AsciidoctorとGradleでつくる文書執筆環境

https://github.com/h1romas4/asciidoctor-gradle-template

2023-06-28

はじめに

本文書は `Asciidoc` とその Ruby による実装である `Asciidoctor` を用いて Asciidoc 文書を執筆する環境を構築する手順を示します。実行環境は Windows、Linux、macOS の各 OS に対応しています。

この文書の手順より以下のことができるようになります。

Asciidoc 形式で執筆した文書を HTML/PDF 形式に変換

Asciidoc 文書変換用 Gradle スクリプト

テキストエディターで変換結果をリアルタイムにプレビューしながら文書を編集

Visual Studio Code と AsciiDoc 拡張設定

Asciidoc は表現力の高い文書をテキストファイルベースで執筆できるテキストプロセッサーです。他の軽量テキストプロセッサーが持たない文書間のインクルードやソースコードの挿入などの機能も有し、かつ簡潔です。特に技術文書の執筆には大きな力を発揮することでしょう。

Asciidoc の表現力を示すひとつの例は、このような脚注表現です。

一般的にこのようなテキストプロセッサーを用いた執筆環境を構築するためには多くの準備が必要となりますが、本文書の手順は極力初期導入するプロダクトを少なく、簡単に快適な執筆環境を整えられるよう考えられました。

具体的には文書の変換に、実行を JVM 環境だけに依存する `AsciidoctorJ` と `Gradle` を活用し、執筆環境については `Visual Studio Code` を用いることでリアルタイムに文書をプレビューしながら、最後にコマンド一つで HTML/PDF 化できるように準備してあります。また PDF 出力に関しては、フォントセットや禁則処理設定をプロジェクトにもたせることにより、実行環境に関わらず同一出力が得られるようになっています。

本文書がみなさんの執筆活動のお手伝いになれば幸いです。

謝辞

本文書の手順の実装であるビルドスクリプトやテーマでは次のプロダクトと技術資料が使われています。

プロダクト名の隣にライセンスを併記しました。商用利用等で制限のあるプロダクトはありませんが、それぞれライセンスを確認してください。

Font

- 源真ゴシック - SIL Open Font License 1.1 - http://jikasei.me/font/genshin/
- 源様明朝 - SIL Open Font License 1.1 - https://github.com/ButTaiwan/genyo-font
- Ricty Diminished - SIL Open Font License 1.1 - https://github.com/edihbrandon/RictyDiminished

- Morisawa BIZ UDGothic - SIL Open Font License 1.1 - https://github.com/googlefonts/morisawa-biz-ud-gothic
- Morisawa BIZ UDMincho - SIL Open Font License 1.1 - https://github.com/googlefonts/morisawa-biz-ud-mincho
- Open Iconic - MIT License - https://github.com/iconic/open-iconic/

Asciidoc

- Asciidoctor - MIT License - https://asciidoctor.org/
- Asciidoctorj - Apache License 2.0 - https://github.com/asciidoctor/asciidoctorj
- Asciidoctor.js - MIT License - https://asciidoctor.org/docs/asciidoctor.js/
- Asciidoctor PDF - MIT License - https://asciidoctor.org/docs/asciidoctor-pdf/
- Asciidoctor Gradle Plugin Suite - Apache License 2.0 - https://github.com/asciidoctor/asciidoctor-gradle-plugin
- asciidoctor-pdf-linewrap-ja - MIT License - https://github.com/fuka/asciidoctor-pdf-linewrap-ja
- asciidoctor-nabetani - MIT License - https://github.com/nabetani/asciidoctor-nabetani

Build Tool

- SDKMAN - Apache License 2.0 - https://sdkman.io/
- Gradle - Apache License 2.0 - https://gradle.org/

Text Editor

- Visual Studio Code - Microsoft - https://code.visualstudio.com/
- asciidoctor-vscode - MIT License - https://github.com/asciidoctor/asciidoctor-vscode

Guide

- asciidoctor-pdfでかっこいいPDFを作る - https://qiita.com/kuboaki/items/67774c5ebd41467b83e2

素晴らしい成果を公開されているみなさまに感謝します。

目次

1. Asciidoc 文書変換用スクリプトを使う準備

本手順で用いる Asciidoc 文書変換用スクリプトはビルドツールである Gradle を活用しており、実行するためには Java 実行環境が必要です。

Java 実行環境は、文書変換スクリプトを動作させる過程で唯一 OS 環境に手動で導入する必要があるプロダクトです。それ以外のものは Gradle によりプロジェクトとして独立した形で自動的に導入されます。

お使いのコンピューターのコマンドライン環境（macOS/Linux ではターミナル、Windows では cmd.exe か powershell.exe）で `java -version` コマンドを入力し、Java 8 以上のバージョンが表示されるようであれば既に準備は整っています。

macOS/Linux の場合

```
$ java -version
openjdk version "1.8.0_192"
OpenJDK Runtime Environment (Zulu 8.33.0.1-macosx) (build 1.8.0_192-b01)
OpenJDK 64-Bit Server VM (Zulu 8.33.0.1-macosx) (build 25.192-b01, mixed mode)
```

Windows の場合

```
C:¥> java -version
openjdk version "1.8.0_192"
OpenJDK Runtime Environment (AdoptOpenJDK)(build 1.8.0_192-b12)
OpenJDK 64-Bit Server VM (AdoptOpenJDK)(build 25.192-b12, mixed mode)
```

本文書では Java 11 を用いて解説します。

1.1. Java 実行環境の導入（macOS / Linux の場合）

もし macOS / Linux 環境に Java 実行環境がなければ `SDKMAN` を利用することで、ターミナルから簡単に導入することができます。

> https://sdkman.io/
>
> SDKMAN! is a tool for managing parallel versions of multiple Software Development Kits on most Unix based systems.
>
> — SDKMAN

手順. SDKMAN を用いた Java の導入

```
$ curl -s "https://get.sdkman.io" | bash ❶
$ source "$HOME/.sdkman/bin/sdkman-init.sh" ❷
$ sdk list java ❸
================================================================================
Available Java Versions for Linux 64bit
================================================================================
 Vendor        | Use | Version      | Dist    | Status     | Identifier
--------------------------------------------------------------------------------
 Temurin       |     | 19           | tem     |            | 19-tem
               |     | 17.0.4       | tem     | local only | 17.0.4-tem
               |     | 17.0.4.1     | tem     |            | 17.0.4.1-tem
               |     | 11.0.16.1    | tem     |            | 11.0.16.1-tem
               |     | 8.0.345      | tem     |            | 8.0.345-tem
               |     | 8.0.342      | tem     |            | 8.0.342-tem
               |     | 8.0.332      | tem     |            | 8.0.332-tem
$ sdk install java 11.0.16.1-tem ❹
```

❶ SDKMAN を導入します。

❷ SDKMAN を環境に設定します。

❸ 導入できる Java のバージョンを一覧します。

❹ Java 11 系の最新バージョンを指定して Java を導入します。

また、Gradle は `JAVA_HOME` 環境変数に実行環境の Java のパスが設定されていることを期待していますので、`.bashrc` で次のように `JAVA_HOME` を設定します。

手順. JAVA_HOME の設定

```
$ vi ~/.bashrc ❶
export JAVA_HOME=~/.sdkman/candidates/java/current ❷
$ source ~/.bashrc ❸
```

❶ vi エディタで `.bashrc` を開きます。

❷ 本ラインをファイルの最下部に追加し vi を保存終了します。

❸ 設定を適用します。

これで準備完了です。

SDKMAN について

SDKMAN は主に Java エコシステムの開発環境をコマンドラインから簡単に導入・設定するためにつくたれた管理ソフトウェアです。

たとえば簡単に各種 Java のバージョンを導入し切り替えることができます。

手順. SDKMAN による Java のバージョン切り替え

```
$ sdk install java 8.0.345-tem ❶
$ sdk default java 8.0.345-tem ❷
$ sdk default java 11.0.16.1-tem ❸
```

❶ Java 8 を導入

❷ Java 8 をデフォルトに設定

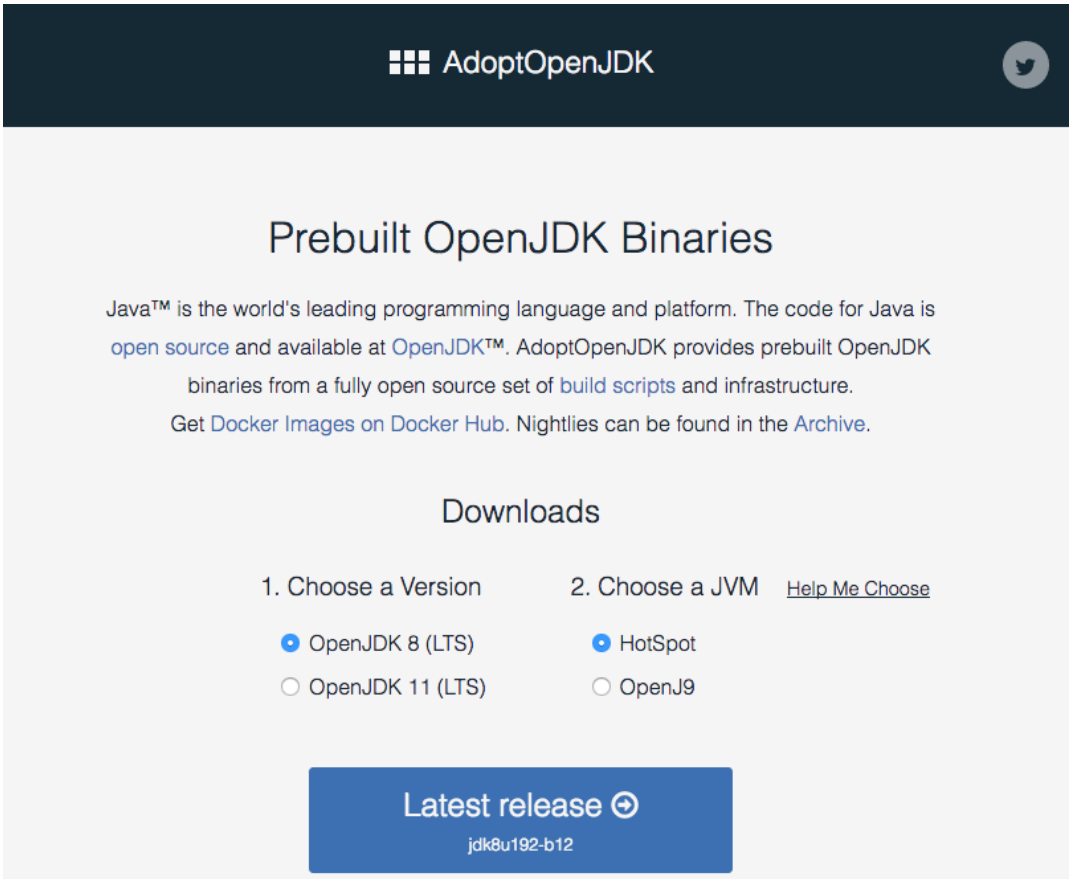
❸ Java 11 をデフォルトに戻す

1.2. Java 実行環境の導入（Windows の場合）

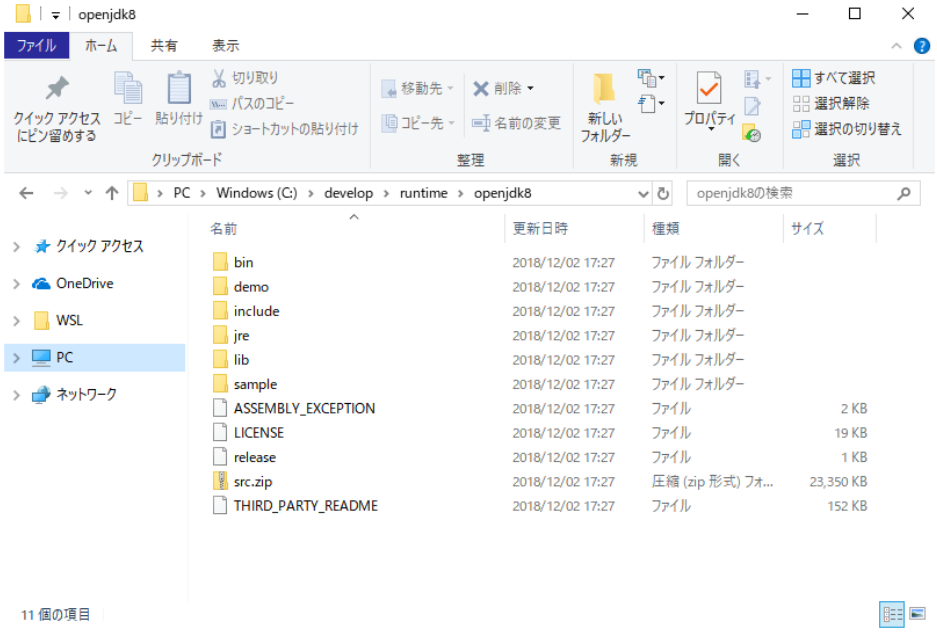
もし Windows 環境に Java 実行環境がなければ AdoptOpenJDK プロジェクトが提供する OpenJDK のバイナリを導入すると良いでしょう。

> https://adoptopenjdk.net
>
> Java™ is the world's leading programming language and platform. The code for Java is open source and available at OpenJDK™. AdoptOpenJDK provides prebuilt OpenJDK binaries from a fully open source set of build scripts and infrastructure.
>
> — AdoptOpenJDK

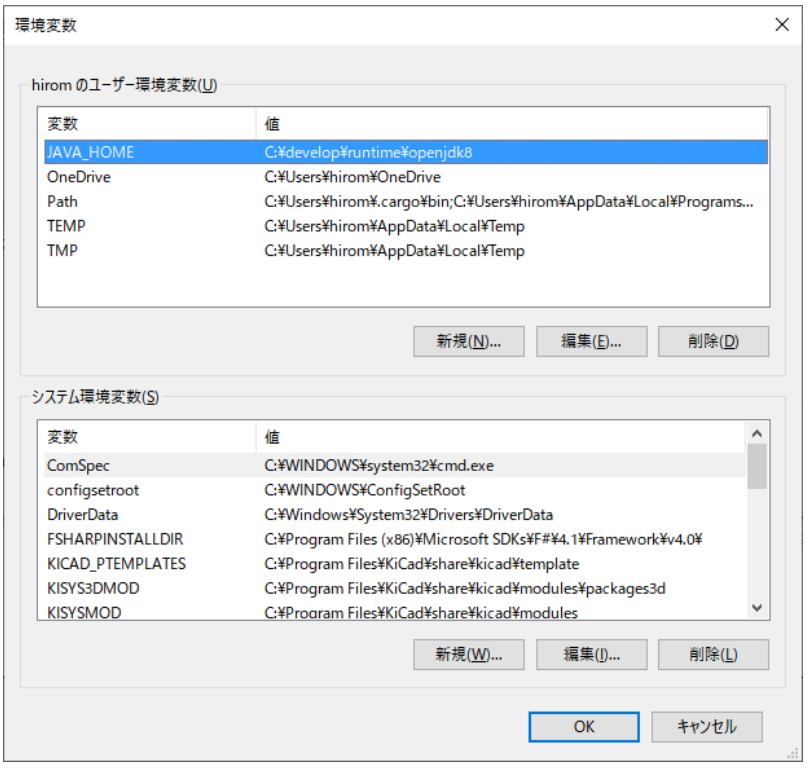
`https://adoptopenjdk.net` サイトにブラウザでアクセスし、OpenJDK 11 (LTS) - HotSpot を選択した後、zip ファイルをダウンロードしてください。



zip ファイルを任意の場所に展開します。ここでは `C:\develop\runtime\openjdk11` に展開したとします。



Gradle は `JAVA_HOME` 環境変数に実行環境の Java のパスが設定されていることを期待していますので、**エクスプローラー › PC（右クリック） › プロパティー › 詳細設定 › 環境設定** からユーザー環境変数に `JAVA_HOME` を追加し、先ほど .zip を展開したパス（`C:\develop\runtime\openjdk11` ）を設定します。



Gradle は JAVA_HOME 環境変数を元に Java の実行環境を探すため、`java` コマンドを使うための PATH 環境変数は設定しなくてもかまいません。

これで準備完了です。

2. Asciidoc から HTML/PDF 文書を作成する

2.1. サンプル文書の変換を試す

環境の準備ができましたので Asciidoc 文書を HTML/PDF に変換してみます。

変換に使うスクリプトは github のリポジトリに公開されており、リポジトリには HTML/PDF 変換に使うファイル一式と、文書サンプルとして "この文書" の Asciidoc ファイルが置かれています。まずはサンプル文書が正しく変換できるかを試してみましょう。

macOS / Linux の場合は次のようにします。

手順. PDF 変換ビルドスクリプトの取得と実行

```
$ curl -L -O https://github.com/h1romas4/asciidoctor-gradle-template/archive/master.zip ❶
$ unzip master.zip ❷
$ cd asciidoctor-gradle-template-master ❸
$ ./gradlew docs ❹
BUILD SUCCESSFUL in 19s ❺
2 actionable tasks: 2 executed
```

❶ リポジトリのファイルをダウンロードします。

❷ ダウンロードした .zip ファイルを展開します。

❸ カレントディレクトリを展開したフォルダの中に移します。

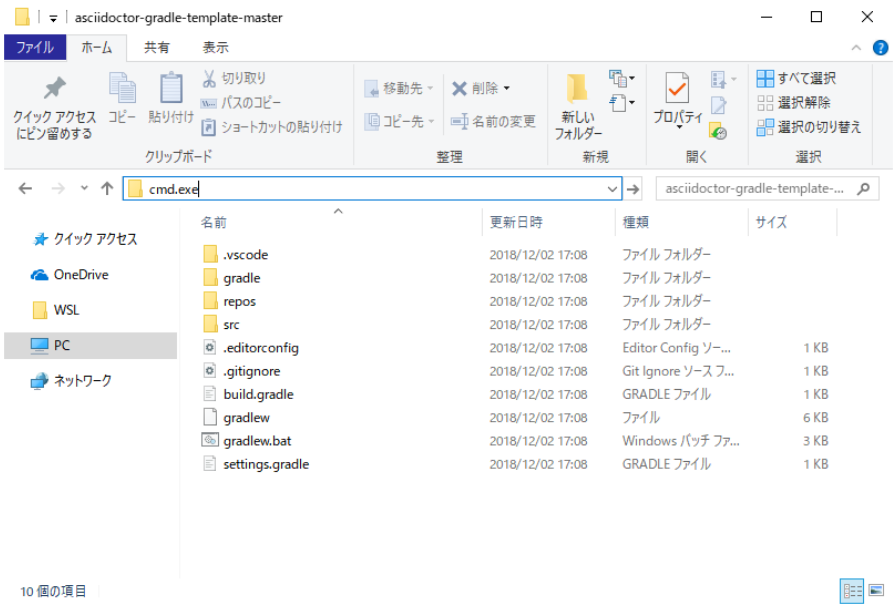
❹ Gradle のビルドを実行します。初回実行時はビルドに必要なファイルをダウンロードするため少し時間がかかります。次回は数秒で完了します。

❺ `BUILD SUCCESSFUL` が出力されればビルド成功です。

Windows をお使いの場合は同等の操作をブラウザとエクスプローラーを使って行います。

Windows の場合 .zip ファイルの展開先はマルチバイト文字を含まないパスにしてください。JRuby の制約により変換処理がエラーとなります。また、プロジェクト内部で使うフォルダやファイル名のマルチバイト名も同様です。

1. ブラウザを使って `https://github.com/h1romas4/asciidoctor-gradle-template/archive/master.zip` にアクセスしリポジトリのファイルを取得します。
2. ダウンロードした .zip ファイルを右クリックし展開します。
3. 展開したフォルダ内をエクスプローラーで表示した上で、アドレスバーに `cmd.exe` と入力し、このフォルダをカレントディレクトリとしてコマンドプロンプトを起動します。



4. `.\gradlew.bat docs` と入力し Gradle ビルドを実行します。初回実行時はビルドに必要なファイルをダウンロードするため少し時間がかかります。次回は数秒で完了します。
5. `BUILD SUCCESSFUL` が出力されればビルド成功です。

プロキシーサーバーの設定

もしお使いのコンピューターがプロキシーサーバー経由のインターネットアクセスを行う場合は、次のコマンドを `./gradlew docs` をする前に入力してください。インターネットを使ったライブラリの取得が正しく行われるようになります。ホスト名（`example.com`） と ポート番号（`8080`）部分はそれぞれの環境に合わせてください。

手順. プロキシーサーバー設定（Windows）

```
set JAVA_OPTS=-DproxyHost=example.com -DproxyPort=8080
```

手順. プロキシーサーバー設定（macOS / Linux）

```
export JAVA_OPTS=-DproxyHost=example.com -DproxyPort=8080
```

Asciidoc から変換された文書は次の場所に格納されます。`docs` フォルダはビルド出力専用となっており、ビルド時にいったん全てのファイルが削除されますのでユーザーファイルは配置しないように注意してください。

```
docs/index.html
docs/index.pdf
```

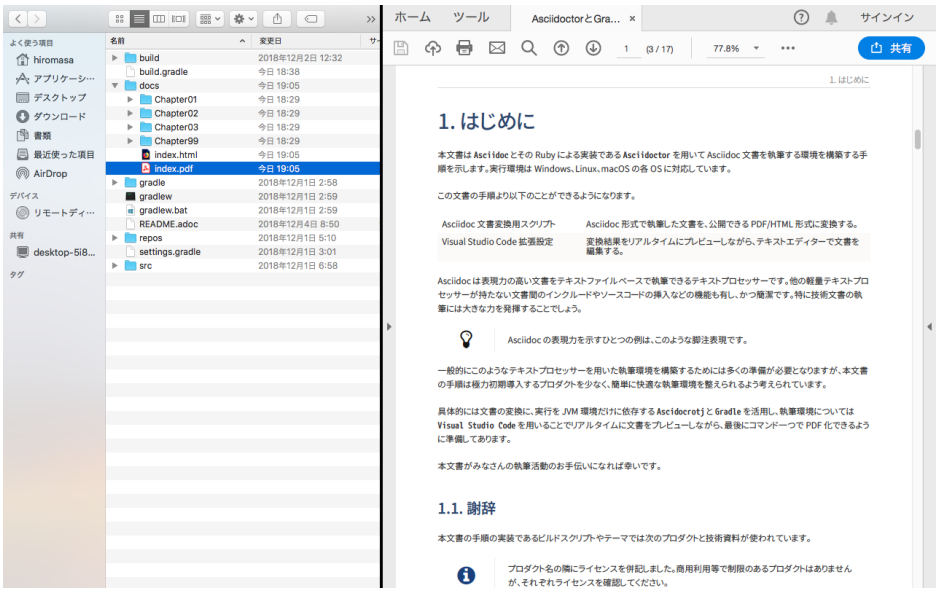


図 1. PDF 文書

図 2. HTML 文書

文書を github 上に公開する場合は、プロジェクトのファイルを全てコミットし、GitHub Pages に `docs` フォルダを指定することで継続的に文書をパブリッシュすることができます。

2.2. テキストエディタで Asciidoc 文書を編集する

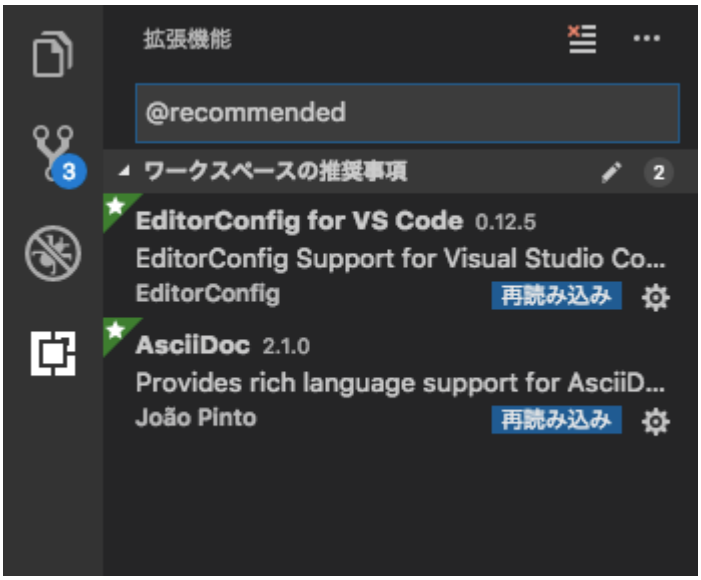
プロジェクトに配置された Asciidoc 文書は Visual Studio Code を利用してリアルタイムに変換結果をプレビューしながら編集できるように準備されています。

プロジェクトフォルダ（`asciidoctor-gradle-template-master`）を Visual Studio Code で開き、拡張機能の推奨事項から拡張を導入します。

拡張機能の推奨は `.vscode/extention.json` で設定されています。文書に応じて設定し、執筆メンバーの環境を揃えることができます。

1. **[すべてインストール]** ボタンをクリックします。

2. **[再読み込み]** ボタンをクリックします。



3. Asciidoc 文書（`src/docs/asciidoc/index.adoc`など）を Visual Studio Code のエクスプローラーから選択して開き、文書を開いたエディタ部右上に配置された **[Open Preview to the Side]**アイコンをクリックすると、画面右側に Asciidoc 文書の変換がリアルタイムに確認できるプレビューが表示されます。

2.3. 文書のファイル構成

文書は次のようなファイルで構成します。サンプル文書から執筆したい文書に合わせてカスタマイズしていくとよいでしょう。

文書を構成するファイル

```
src/docs/asciidoc/index.adoc ❶
src/docs/asciidoc/attribute.adoc ❷
src/docs/asciidoc/@style/*.css ❸
src/docs/asciidoc/@style/pdf-theme.yml ❹
src/docs/asciidoc/@font/**/*.ttf ❺
src/docs/asciidoc/Chapter{number}/index.adoc ❻
```

❶ 文書を作成する起点となる Asciidoc 文書です。大きな文書の場合はここから他の Asciidoc 文書を `include` して構成していきます。

❷ 文書設定をするためのファイルです。各文書から `incliude` します。

❸ HTML 出力とプレビュー用のスタイルシートです。文書に合わせて修正することができます。

❹ PDF 文書に変換する際に使われるスタイル定義です。文書に合わせて修正することができます。Asciidoctor PDF Theming Guide からドキュメントが参照できます。

❺ PDF 文書に埋め込みされるフォントファイルです。`pdf-theme.yml` から参照されています。TrueType フォント `.ttf` が指定できます。

❻ `src/docs/asciidoc/index.adoc` から参照される子文書です。後述の `build.gradle` による画像パス解決のためフォルダ名は `Chapter{number}` とします。

変換スクリプトとなる `build.gradle` の `attribute` 設定でこれらのファイルパスや変換に必要な属性を設定しています。

build.gradle

```
asciidoctor {
    asciidoctorj {
        attributes 'stylesdir': '@style',
            'stylesheet': 'asciidoctor.css'
    }
}

asciidoctorPdf {
    asciidoctorj {
        attributes 'pdf-themesdir': "@style",
            'pdf-theme': 'pdf-theme.yml'
    }
}
```

`src/docs/asciidoc/attribute.adoc` ではエディタのリアルタイムプレビュー用の属性定義を行い、文書変換スクリプトでその値を上書きするように構成すると良いでしょう。

Asciidoctor で利用可能な属性は次から参照できます。

> https://asciidoctor.org/docs/user-manual/#attributes
>
> Attributes are one of the features that sets Asciidoctor apart from other lightweight markup languages. Attributes can activate features (behaviors, styles, integrations, etc) or hold replacement (i.e., variable) content.
>
> — asciidoctor.org

また、文書に挿入する画像ファイルは `images` というフォルダ名内に配置され、その文書からの相対パスでリンクされることを期待しています。設定は次の部分で変更可能です。

src/docs/asciidoc/attribute.adoc

```
ifndef::imagesdir[:imagesdir: ./images]
```

build.gradle

```
copy {
    from 'src/docs/asciidoc/'
    include 'Chapter*/images/*' // 子文書のフォルダ名
    into 'docs'
}
```

3. Asciidoc 記法

3.1. 文書中の相互参照

文書内の参照リンクは `.adoc` 文書中で次のように指定します。HTML/PDF でクリックリンクが生成されます。

```
[[project-structure]] ❶
== Asciidoc 記法

[[introduction]] ❷
== はじめに

<<project-structure,章の冒頭>> ❶
<<Chapter01/index.adoc#introduction,はじめに>> ❷
```

❶ 同一 `.adoc` 内での内部参照

❷ `.adoc` をまたいだドキュメント間参照

リンクの例

- 章の冒頭 を参照してください･･･
- はじめに で述べたように･･･

3.2. ソースコードシンタックスハイライト

`[sorce]` ブロックを用いて文書中にソースコードをシンタックスハイライト付きで埋め込みます。また、`// <1>` 形式でソースコード下部に対応する説明が書けます。

```
[source, javascript]
[caption=""]
.JavaScript ソースコードのシンタックスハイライトの例
----
function hello() {
    console.log("Hello, World!"); // <1>
}
----

<1> コンソールに ``Hello, World`` を出力
```

JavaScript ソースコードのシンタックスハイライトの例

```
function hello() {
    console.log("Hello, World!"); ❶
}
```

❶ コンソールに `Hello, World` を出力

3.3. ファイルインクルード

`.adoc` ファイルから別のファイルをインクルードできます。ソースコードや後述のダイアグラム形式などを別ファイルとして切り出すのに便利です。

Gradle のファイル更新監視は `.adoc` からインクルードされたファイルまでは及びません。インクルード先のファイルだけを書き換えた場合は `./gradlew docs` が処理なしで終了してしまいますので、この場合は `./gradlew clean` するなどしてからリビルドしてください。

```
[source, rust]
----
include::source/hello.rs[]
----
```

インクルードにより挿入されたソースコードの例

```rust
fn main() {
    println!("Hello, world!");
}
```

3.4. 表形式

表形式は次のようなコードで定義することができます。

```
[format="csv",cols="1,2"]
[frame="topbot",grid="none"]
[caption=""]
.表形式の例
|======
カラム1,カラム2
キー1,バリュー1
キー2,バリュー2
|======
```

表形式の例

カラム1	カラム2
キー1	バリュー1
キー2	バリュー2

3.5. 脚注

補足点を脚注するいくつかの表現があります。

```
NOTE: メモ

TIP: チップス

IMPORTANT: 重要

WARNING: 警告

CAUTION: 注意
```

メモ

チップス

重要

警告

注意

3.6. コラム表現

```
.コラム
****
ブロックを使ってコラム表現を挿入します。

吾輩はコラムである。名前はまだ無い。
****
```

コラム

ブロックを使ってコラム表現を挿入します。

吾輩はコラムである。名前はまだ無い。

3.7. asciidoctorj-diagram

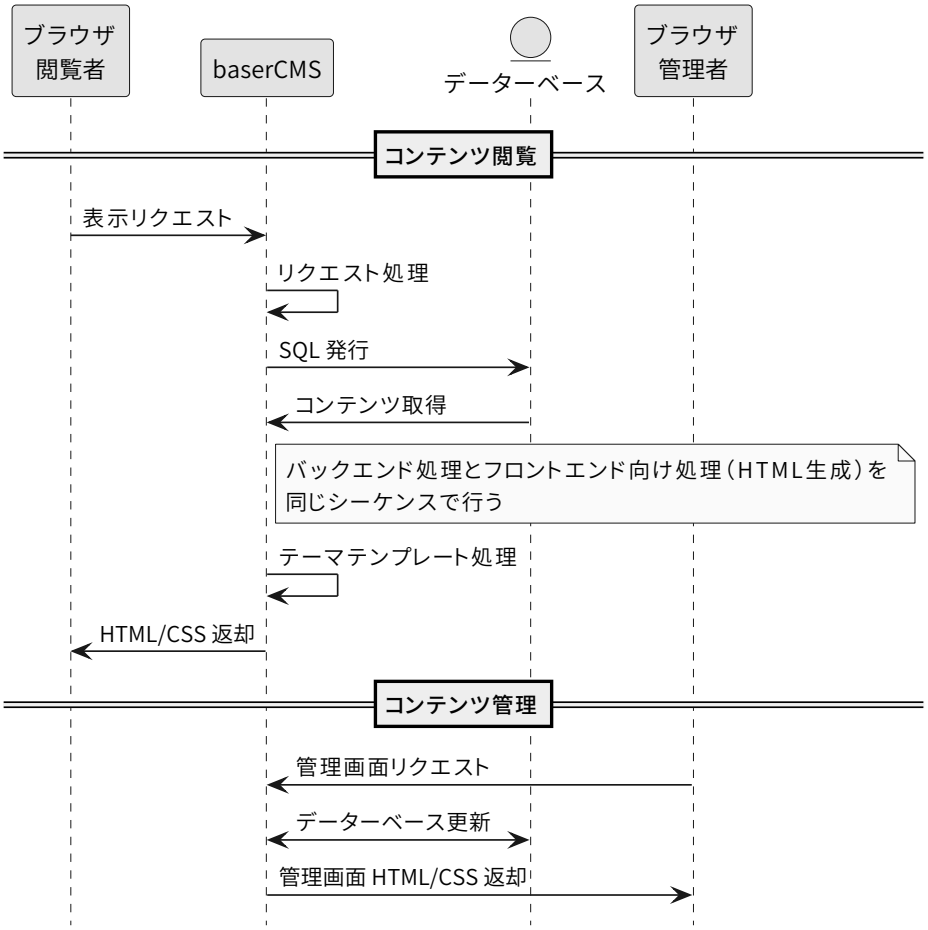
本変換ビルドスクリプトでは `asciidoctorj-diagram` が有効になっており、いくつかのダイアグラム記法を追加のアドイン無しで使うことができます。

3.7.1. PlantUML

記事中に PlantUML 形式のダイアグラムを記述し、生成された `.svg` ベクター画像を文書に展開するには次のようにします。

シーケンス図の例

```
[plantuml, diag-sequence-sample2, svg, pdfwidth=70%, width=70%]
[caption=""]
----
@startuml diag-sequence-sample2
skinparam monochrome true
hide footbox
participant "ブラウザ\n閲覧者" as browser
participant "baserCMS" as basercms
entity データーベース as Entity
participant "ブラウザ\n管理者" as admin
== コンテンツ閲覧==
browser -> basercms: 表示リクエスト
basercms -> basercms: リクエスト処理
basercms -> Entity: SQL 発行
Entity -> basercms: コンテンツ取得
note right of basercms: バックエンド処理とフロントエンド向け処理（HTML生成）を\n同じシーケンスで行う
basercms -> basercms: テーマテンプレート処理
basercms -> browser: HTML/CSS 返却
== コンテンツ管理==
admin -> basercms: 管理画面リクエスト
basercms <-> Entity: データーベース更新
basercms -> admin: 管理画面 HTML/CSS 返却
@enduml
----
```

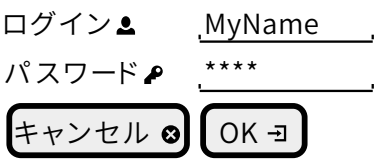

ブラウザ
閲覧者
baserCMS
データーベース
ブラウザ
管理者
コンテンツ閲覧
表示リクエスト
リクエスト処理
SQL 発行
コンテンツ取得
バックエンド処理とフロントエンド向け処理（HTML生成）を
同じシーケンスで行う
テーマテンプレート処理
HTML/CSS 返却
コンテンツ管理
管理画面リクエスト
データーベース更新
管理画面 HTML/CSS 返却

Salt の例(1)

```
[plantuml, diag-salt-sample1, svg, pdfwidth=30%, width=280px]
[caption=""]
----
' https://github.com/iconic/open-iconic/
@startsalt diag-salt-sample1
{
{T
 + <&folder> wp-admin
 + <&folder> wp-content
 ++ <&folder> themes
 +++ <&folder> my-theme   | 新規作成
 ++++ <&file> style.css   | 新規作成
 ++++ <&file> index.php   | 新規作成
 ++ <&folder> uploads
 + <&folder> wp-includes
 + <&file> .htaccess      | 自動作成
 + <&file> index.php
}
}
@endsalt
----
```

Salt の例(2)

```
[plantuml, diag-salt-sample2, svg, pdfwidth=30%, width=30%]
[caption=""]
----
@startsalt diag-salt-sample2
{
  ログイン<&person> | "MyName    "
  パスワード<&key>  | "****      "
  [キャンセル <&circle-x>] | [OK <&account-login>]
}
@endsal
----
```



AsciiMath の例

```
[plantuml, math-sample1, svg, pdfwidth=70%, width=70%]
[caption=""]
----
@startmath math-sample1
f(t)=(a_0)/2 + sum_(n=1)^ooa_ncos((npit)/L)+sum_(n=1)^oo b_n\ sin((npit)/L)
@endmath
----
```

$$f\left(t\right)=\frac{a_0}{2}+\sum_{n=1}^{\infty}a_n\cos\left(\frac{n\pi t}{L}\right)+\sum_{n=1}^{\infty}b_n\ \sin\left(\frac{n\pi t}{L}\right)$$