차례

GitBook 도움말

GitBook 공식 도움말을 보며 개인적으로 사용법을 익히다가 혹시 다른 분들께도 도움이 될까 싶어 공개합니다.

마음먹고 한 번역은 아니기에 오타, 오역이 많으며 문체도 거칠고 일관되지 않아 부끄럽지만 GitBook을 처음 활용하실 때 도움이 되었으면 하는 바람입니다.

오역으로 인해 잘못된 사용법이 기제되어 있다면 bluekms21@naver.com이나 블로그로 제보 부탁드립니다.

Format

포멧의 주안점은 간단함과 가독성이다.

깃북은 사용한다 마크업 파일 상단에 규칙을 사용한다:

- README: 책을 소개한다.
- SUMMARY: 쳅터 구조를 정의한다.
- LANGS: 다중 언어 책을 지원한다.
- GLOSSARY: 용어와 설명 목록

책은 `READMY` 와 `SUMMARY` 를 필요로 한다.

출력 형식

깃북은 각기 다른 e북 포멧으로 당신의 책을 생성할 수 있습니다.

고정된 웹사이트

이것은 일반적인 형태이다. 완전한 대화형 고정된 웹사이트로 생성된다.

PDF (Portable Document Format)

PDF는 파일형태로 많은 하드웨어, 소프트웨어, OS에서 제공되는 매우 일반적인 형태이다.

파일의 확장자는 `.pdf` 이다.

ePub (전자 출판)

EPUB (전자 출판; 때때로 ePub)은 자유롭게 사용 가능한 오픈 e북 포멧이다.

또한 국제 전자 출판 포럼(IDPF; International Digital Publishing Forum)의 표준 양식이다.

파일의 확장자는 `.epub` 이며 ePubs은 애플과 구글 드라이브를 지원한다.

Mobi (Mobipocket)

모비포켓은 e북 포멧이다.

XHTML을 사용한 오픈 e북 표준에 기반하고 있으며 자바스크립트와 프레임들을 포함할 수 있다.

이것은 아마존의 kindle기기를 지원한다.

Readme로 책 소개

네 책의 첫번째 페이지는 README.md 이다. 만약 요약 파일이 제공되지 않는다면 이 파일은 첫번째 엔트리로 추가된다.

다른 파일을 README.md로 사용하기

깃허브에서 호스팅되는 일부 책들의 소개 페이지 대신 README.md를 유지하라.

깃허브 버전 2.0.0 이상부터 사용하던 README를 `book.json` 으로 정의하는것이 가능해졌다.

```
{
    "structure": {
        "readme": "myIntro.md"
    }
}
```

챕터와 서브챕터

깃북은 SUMMARY.md 파일을 사용하여 책에 쓰일 챕터와 서브챕터를 정의할 수 있다.

SUMMARY.md 파일은 일반적으로 책의 컨텐츠 테이블을 만드는데 사용된다.

SUMMARY.md 의 포멧은 간단한 링크들의 목록이다.
링크의 이름은 챕터 이름으로 사용된다. 그리고 타겟은 챕터파일의 경로이다.

서브챕터는 부모 챕터에 속한 리스트 형태로 간단히 정의할 수 있다.

간단한 예제

```
# Summary

* [Chapter 1](chapter1.md)
* [Chapter 2](chapter2.md)
* [Chapter 3](chapter3.md)
```

서브챕터 예제

```
# Summary

* [Part I](part1/README.md)
    * [Writing is nice](part1/writing.md)
    * [GitBook is nice](part1/gitbook.md)
* [Part II](part2/README.md)
    * [We love feedback](part2/feedback_please.md)
    * [Better tools for authors](part2/better_tools.md)
```

역주

챕터와 서브챕터에는 넘버링을 하지 않아도 자동으로 넘버링 된다.

```
[제목](파일링크) 로 작성하면 된다.

* [형식](1. 형식.md)
  * [출력](1.1. 출력.md)
  * [Readme로 책 소개](1.2. Readme로 책 소개.md)

위와 같이 작성하면 실제 표시되는 챕터에는

1. 형식
  1.1. 출력
  1.2. Readme로 책 소개

로 표시된다.
```

마크다운

깃북은 마크다운 문법을 기본으로 하고있다. 이것은 빠른 참조와 쇼케이스를 위해 의도되었다. 보다 완벽한 정보를 원한다면 John Gruber의 오리지널 사양서와 깃허브-플레이버 마크다운 정보 페이지를 참조해라.

헤더들

```
# H1
## H2
### H3
#### H4
##### H5
###### H6
```

또한 H1과 H2제목에는 밑줄을 줄 수 있다. (밑줄을 사용하면 H1과 H2 스타일로 적용된다.)

```
Alt-H1
=====

Alt-H2
-----
```

단락과 줄 바꿈

단락은 단순히 한 줄 이상의 공백 라인에 의해 분리된다. (공백 라인은 스페이스바나 텝으로만 이루어진 행을 뜻합니다.) 일반 단락은 공백이나 탭으로 들여쓰기를 하지 않아야 한다.

```
이곳은 우리가 시작하는 줄이다.

이 줄은 위의 공백 라인에 의해 *별도의 단락*이 될 것이다.

이 줄도 별도의 단락이다. 하지만...
이 줄은 공백 라인 없는 줄 바꿈으로 구분되어 *같은 단락*의 별도의 라인이 된다.
```

강조

```
강조는 이텔릭체 이며 *별표* 또는 _언더바_에 의해 표현된다.

강한 강조는 볼드체이며 **쌍별** 또는 __쌍언더바__에 의해 표현된다.

결합된 강조는 **별표 안에 _언더바_**로 표현한다.

취소선은 ~~두 물결표시~~로 표현한다.
```

리스트

(이 예에서는 선행과 후행의 빈 공간을 점으로 표시한다)

```
1. 첫 번째 리스트 아이템
2. 다른 아이템
  * 순서 없는 서브 리스트
1. 숫자의 순서는 중요하지 않는다. 그냥 숫자로 시작한다는것이 중요하다.
  1. 순서 있는 서브 리스트
4. 그리고 다른 아이템

   너는 정확히 단락의 아이템들을 들여쓰기 할 수 있다. 위의 빈 줄과 선행 빈 칸에 주목하라. (원시 마크다운을 정렬하기 위해서는 다음 3가지 방법 중 하나를 사용해라.)

   단락 없이 줄 바꿈을 하려면 줄 뒤에 빈칸 두개를 사용해야 한다.
   이 줄은 분리되어 있지만 같은 문단이다.
   (이것은 후행 공백이 필요하지 않다.)

* 정렬없는 리스트는 별로 시작할 수 있다.
- 또는 마이너스
+ 또는 플러스
```

링크

```
다음 두 가지 방법으로 링크를 만들 수 있다.

[인라인 링크](https://www.google.com)

[제목있는 인라인 링크](https://www.google.com "Google's Homepage")

[참조 링크][대소문자를 구분하는 임의 참조 텍스트]

[저장소 파일의 상대경로 링크](../blob/master/LICENSE)

[정의된 숫자를 사용한 참조 링크][1]

링크 텍스트 자체를 링크로 사용할 수 있다.

다음은 참조 링크가 나중에 올 수 있음을 보여준다.

[대소문자를 구분하는 임의 참조 텍스트]: https://www.mozilla.org

[1]: http://slashdot.org

[링크 텍스트 자체]: http://www.reddit.com
```

각주

마크다운에서 사용되는 기본 각주 스타일 링크는 페이지에 표시되지 않는다. 링크없이 포함된 각주는 리더에서 볼 때 유용하다. 깃북은 간단한 문법으로 각주를 찾을 수 있도록 제공한다.

```
선 텍스트 후 각주 [^2]
[^2] 각주의 상세 내용
```

이미지

```
인라인 스타일:
![alt text](https://github.com/adam-p/markdown-here/raw/master/src/common/images/icon48.png "Logo Title Text 1")

참조 스타일:
![art text][logo]

[logo]: https://github.com/adam-p/markdown-here/raw/master/src/common/images/icon48.png "Logo Title Text 1"
```

코드와 문법 강조

코드블록은 마크다운 스팩의 한 부분이다, 그러나 문법 강조는 아니다. 그러나 깃허브와 마크다운과 같은 많은 렌더러들은 문법 강조를 지원한다. 몇몇 언어는 렌더러에 맞게 다르게 작성되어야 한다. 마크다운은 수십개 언어에 대한 강조를 지원한다.(HTTP와 같은 현실의 언어 이외에도) 다음 예제에는 어떻게 언어의 이름들을 써야 하는지 보여준다. 하이라이트.js 데모 페이지

```
인라인 코드는 `역따옴표`로 그것을 감싼다.
(역자: 역따옴표는 키보드 1키의 왼쪽 ~와 함께 있다.)
```

코드블록들은 세개의 역따옴표 ````` 라인으로 구분하거나 4칸 들여쓰기로 구분된다. 나는 3개의 역따옴표로 구분하는것을 추천한다. 그것은 쉽고 문법강조를 지원받을 수 있다.

```
```javascript
var s = "JavaScript syntax highlighting";
alert(s);
```
```

```
```python
s = "파이썬 문법 강조"
print s
```
```

```
언어가 아닐경우 문법강조를 지원하지 않는다.
그러나 이것은 <b>tag</b> 를 발생시킨다.
```

테이블

테이블은 마크다운 스팩에 포함되지 않는다. 그러나 그들은 GFM의 부분이며 마크다운은 그것을 지원한다.

콜론으로 줄 컬럼을 사용할 수 있다

테이블	은	좋다
3번째 열은	왼쪽정렬이다	$1600
2번째 열은	중앙정렬이다	$12
얼룩말 줄무늬는	산뜻하다	$1

밖의 파이프(|)는 필수는 아니다. 그리고 너는 반드시 행을 정렬해 둘 필요가 없다. 너는 또한 인라인 마크다운을 쓸 수 있따.

마크다운	정렬	없이
여전히	`표현된다`	**나이스**
1	2	3

인용블록

> 인용블록은 이메일에서 회신 텍스트를 표현할때 유용합니다. 이 라인은 같은 인용문으로 표시됩니다.

인용블록

> 이것은 인용블록 안에 들어있는 매우 긴 문자열이다. 동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리 나라 만세. 오 너는 인용블록 안에서 **마크다운**을 사용 가능하다.

인라인 HTML

너는 또한 원시 HTML을 네 마크다운에서 사용할 수 있다. 그리고 이것은 대부분 꽤 잘 작동한다.

Definition list
Is something people use sometimes.

Markdown in HTML
Does *not* work **very** well. Use HTML *tags*.

수평선

3개 이상의 기호들로 표현 가능하다.

```
---

하이픈들

***

별모양들

___

언더바들
```

아스키 문서

`2.0.0` 버전부터 깃북은 또한 아스키 문서를 입력 형식으로 사용할 수 있다.

아스키 문서 형식의 자세한 정보는 아스키 문서 빠른 참조에서 볼 수 있다.

마크다운과 깃북은 일부 특수한 파일로 구조를 추출할 수 있다. `README.adoc` , `SUMMARY.adoc` , `LANGS.adoc` , 그리고 `GLOSSARY.adoc`

README.adoc

이것은 책을 소개하는 메인 엔트리이다. 이 파일은 필수이다.

SUMMARY.adoc

이 파일은 마크다운과 유사하게 챕터와 서브챕터들을 정의한다. `SUMMARY.adoc` 의 형태는 간단한 링크들의 리스트이다. 링크의 이름은 챕터의 이름으로 사용되며 타겟 경로는 챕터의 파일이다.

서브챕터는 부모 챕터에 포함된 리스트로 간단히 정의된다.

```
= Summary

. link:chapter-1/README.adoc[Chapter 1]
.. link:chapter-1/ARTICLE1.adoc[Article 1]
.. link:chapter-1/ARTICLE2.adoc[Article 2]
... link:chapter-1/ARTICLE-1-2-1.adoc[Article 1.2.1]
. link:chapter-2/README.adoc[Chapter 2]
. link:chapter-3/README.adoc[Chapter 3]
. link:chapter-4/README.adoc[Chapter 4]
.. Unfinished article
. Unfinished Chapter
```

LANGS.adoc

다중 언어 책들의 파일은 별도로 정의된 언어 파일들을 사용한다.

이 파일들은 `SUMMARY.adoc` 의 문법을 따른다.

```
= Languages

. link:en/[English]
. link:fr/[French]
```

GLOSSARY.adoc

이 파일은 용어집이다. 자세한 사항은 용어집항목 참조.

```
= Glossary

== Magic
Sufficiently advanced technology, beyond the understanding of the observer producing a
 sense of wonder.

== PHP
An atrocious language, invented for the sole purpose of inflicting pain and suffering
amongst the programming wizards of this world.
```

표지

깃북을 더욱 우아하게 꾸미기 위해 특별한 표지를 만들 수 있다.

표지는 *cover.jpg* 파일과 *cover_small.jpg* 파일이며 반드시 JPEG 파일이어야 한다.

최적 크기

	큰 사이즈	작은 사이즈
File	`cover.jpg`	`cover_small.jpg`
픽셀	1800x2360	200x262

가이드라인

좋은 표지는 다음과 같은 지침을 따른다.

- 테두리 없음
- 눈에 띄는 제목
- 중요한 텍스트는 작은 이미지에서도 뚜렷히 보여야 함

자동 표지

깃북 플러그인(`autocover`)를 사용하여 자동 커버파일을 생성할 수 있다. 또는 `cover_small.jpg` 파일로부터 큰 표지를 만들 수 있다. 이 플러그인은 호스트 책에 기본으로 추가된다.

자동 표지에 대한 추가 정보

다중 언어

깃북은 다양한 언어로 된 책들을 만들 수 있다.

각 언어는 깃북의 포멧을 따르는 하위 디렉토리에 위치해야 하며 파일명은 LANGS.md여야 한다.

```
* [English](en/)
* [French](fr/)
* [Espanol](es/)
```

예제는 Learn Git 에서 볼 수 있다.

용어집

용어와 그 용어의 뜻을 지정할 수 있다.

이러한 용어들을 바탕으로 깃북은 자동으로 인덱스를 생성하고 페이지에서 강조되어 보이도록 한다.

GLOSSARY.md 포멧은 매우 간단하다.

```
# 용어
이 용어의 뜻

# 다른 용어
이 용어의 뜻. 이것은 강조되어 표시된다.
```

템플릿

깃북은 Nunjucks와 Jinja2의 문법을 사용한다.

이스케이프

어떤 특수한 템플릿 테그를 출력하고 싶다면 raw를 사용하여 일반 텍스트로 출력할 수 있다. 이것은 어떤 것이든 표현할 수 있다.

```
{% raw %}
    *이것은* 일반 텍스트로 출력된다. {{ 처리되지 않는다 }}
{% endraw %}
```

변수

변수는 책의 문맥에서 값을 참조한다. 변수는 `book.json` 파일에 정의되어 있다.

```
{
    "variables": {
        "myVeriable": "Hello World"
    }
}
```

변수 출력

`book.json` 에 정의된 변수들은 다음과 같은 방법으로 접근할 수 있다.

```
{{ book.myVariable }}
```

book에 있는 변수인 myVariable을 보여준다. 변수 이름은 프로퍼티들을 가질 수 있다. 또한 대괄호를 사용하여 표현할 수 있다.

```
{{ book.foo.bar }}
{{ book["bar"] }}
```

만약 변수가 정의되어 있지 않다면 아무것도 아닌 취급을 받는다. 만약 `foo` 가 정의되어 있지 않다면 `{{ foo }}, {{ foo.bar }}, {{foo.bar.baz }}` 로 보인다.

> 역주: 단, 문서 작성중 미리보기에는 보이지 않는다. 또한 반영되는데 어느정도 시간이 필요한 것도 같다. 2015-10-21 23:52

문맥 변수

일부 변수들은 현재 파일이나 깃북 인스턴스의 정보를 얻는것이 가능하다.

이름	설명
file.path	책 파일의 상대 경로
file.mtime	파일이 마지막으로 수정된 시간

테그

테그는 템플릿 부분에 대한 작업을 수행하기 위한 특수 블록이다.

If

if는 조건을 테스트하고 문맥을 선택적으로 표시한다. 이것은 프로그래밍 언어와 유사하다.

```
{% if 변수명 %}
    참
{% endif %}
```

만약 변수가 정의되었거나 값이 참인경우 "참"이 출력된다. 아닌경우 아무것도 표시되지 않는다. 또한 선택적으로 elif를 사용할 수 있다.

```
{% if book.hungry %}
    배고프다
{% elif book.tired %}
    피곤하다
{% else %}
    좋다!
{% endif %}
```

for

for 문은 array나 dictionary들을 순회할때 사용한다. book.json에 다음과 같은 배열변수를 선언하고 순회해보자.

```
{
    "variables": {
        "authors": [
            { "name": "Samy" },
            { "name": "Aaron" }
        ]
    }
}
```

```
# 저자들

{% for author in book.authors %}
    - {{ author.name }}
{% endfor %}
```

이 예제는 모든 `authors`를 순회하면서 `name` 속성을 출력한다.

include

include에 대한 자세한 정보는 콘텐츠 참조에 나와있다.

```
변수연습
```

`변수출력`

`참1`

`참2`

```
피곤하다
```

`else사용`

```
- {{ author.name }}

- {{ author.name }}
```

`for문 사용`

`배열의 일부만 접근`

1.9. 탬플릿.md `파일경로`

Thu Mar 16 2017 00:25:37 GMT+0000 (UTC) `파일수정시간`

`테스트 종료`

콘텐츠 참조

콘텐츠 참조(conref)는 다른 파일이나 책들에서 콘텐츠를 재사용하기 편리한 메커니즘이다.

현재 책의 다른 파일에 있는 컨텐츠 참조

다른 파일에 있는 콘텐츠를 `include` 테그를 이용하여 정말 쉽게 참조할 수 있다.

```
{% include "./test.md %}
```

다른 책의 파일에 있는 컨텐츠 참조

깃북은 또한 git테그를 사용하여 다른 책의 파일에 있는 컨텐츠를 참조할 수 있다.

```
{% include "git+https://github.com/GitbookIO/documentation.git/README.md#0.0.1" %}
```

git url의 형태는 다음과 같다.

```
git+https://user@hostname/project/blah.git/file#commit-ish
```

실제 git url은 반드시 `.git` 로 끝나야 한다. 그리고 가져올 파일이름은 `.git` 뒤에 나와야 한다.

`commit-ish` 및 어떠한 테그, sha, 또는 브랜치를 지정할 수 있다. `git checkout` 인자값으로. 그리고 기본은 `master` 이다.

상속

템플릿 상속은 재사용 탬플릿을 쉽게 만드는 방법이다. 템플릿을 쓸 때 너는 "blocks"를 정의 할 수 있고, 자식 탬플릿은 이것을 오버라이드 할 수 있다. 상속은 원하는 만큼 가능하다.

`block` 은 템플릿의 구역과 식별자를 정의한다. 식별자는 이름이다. 부모 템플릿은 특수한 블록들이다. 그리고 자식 템플릿들은 새로운 컨탠츠로 오버라이드 할 수 있다.

```
{% extends "./mypage.me" %}

{% block pageContent %}
# 이것은 내 페이지 콘텐츠이다
{% endblock %}
```

`mypage.md` 파일에는 반드시 블록을 정의해야 한다.

```
{% block pageContent %}
이것은 기본 콘텐츠이다
{% endblock %}

# License

{% import "./LICENSE" %}
```

파일 & 폴더 무시

깃북은 `.gitignore`, `.bookignore` 그리고 `.ignore` 파일들로 파일이나 폴더를 무시하고 넘어갈 수 있다.

이러한 파일들의 형태는 `.gitignore` 와 같다.

```
# 코맨트(주석)

# test.md 파일을 무시
test.md

# bin 디렉토리 안의 모든 파일을 무시
bin/*
```

설정

모든 설정들은 `book.json` 에 저장된다.

jsonlint.com에서 문법을 검사한 뒤 너의 `book.json` 에 붙여넣기 할 수 있다.

모든 필드는 일부 추출된 값으로 옵션이거나 기본이다.

필드

gitbook

```
{ "gitbook": "2.x.x" }
```

이 옵션은 깃북의 버전을 찾는데 사용된다. 이것은 책을 생성하는데 사용된다. 이 형태는 시멘틱조건을 따른다.

제목

```
{ "title": "내 놀라운 책" }
```

이 옵션은 책의 제목을 정의한다. 기본값은 README 첫 번째 제목을 따른다.

gitbook.com의 이 값은 플렛폼에 입력 타이틀에서 정의된다.

> 역주
>
> `book.json` 을 수정하는 것 보다 https://www.gitbook.com/book/[아이디]/[책이름]/details
>
> 페이지의 setting 에 들어가서 설정하는 것이 더 확실히 반영됩니다.

설명

```
{ "discription": "이 책은 좋은 책이다!" }
```

이 옵션은 책에대한 설명을 정의한다. 기본 값은 README의 첫 번째 절을 따른다.

gitbook.com의 이 값은 설정된 내용에 따르거나 특정 언어에서 정의된다.

방향

```
{ "direction": "rtl" }
```

이 옵션은 언어의 출력방향을 변경할 때 사용된다. 이것은 `language` 의 언어에 따라 올바른 텍스트 방향으로 설정하는것을 권장한다.

스타일

이 옵션은 커스텀 css를 책에서 사용할 때 쓰인다.

예:

```
{
    "styles": {
        "website": "styles/website.css",
        "ebook": "styles/ebook.css",
        "pdf": "styles/pdf.css",
        "mobi": "styles/mobi.css",
        "epub": "styles/epub.css"
    }
}
```

플러그인

```
{ "plugins": ["mathjax"] }
```

책에서 쓰이는 플러그인 리스트의 시작은 'book.json' 설정에 정의된다.

플러그인 설정

```
{
    "plugins": ["myplugin"],
    "pluginsConfig": {
        "myPlugin": {
            "message": "Hello World"
        }
    }
}
```

이 옵션은 모든 플러그인들의 설정들을 포함한다.

구조

이 옵션은 GitBook의 파일경로를 덮는데 사용된다.

예를들어 만역 `INTRO.md` 파일을 `README.md` 대신 사용하길 원한다면

```
{
    "structure": {
        "readme": "INTRO.md"
    }
}
```

구조 타입은 `readme` , `langs` , `summary` , `glossary` 이다.

변수

```
{
    "variables": {
        "myTest": "Hello World"
    }
}
```

변수에 대한 자세한 정보는 템플릿을 참조하라.

플러그인

플러그인은 GitBook이나 ebook 그리고 웹 사이트의 기능을 포함하는 최고의 방법이다. 플러그인들은 그 종류도 다양하다.수학함수 출력이나 Google Analytic과 같은...

플러그인 찾는 방법

플러그인은 plugins.gitbook.com에서 쉽게 찾을 수 있다.

플러그인 인스톨 방법

플러그인을 추가하고 싶다면 `book.json` 에 다음과 같이 쓰면 된다.

```
{
    "plugins": ["myPlugin", "anotherPlugin"]
}
```

너는 또한 특정한 버전을 지정하여 사용할 수도 있다. `myPlugin@0.3.1` 이것은 구 버전 Gitbook등을 사용할 때 유용하다.

만약 책을 로컬에서 빌드하고 싶다면 다운로드 하고 플러그인을 준비하여 `gitbook install` 하는 것으로 간단하게 할 수 있다.

수학 & TeX

GitBook은 수학 방정식과 TeX 플러그인을 지원한다. 그를 위한 공식 플러그인은 mathjax와 katex 두가지 이다.

수학 플러그인 사용하기

수학 플러그인을 사용하려면 `book.json` 에 다음과 같이 추가하라

```
{
    "plugins": ["mathjax"]
}
```

MathJax와 KaTeX의 차이점

mathjax와 katex플러그인은 TeX를 출력하는 구현 방식이 다르다. KaTeX와 MathJax의 자세한 내용은 각각은 오픈 소스 라이브러리이며 해당 내용을 참조하라.

MathJax 전체 TeX 문법을 지원한다. 그러나 ebooks(PDF, ePub 그리고 Mobi)로 출력할 수 없다. KaTeX는 모든 포멧을 지원한다. (web, ebooks) 그러나 모든 문법을 지원하지는 않는다.

콘텐츠에 수학 추가하기

```
임의 수학 식: $$a \ne 0$$
```

수학 식은 새 줄에 추가된다.

```
임의 수학 식:

$$
a \ne 0
$$
```