":\"A very nice website\"}"

8.7.1 Member Function Documentation

8.7.1.1 CopyPicturesToOutput()

```
void StaticSiteGenerator.Creator.CopyPicturesToOutput (
            string inputDirectoryName,
            string outputDirectoryName )
```

Copies files from the input directory into the ouput directory.

Parameters

inputDirectoryName	Name of the input directory.
outputDirectoryName	Name of the output directory.

8.7.1.2 GetPostNames()

```
string[] StaticSiteGenerator.Creator.GetPostNames ( )  [private]
```

Searches the posts directory and returns an array of names of all markdown files.

Returns

All names of markdown files inside the posts directory or null if posts directory cannot be found.

The documentation for this class was generated from the following file:

- Creator.cs

8.8 StaticSiteGenerator.DirectoryCreator Class Reference

Static Public Member Functions

- static void **CreateTemplateDirectories** ()

  Creates the template website directory structure.

The documentation for this class was generated from the following file:

- DirectoryCreator.cs

8.9 MarkdownCompiler.Generator Class Reference

Public Member Functions

- string **GenerateParagraphText** (ParagraphNode paragraphNode)
- string **GetHeader** (string title)
- string **GetBodyEnd** ()
- string **GetFooter** (ConfigContent authorAndTitle)

Private Member Functions

- string **GenerateHeader** (ParagraphNode paragraphNode)
- string **GenerateOther** (Node node)
- string **GenerateHeaderString** (string headerType, ParagraphNode paragraphNode)

Static Private Attributes

- static readonly NodeType[ ] **headerTypes** = new NodeType[ ] { NodeType.H1, NodeType.H2, NodeType.H3, NodeType.H4, NodeType.H5, NodeType.H6 }

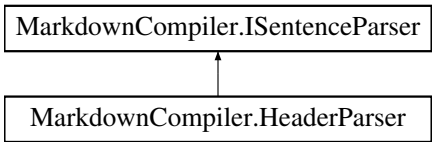
The documentation for this class was generated from the following file:

- Generator.cs

8.10 MarkdownCompiler.HeaderParser Class Reference

A paprser specialised to recognise a token patter representing a header.

Inheritance diagram for MarkdownCompiler.HeaderParser:



Public Member Functions

- Node Parse (IToken[ ] tokens)

  Parses tokens and looks for a pattern representing a header.

8.10.1 Detailed Description

A paprser specialised to recognise a token patter representing a header.

8.10.2 Member Function Documentation

8.10.2.1 Parse()

```
Node MarkdownCompiler.HeaderParser.Parse (
            IToken[] tokens )
```

Parses tokens and looks for a pattern representing a header.

Parameters

tokens	A list of tokens.

Returns

Null if no sequence was found or a node containing the text which should be coded.

Implements MarkdownCompiler.ISentenceParser.

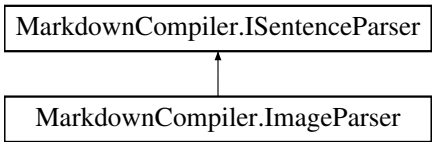
The documentation for this class was generated from the following file:

- HeaderParser.cs

8.11 MarkdownCompiler.ImageParser Class Reference

A paprser specialised to recognise a token patter representing an image link.

Inheritance diagram for MarkdownCompiler.ImageParser:



Public Member Functions

- Node Parse (IToken[ ] tokens)

  Parses tokens and looks for a pattern representing an image link.

Private Attributes

- List< TokenType > **ImageNoNameTemplate** = new List<TokenType>() { TokenType.EXCLAMATIONMARK, TokenType.LEFTSQUAREBRACKET, TokenType.RIGHTSQUAREBRACKET, TokenType.LEFTBRACKET, TokenType.TEXT, TokenType.RIGHTBRACKET }
- List< TokenType > NamedImageTemplate

8.11.1 Detailed Description

A paprser specialised to recognise a token patter representing an image link.

8.11.2 Member Function Documentation

8.11.2.1 Parse()

```
Node MarkdownCompiler.ImageParser.Parse (
            IToken[] tokens )
```

Parses tokens and looks for a pattern representing an image link.

Parameters

tokens	A list of tokens.

Returns

Null if no sequence was found or a node containing the text representing the link and an optional name.

Implements MarkdownCompiler.ISentenceParser.

8.11.3 Member Data Documentation

8.11.3.1 NamedImageTemplate

```
List<TokenType> MarkdownCompiler.ImageParser.NamedImageTemplate  [private]
```

Initial value:

```
= new List<TokenType>() { TokenType.EXCLAMATIONMARK, TokenType.LEFTSQUAREBRACKET, TokenType.TEXT,
                                                        TokenType.RIGHTSQUAREBRACKET,
      TokenType.LEFTBRACKET, TokenType.TEXT, TokenType.RIGHTBRACKET }
```

The documentation for this class was generated from the following file:

- ImageParser.cs

8.12 MarkdownCompiler.IndexFileCreator Class Reference

Public Member Functions

- **IndexFileCreator** (Dictionary< string, string > fileTitles, string configFileName)
- async Task< string > GetIndexFileContents ()

  Returns the contents of the index file(header, all postst and the footer).

Properties

- Dictionary< string, string > **FileTitles** [get]
- bool **IsInsideWebDirectory** [get]
- string **ConfigFileName** [get]

Private Member Functions

- string GetHeader (ConfigContent authorAndTitle)

  Returns a string representing the header of the index page.
- IEnumerable< string > GetPosts ()

  Returns an IEnumerable of strings representing a post which should be written to the index page.
- string GetFooter (ConfigContent authorAndTitle)

  Returns the footer.

8.12.1 Member Function Documentation

8.12.1.1 GetFooter()

```
string MarkdownCompiler.IndexFileCreator.GetFooter (
            ConfigContent authorAndTitle )  [private]
```

Returns the footer.

Parameters

authorAndTitle	A class containg fields WebsiteName and Author.

Returns

A string representing the footer.

8.12.1.2 GetHeader()

```
string MarkdownCompiler.IndexFileCreator.GetHeader (
            ConfigContent authorAndTitle )  [private]
```

Returns a string representing the header of the index page.

Parameters

authorAndTitle	A class containg fields WebsiteName and Author.

Returns

A string representing the header.

8.12.1.3 GetIndexFileContents()

```
async Task< string > MarkdownCompiler.IndexFileCreator.GetIndexFileContents ( )
```

Returns the contents of the index file(header, all poststt and the footer).

Returns

A string representing the contents of the index page.

8.12.1.4 GetPosts()

```
IEnumerable< string > MarkdownCompiler.IndexFileCreator.GetPosts ( )  [private]
```

Returns an IEnumerable of strings representing a post which should be written to the index page.

Returns

IEnumerable of strings.

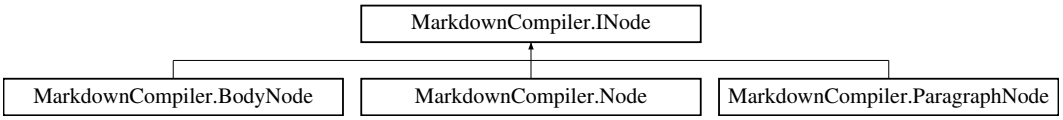
The documentation for this class was generated from the following file:

- IndexFileCreator.cs

8.13 MarkdownCompiler.INode Interface Reference

Represents a node of an Abstract Syntax Tree

Inheritance diagram for MarkdownCompiler.INode:



8.13.1 Detailed Description

Represents a node of an Abstract Syntax Tree

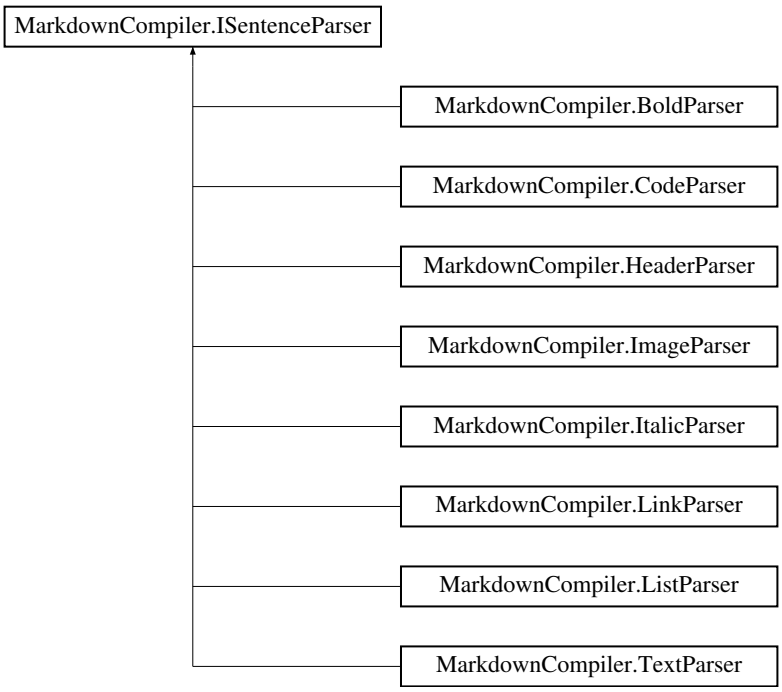
The documentation for this interface was generated from the following file:

- INode.cs

8.14 MarkdownCompiler.ISentenceParser Interface Reference

Represents a parser which looks for a specific set of tags which form a specific sentence.

Inheritance diagram for MarkdownCompiler.ISentenceParser:



Public Member Functions

- Node Parse (IToken[ ] tokens)

  Parses tokens and tries to recognize a certain pattern representing a sentence in the grammar.

8.14.1 Detailed Description

Represents a parser which looks for a specific set of tags which form a specific sentence.

8.14.2 Member Function Documentation

8.14.2.1 Parse()

```
Node MarkdownCompiler.ISentenceParser.Parse (
            IToken[] tokens )
```

Parses tokens and tries to recognize a certain pattern representing a sentence in the grammar.

Parameters

tokens	A list of tokens to be parser.

Returns

Null if no acceptable pattern was found.

Implemented in MarkdownCompiler.BoldParser, MarkdownCompiler.CodeParser, MarkdownCompiler.HeaderParser, MarkdownCompiler.ImageParser, MarkdownCompiler.ItalicParser, MarkdownCompiler.LinkParser, MarkdownCompiler.ListParser, and MarkdownCompiler.TextParser.

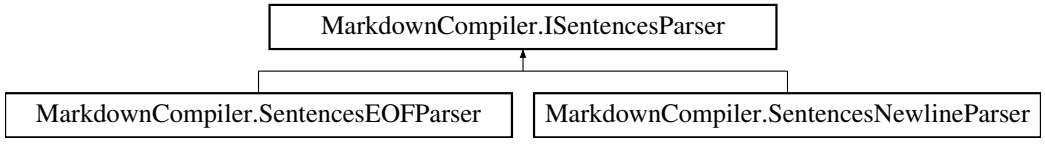
The documentation for this interface was generated from the following file:

- ISentenceParser.cs

8.15 MarkdownCompiler.ISentencesParser Interface Reference

Represents a parser which parses sentences of tokens. Sentence can be for example a bold text, normal text, link, etc.

Inheritance diagram for MarkdownCompiler.ISentencesParser:



Public Member Functions

- ParagraphNode Parse (IToken[ ] tokens)

8.15.1 Detailed Description

Represents a parser which parses sentences of tokens. Sentence can be for example a bold text, normal text, link, etc.

8.15.2 Member Function Documentation

8.15.2.1 Parse()

```
ParagraphNode MarkdownCompiler.ISentencesParser.Parse (
            IToken[] tokens )
```

Implemented in MarkdownCompiler.SentencesEOFParser, and MarkdownCompiler.SentencesNewlineParser.

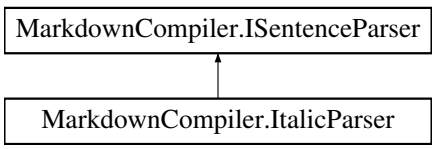
The documentation for this interface was generated from the following file:

- ISentencesParser.cs

8.16 MarkdownCompiler.ItalicParser Class Reference

A paprser specialised to recognise a token patter representing text in italic.

Inheritance diagram for MarkdownCompiler.ItalicParser:



Public Member Functions

- Node Parse (IToken[ ] tokens)

  Parses tokens and looks for a pattern representing text in italic.

Private Attributes

- List< TokenType > **UnderscoreTemplate** = new List<TokenType>() { TokenType.UNDERSCORE, Token↩ Type.TEXT, TokenType.UNDERSCORE }
- List< TokenType > **StarTemplate** = new List<TokenType>() { TokenType.STAR, TokenType.TEXT, Token↩ Type.STAR }

8.16.1 Detailed Description

A paprser specialised to recognise a token patter representing text in italic.

8.16.2 Member Function Documentation

8.16.2.1 Parse()

```
Node MarkdownCompiler.ItalicParser.Parse (
            IToken[] tokens )
```

Parses tokens and looks for a pattern representing text in italic.

Parameters

tokens	A list of tokens.

Returns

Null if no sequence was found or a node containing the text representing the text in italic.

Implements MarkdownCompiler.ISentenceParser.

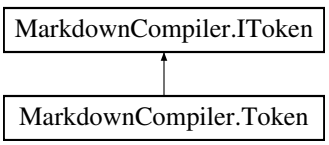
The documentation for this class was generated from the following file:

- ItalicParser.cs

8.17 MarkdownCompiler.IToken Interface Reference

An implementation must represent a token with a type and a value created by the Tokenizer while reading the correct char from the file.

Inheritance diagram for MarkdownCompiler.IToken:



Properties

- TokenType **Type** `[get]`
- string **Value** `[get]`

8.17.1 Detailed Description

An implementation must represent a token with a type and a value created by the Tokenizer while reading the correct char from the file.

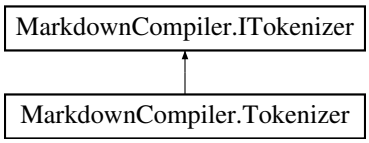
The documentation for this interface was generated from the following file:

- IToken.cs

8.18 MarkdownCompiler.ITokenizer Interface Reference

Must represent a Tokenizer. Must be accompanied by a creation of a specific parser.

Inheritance diagram for MarkdownCompiler.ITokenizer:



Public Member Functions

- List< IToken > Tokenize (string text)

8.18.1 Detailed Description

Must represent a Tokenizer. Must be accompanied by a creation of a specific parser.

8.18.2 Member Function Documentation

8.18.2.1 Tokenize()

```
List< IToken > MarkdownCompiler.ITokenizer.Tokenize (
            string text )
```

Implemented in MarkdownCompiler.Tokenizer.

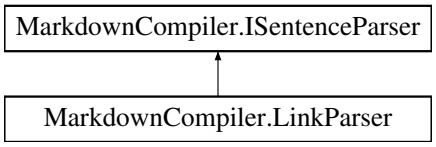
The documentation for this interface was generated from the following file:

- ITokenizer.cs

8.19 MarkdownCompiler.LinkParser Class Reference

A paprser specialised to recognise a token patter representing a link.

Inheritance diagram for MarkdownCompiler.LinkParser:



Public Member Functions

- Node Parse (IToken[ ] tokens)

  Parses tokens and looks for a pattern representing a link.

Private Attributes

- List< TokenType > **LinkNoNameTemplate** = new List<TokenType>() { TokenType.LEFTSQUAREBRACKET, TokenType.RIGHTSQUAREBRACKET, TokenType.LEFTBRACKET, TokenType.TEXT, TokenType.↩ RIGHTBRACKET }
- List< TokenType > NamedLinkTemplate

8.19.1 Detailed Description

A paprser specialised to recognise a token patter representing a link.

8.19.2 Member Function Documentation

8.19.2.1 Parse()

```
Node MarkdownCompiler.LinkParser.Parse (
            IToken[] tokens )
```

Parses tokens and looks for a pattern representing a link.

Parameters

tokens	A list of tokens.

Returns

Null if no sequence was found or a node containing the text representing the link and an optional name.

Implements MarkdownCompiler.ISentenceParser.

8.19.3 Member Data Documentation

8.19.3.1 NamedLinkTemplate

```
List<TokenType> MarkdownCompiler.LinkParser.NamedLinkTemplate  [private]
```

Initial value:

```
= new List<TokenType>() { TokenType.LEFTSQUAREBRACKET, TokenType.TEXT,
                                                        TokenType.RIGHTSQUAREBRACKET,
      TokenType.LEFTBRACKET, TokenType.TEXT, TokenType.RIGHTBRACKET }
```

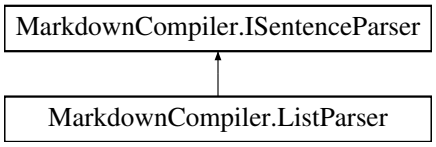
The documentation for this class was generated from the following file:

- LinkParser.cs

8.20 MarkdownCompiler.ListParser Class Reference

A paprser specialised to recognise a token patter representing a unordered list.

Inheritance diagram for MarkdownCompiler.ListParser:



Public Member Functions

- Node Parse (IToken[ ] tokens)

  Parses tokens and looks for a pattern representing a member of an unordered list.

8.20.1 Detailed Description

A paprser specialised to recognise a token patter representing a unordered list.

8.20.2 Member Function Documentation

8.20.2.1 Parse()

```
Node MarkdownCompiler.ListParser.Parse (
            IToken[] tokens )
```

Parses tokens and looks for a pattern representing a member of an unordered list.

Parameters

tokens	A list of tokens.

Returns

Null if no sequence was found or a node containing the text representing a member of an unordered list.

Implements MarkdownCompiler.ISentenceParser.

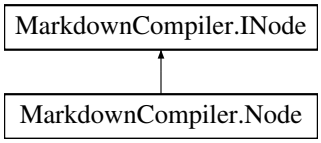
The documentation for this class was generated from the following file:

- ListParser.cs

8.21 MarkdownCompiler.Node Class Reference

A node representing a sentence in the grammar.

Inheritance diagram for MarkdownCompiler.Node:



Static Public Member Functions

- static Node **NodeFactory** (NodeType type, string value, int consumed)
- static Node **NamedNodeFactory** (NodeType type, string value, int consumed, string name)

Properties

- NodeType **Type** `[get]`
- string **Value** `[get]`
- int **Consumed** `[get]`
- string **Name** `[get, set]`

Private Member Functions

- **Node** (NodeType type, string value, int consumed)
- **Node** (NodeType type, string value, int consumed, string name)

8.21.1 Detailed Description

A node representing a sentence in the grammar.

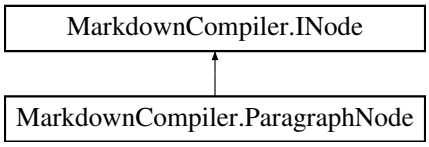
The documentation for this class was generated from the following file:

- Node.cs

8.22 MarkdownCompiler.ParagraphNode Class Reference

A node representing a paragraph. Contains all sentences inside of the paragraph.

Inheritance diagram for MarkdownCompiler.ParagraphNode:



Static Public Member Functions

- static ParagraphNode **ParagraphNodeFactory** (List< Node > sentences, int consumed)

Properties

- List< Node > **Sentences** `[get]`
- int **Consumed** `[get]`

Private Member Functions

- **ParagraphNode** (List< Node > sentences, int consumed)

8.22.1 Detailed Description

A node representing a paragraph. Contains all sentences inside of the paragraph.

The documentation for this class was generated from the following file:

- ParagraphNode.cs

8.23 MarkdownCompiler.ParagraphParser Class Reference

A paprser specialised to recognise a paragraph node from the list of tokens.

Public Member Functions

- ParagraphNode Parse (IToken[ ] tokens)

  Parses the given tokens. Tries to find a paragraph in the tokens.

Private Member Functions

- ParagraphNode MatchOneParagraph (IToken[ ] tokens, ISentencesParser[ ] parsers)

  Matches one paragraph from the tokens. Tries different sentence parsers until a pararaph is found.

8.23.1 Detailed Description

A paprser specialised to recognise a paragraph node from the list of tokens.

8.23.2 Member Function Documentation

8.23.2.1 MatchOneParagraph()

```
ParagraphNode MarkdownCompiler.ParagraphParser.MatchOneParagraph (
            IToken[] tokens,
            ISentencesParser[] parsers )  [private]
```

Matches one paragraph from the tokens. Tries different sentence parsers until a pararaph is found.

Parameters

tokens	A list of tokens.
parsers	Parser which parse the tokens until a match is found.

Returns

Null if no match was found or a ParagraphNode.

8.23.2.2 Parse()

```
ParagraphNode MarkdownCompiler.ParagraphParser.Parse (
            IToken[] tokens )
```

Parses the given tokens. Tries to find a paragraph in the tokens.

Parameters

tokens	A list of tokens to be parsed.

Returns

Null if no sequence was found or a ParagraphNode containing sentences.

The documentation for this class was generated from the following file:

- ParagraphParser.cs

8.24 MarkdownCompiler.Parser Class Reference

Static Public Member Functions

- static IEnumerable< ParagraphNode > **GetAllParagraphs** (IToken[ ] tokens)

The documentation for this class was generated from the following file:

- Parser.cs

8.25 StaticSiteGenerator.Program Class Reference

Class containing the Main method.

Static Private Member Functions

- static async Task Main (string[ ] args)

  The main entrypoint of the program. If no arguments are given to the program the generator will create the correct directories and a config file. If the argument "compile" is given, the generator will create the output directory compile all markdown files into HTML.

8.25.1 Detailed Description

Class containing the Main method.

8.25.2 Member Function Documentation

8.25.2.1 Main()

```
static async Task StaticSiteGenerator.Program.Main (
            string[] args )  [static], [private]
```

The main entrypoint of the program. If no arguments are given to the program the generator will create the correct directories and a config file. If the argument "compile" is given, the generator will create the output directory compile all markdown files into HTML.

Parameters

args	

The documentation for this class was generated from the following file:

- Program.cs

8.26 MarkdownCompiler.SentenceParser Class Reference

A paprser specialised to recognise a sentence node from the list of tokens.

Public Member Functions

- Node Parse (IToken[ ] tokens)
  Parses the given tokens. Tries to find a sentence in the tokens.

Static Public Member Functions

- static bool CheckTypes (IToken[ ] tokens, List< TokenType > template)
  Tries to find a pattern corresponding to a template pattern.

Private Member Functions

- Node MatchOneSentence (IToken[ ] tokens, ISentenceParser[ ] parsers)
  Matches one sentence from the tokens. Tries different sentence parsers until a sentence is found.

Private Attributes

- ISentenceParser[ ] SentenceParsers

8.26.1 Detailed Description

A paprser specialised to recognise a sentence node from the list of tokens.

8.26.2 Member Function Documentation

8.26.2.1 CheckTypes()

```
static bool MarkdownCompiler.SentenceParser.CheckTypes (
            IToken[ ] tokens,
            List< TokenType > template )  [static]
```

Tries to find a pattern corresponding to a template pattern.

Parameters

tokens	Tokens to be checked.
template	A template list.

Returns

False if the two lists differ or true if a match is found.

8.26.2.2 MatchOneSentence()

```
Node MarkdownCompiler.SentenceParser.MatchOneSentence (
            IToken[] tokens,
            ISentenceParser[] parsers )  [private]
```

Matches one sentence from the tokens. Tries different sentence parsers until a sentence is found.

Parameters

tokens	A list of tokens.
parsers	Parser which parse the tokens until a match is found./param>
	Returns
	Null if no match was found or a Node.

8.26.2.3 Parse()

```
Node MarkdownCompiler.SentenceParser.Parse (
            IToken[] tokens )
```

Parses the given tokens. Tries to find a sentence in the tokens.

Parameters

tokens	A list of tokens to be parsed.

Returns

Null if no sequence was found or a Node representing sentence.

8.26.3 Member Data Documentation

8.26.3.1 SentenceParsers

```
ISentenceParser [] MarkdownCompiler.SentenceParser.SentenceParsers  [private]
```

Initial value:

```
= new ISentenceParser[] { new ImageParser(), new LinkParser(), new ItalicParser(), new BoldParser(),
            new CodeParser(), new ListParser(), new HeaderParser(), new TextParser() }
```

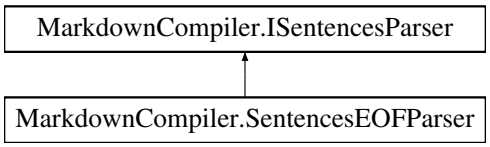
The documentation for this class was generated from the following file:

- SentenceParser.cs

8.27 MarkdownCompiler.SentencesEOFParser Class Reference

A paprser specialised to recognise a sentences followed by the End of File character.

Inheritance diagram for MarkdownCompiler.SentencesEOFParser:



Public Member Functions

- ParagraphNode Parse (IToken[ ] tokens)

  Parses the given tokens. Tries to find all sentences which are part of a paragraph in the tokens.

Private Member Functions

- void MatchAllSentences (IToken[ ] tokens, SentenceParser sentenceParser, out List< Node > sentences, out int consumed)

  Tries to find all sentences which make up a paragraph.

8.27.1 Detailed Description

A paprser specialised to recognise a sentences followed by the End of File character.

8.27.2 Member Function Documentation

8.27.2.1 MatchAllSentences()

```
void MarkdownCompiler.SentencesEOFParser.MatchAllSentences (
            IToken[ ] tokens,
            SentenceParser sentenceParser,
            out List< Node > sentences,
            out int consumed )  [private]
```

Tries to find all sentences which make up a paragraph.

Parameters

tokens	A list of tokens.
sentenceParser	A sentence parser.
sentences	Found sentences.
consumed	How many tokens have been consumed.

8.27.2.2 Parse()

```
ParagraphNode MarkdownCompiler.SentencesEOFParser.Parse (
            IToken[] tokens )
```

Parses the given tokens. Tries to find all sentences which are part of a paragraph in the tokens.

Parameters

tokens	A list of tokens.

Returns

Null if no paragraph was found or a ParagraphNode.

Implements MarkdownCompiler.ISentencesParser.

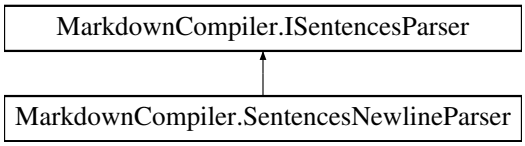
The documentation for this class was generated from the following file:

- SentencesEOFParser.cs

8.28 MarkdownCompiler.SentencesNewlineParser Class Reference

A paprser specialised to recognise a sentences followed by an empty line.

Inheritance diagram for MarkdownCompiler.SentencesNewlineParser:



Public Member Functions

- ParagraphNode Parse (IToken[ ] tokens)

  Parses the given tokens. Tries to find all sentences which are part of a paragraph in the tokens followed by an empty line.

Private Member Functions

- void MatchAllSentences (IToken[ ] tokens, SentenceParser sentenceParser, out List< Node > sentences, out int consumed)

  Tries to find all sentences which make up a paragraph.

8.28.1 Detailed Description

A paprser specialised to recognise a sentences followed by an empty line.

8.28.2 Member Function Documentation

8.28.2.1 MatchAllSentences()

```
void MarkdownCompiler.SentencesNewlineParser.MatchAllSentences (
            IToken[] tokens,
            SentenceParser sentenceParser,
            out List< Node > sentences,
            out int consumed )  [private]
```

Tries to find all sentences which make up a paragraph.

Parameters

tokens	A list of tokens.
sentenceParser	A sentence parser.
sentences	Found sentences.
consumed	How many tokens have been consumed.

8.28.2.2 Parse()

```
ParagraphNode MarkdownCompiler.SentencesNewlineParser.Parse (
            IToken[] tokens )
```

Parses the given tokens. Tries to find all sentences which are part of a paragraph in the tokens followed by an empty line.

Parameters

tokens	A list of tokens.

Returns

Null if no paragraph was found or a ParagraphNode.

Implements MarkdownCompiler.ISentencesParser.

The documentation for this class was generated from the following file:

- SentencesNewlineParser.cs

8.29 MarkdownCompiler.SpecialCharScanner Class Reference

Static Public Member Functions

- static IToken ScanChar (char character)

  Checks if a character has been assigned a specific tag and returns one if it does.

8.29.1 Member Function Documentation

8.29.1.1 ScanChar()

```
static IToken MarkdownCompiler.SpecialCharScanner.ScanChar (
            char character )  [static]
```

Checks if a character has been assigned a specific tag and returns one if it does.

Parameters

text	Markdown text

Returns

Will return a tag of a special type. Will return null if no special tag is assigned.

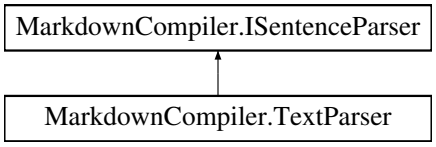
The documentation for this class was generated from the following file:

- Tokenizer.cs

8.30 MarkdownCompiler.TextParser Class Reference

A paprser specialised to recognise a token patter representing plain text.

Inheritance diagram for MarkdownCompiler.TextParser:



Public Member Functions

- Node Parse (IToken[ ] tokens)

  Parses tokens and looks for a pattern representing plain text.

8.30.1 Detailed Description

A paprser specialised to recognise a token patter representing plain text.

8.30.2 Member Function Documentation

8.30.2.1 Parse()

```
Node MarkdownCompiler.TextParser.Parse (
            IToken[] tokens )
```

Parses tokens and looks for a pattern representing plain text.

Parameters

tokens	A list of tokens.

Returns

Null if no sequence was found or a node containing the text.

Implements MarkdownCompiler.ISentenceParser.

The documentation for this class was generated from the following file:

- TextParser.cs

8.31 MarkdownCompiler.TextScanner Class Reference

Static Public Member Functions

- static IToken ScanChar (string text)

  Consumes text until it finds a character with a specific assigned tag. Creates a new TEXT token containg the found text.

8.31.1 Member Function Documentation

8.31.1.1 ScanChar()

```
static IToken MarkdownCompiler.TextScanner.ScanChar (
            string text )  [static]
```

Consumes text until it finds a character with a specific assigned tag. Creates a new TEXT token containg the found text.

Parameters

text	Markdown text

Returns

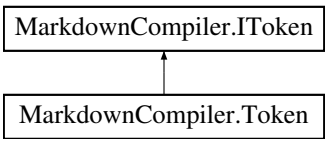
A token of the type TEXT containg the found text.

The documentation for this class was generated from the following file:

- Tokenizer.cs

8.32 MarkdownCompiler.Token Class Reference

Inheritance diagram for MarkdownCompiler.Token:



Static Public Member Functions

- static Token **TokenFactory** (TokenType Type, string Value)

Properties

- TokenType Type `[get]`
- string Value `[get]`

Private Member Functions

- **Token** (TokenType type, string value)

8.32.1 Property Documentation

8.32.1.1 Type

```
TokenType MarkdownCompiler.Token.Type  [get]
```

Implements MarkdownCompiler.IToken.

8.32.1.2 Value

```
string MarkdownCompiler.Token.Value  [get]
```

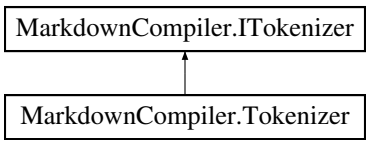
Implements MarkdownCompiler.IToken.

The documentation for this class was generated from the following file:

- BaseToken.cs

8.33 MarkdownCompiler.Tokenizer Class Reference

Inheritance diagram for MarkdownCompiler.Tokenizer:



Public Member Functions

- List< IToken > Tokenize (string text)

  Tokenizes the markdown text. See the enum TokenType to see all the different possible tags.

Private Member Functions

- List< IToken > TokensToList (string text, List< IToken > tokens)

  Takes the markdown text and turns it into a list of tokens.
- IToken CreateToken (string text)

  Create a token from a singe character or a text token from multiple characters.

8.33.1 Member Function Documentation

8.33.1.1 CreateToken()

```
IToken MarkdownCompiler.Tokenizer.CreateToken (
            string text )  [private]
```

Create a token from a singe character or a text token from multiple characters.

Parameters

text	Markdown text.

Returns

A single token.

8.33.1.2 Tokenize()

```
List< IToken > MarkdownCompiler.Tokenizer.Tokenize (
            string text )
```

Tokenizes the markdown text. See the enum TokenType to see all the different possible tags.

Parameters

text	A markdown text set to be Tokenized.

Returns

A List of tokens containing the found tokens. Will return null if the input text was null.

Implements MarkdownCompiler.ITokenizer.

8.33.1.3 TokensToList()

```
List< IToken > MarkdownCompiler.Tokenizer.TokensToList (
            string text,
            List< IToken > tokens )  [private]
```

Takes the markdown text and turns it into a list of tokens.

Parameters

text	A markdown text to be tokenized.
tokens	An(empty) array of tokens.

Returns

A list of tokens.

The documentation for this class was generated from the following file:

- Tokenizer.cs

8.34 StaticSiteGeneratorTests.TokenizerTests Class Reference

Public Member Functions

- void **TokenizeTextNotNUll** ()
- void **TokenizeTextNoText** ()
- void **TokenizeTextCountMoreThanZero** ()
- void **TokenizeNullString** ()

The documentation for this class was generated from the following file:

- TokenizerTests.cs

Index