

**Андреас Оффенхайзер**Posted on 1 авг. 2018 г. • Первоначально опубликовано на anoff.io 31 февраля 2018 г..

9



3

Собственные диаграммы Markdown с помощью PlantUML

#webdev #Учебник #github #Продуктивность

Этот пост был первоначально опубликован в моем блоге в разделе<https://anoff.io/blog/2018-07-31-diagrams-with-plantuml/> и размещен для

В этом посте будут рассмотрены основы PlantUML и то, как его можно использовать в проектах GitLab или GitHub, а также бесшовная локальная среда разработки с использованием Visual Studio Code.

Этот пост я хотел написать месяцами. В последнее время я широко использую PlantUML на работе, а также в своих частных проектах. Вы можете видеть, что они используются в моих проектах [plantbuddy](#) и [techradar](#) на GitHub. Используя его в разных местах и для разных целей, я столкнулся с кучей проблем, которыми хочу поделиться в этом посте.

Основы PlantUML

Для тех, кто не знает [PlantUML](#): Это инструмент с открытым исходным кодом, который позволяет вам определять диаграммы UML с помощью обычного текста. Доступны различные [типы диаграмм](#), которые описываются с помощью пользовательского синтаксиса, но по общей схеме. В этом посте не будут вдаваться в подробности каждого из этих типов диаграмм, потому что веб-сайт PlantUML довольно хорошо справляется с описанием [последовательности](#), [компонента](#), [действия](#) и других типов диаграмм.

Базовая диаграмма компонентов, показывающая поток данных, может быть построена с использованием следующей разметки:

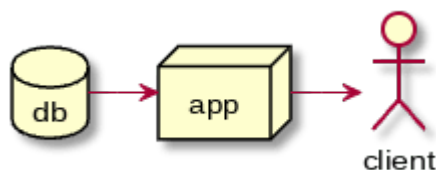
```
@startuml
component
actor client
node app
```

```
database db
```

```
db -> app
```

```
app -> client
```

```
@enduml
```



Причины любить PlantUML 🤗

Управление версиями 📦

Очень важным аспектом при разработке программного обеспечения и написании документации является их синхронизация. Одна из частей заключается в обновлении документации, если обновляется сам код. Другой важной частью является управление версиями - обычно программное обеспечение версифицируется с помощью `git` или аналогичных систем. Помещая документацию в тот же репозиторий, что и код, вы всегда проверяете правильное состояние документации на соответствующий момент времени.

По этой причине я люблю размещать всю свою документацию либо в исходном коде в виде комментариев, либо в виде файлов Markdown рядом с исходным кодом. Чего мне всегда не хватало при таком подходе, так это визуализации вещей. Размещение файлов PowerPoint / Keynote / Visio / Enterprise Architect.. в репозитории гарантирует, что ваши диаграммы всегда имеют версию кода, но они недоступны для просмотра в веб-интерфейсах. На помощь приходят рендеринг PlantUML и GitLab: GitLab позволяет вам [встроить диаграммы PlantUML](#) непосредственно в ваши файлы Markdown, и они будут отображаться "на лету" при просмотре файлов в браузере.

Еще одно преимущество PlantUML перед упомянутыми инструментами заключается в том, что, определяя ваши диаграммы простым текстом, вы делаете их различимыми в запросах на извлечение. Рецензенты всегда могут увидеть, какие изменения были внесены, и легко сравнить изменения на диаграмме с изменениями, внесенными внутри кода.

Синтаксис 🐟

Базовый синтаксис PlantUML очень лаконичен и создает хорошую основу для различных типов диаграмм. Это также очень умно в том смысле, что позволяет

писать диаграммы с разными вариантами, например, вы можете объявлять / создавать экземпляры всех узлов вверху, но если вы их не объявите, они будут выведены автоматически. То же самое касается [макросов и определений](#), которые позволяют создавать диаграммы большего размера или общую библиотеку для вашей команды.

Недавно я создал [чит-лист PlantUML](#) для множества полезных трюков - однако он не охватывает самые основы синтаксиса PlantUML. Вы можете просмотреть [последнюю версию](#) или [исходный код LaTeX](#) на GitHub.

Верстка 🖨️

По сравнению с редакторами WYSIWYG, диаграммы PlantUML определяют только компоненты и их взаимосвязь, но не фактический макет диаграммы. Вместо этого диаграмма выводится с помощью детерминированного алгоритма в процессе рендеринга. Это полезно при указании диаграммы, потому что вы фокусируетесь только на содержимом - сравнимо с написанием документа LaTeX.

К сожалению, механизм верстки не так хорош, как вам иногда хотелось бы, и особенно в диаграммах компонентов с более чем 10 узлами вы можете в конечном итоге потратить много времени на принудительное выполнение определенных макетов вручную.

Для диаграмм последовательности и действий автоматическая компоновка отлично работает даже для очень больших диаграмм. После того, как вы построите несколько диаграмм и заметите, как легко просто перемещать строки кода вверх и вниз, а изменения в коде сразу же отражаются в вашей документации, вам понравится автоматическая компоновка.

Делитесь ими где угодно 📱

Если вы хотите *заморозить* версию диаграммы и отправить ее кому-то за пределами вашей организации, вы можете просто отправить им безумно длинный URL (например, <http://www.plantuml.com/plantuml/png/5Son3G8n34RXtbFyb1GiG9MM6H25XGsnH9p8SGgs1rpxzF1lcHovse-yYw8QdlJl2v--N93rJ2Bg4ED1SBlG0pn6wDiu5NiDcAU6piJzTgKN5NNPu040>), который кодирует полное определение диаграммы. Вы также можете просто встроить этот URL-адрес в тег HTML ``. Если кому-нибудь когда-нибудь понадобится поработать с изображением, все, что вам нужно сделать, это поменять `/plantuml/png` на `/plantuml/uml`, и вы увидите [определение](#) диаграммы.

Это дает всему набору инструментов PlantUML чрезвычайно универсальный способ передачи информации, а также доступные для просмотра изображения.

Локальное развитие

Самый быстрый, не зависящий от платформы и простой способ начать создавать диаграммы PlantUML - это использовать их [онлайн-редактор](#) (кстати. вы можете легко разместить его на ргем, используя [образ plantuml-server Docker](#)). Это прекрасно подходит для создания простых диаграмм с несколькими узлами, но диаграммы большего размера требуют много *предварительного просмотра*, что раздражает в онлайн-редакторе.

Если у вас есть какие-либо другие локальные настройки, пожалуйста, дайте мне знать через [Twitter](#)

Код Visual Studio

Если вы уже используете VS Code, настроить это несложно. В противном случае вам, возможно, всерьез захочется рассмотреть возможность использования его для редактирования диаграмм PlantUML (в Markdown) только потому, что это очень простой процесс.

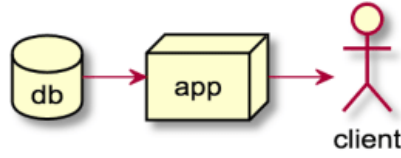
Все, что вам нужно сделать, это получить [расширение PlantUML](#), чтобы включить встроенную в codes функцию [предварительного просмотра Markdown](#) для разбора встроенных диаграмм.

```

33 ``plantuml
34 @startuml component
35 actor client
36 node app
37 database db
38
39 db -> app
40 app -> client
41 @enduml
42 ...
43
44 ![basic component diagram showing
data flowing from a database via

```


following markup:



```

![basic component diagram showing
data flowing from a database via an
app to a client]({{ site.baseurl
|/img/assets/plantuml/diagrams/di1/

```



По умолчанию для плагина требуется, чтобы локальный процесс PlantUML был запущен и принимал запросы на рендеринг. Я рекомендую переключить его на использование сервера для рендеринга; это может быть официальный сервер [plantuml.com](#) сервер, локальный экземпляр или [контейнер](#), работающий локально. После установки плагина перейдите в параметры VS Code (ctrl/⌘ + ,) и измените `plantuml.render` свойство.

```
// PlantUMLServer: Render diagrams by server which is specified with "plantuml.server". It's mu
// Local is the default configuration.
"plantuml.render": "PlantUMLServer",

// Plantuml server to generate UML diagrams on-the-fly.
"plantuml.server": "http://www.plantuml.com/plantuml",
```

Если вы когда-нибудь отключитесь от сети и все еще захотите работать, не забудьте сделать это, `docker run -d -p 8080:8080 plantuml/plantuml-server:jetty` пока у вас еще есть подключение к Интернету. Изображение ~250MB для загрузки. После этого установите `plantuml.server` значение `http://localhost:8080/`, и вы готовы к приключениям в автономном режиме.

На моем MacBook я иногда сталкиваюсь с большим потреблением процессора запущенным контейнером - даже при неактивном рендеринге. Помогает перезапуск контейнера 🙄

Визуализация в SVG / PDF

Этот метод работает, только если диаграммы явно определены в файлах, а не встроены в Markdown.

Чтобы написать этот пост в блоге и создать [ЧИТ-ЛИСТ](#), я поиграл со способами рендеринга диаграмм PlantUML в изображения не в реальном времени. Вы можете использовать [Makefile](#) и [Shell-скрипт](#) для преобразования целой [папки](#) диаграмм PlantUML с `.puml` расширением в `.svg` и `.pdf` [файлы](#).

Скрипт, по сути, запускает определение диаграммы через докеризованный процесс PlantUML, который выводит `.svg`, а затем использует Inkscape для создания `.pdf` файла для импорта его, например, в документы LaTeX.

```
#!/bin/sh
# converts all puml files to svg

BASEDIR=$(dirname "$0")
mkdir -p $BASEDIR/dist
rm $BASEDIR/dist/*
for FILE in $BASEDIR/*.puml; do
  echo Converting $FILE..
  FILE_SVG=${FILE//puml/svg}
  FILE_PDF=${FILE//puml/pdf}
  cat $FILE | docker run --rm -i think/plantuml > $FILE_SVG
```

```
docker run --rm -v $PWD:/diagrams productionwentdown/ubuntu-inkscape inkscape
done
mv $BASEDIR/*.svg $BASEDIR/dist/
mv $BASEDIR/*.pdf $BASEDIR/dist/
echo Done
```

Интеграция с GitLab

В настоящее время эта функция доступна только при предварительной установке GitLab, ее включение [gitlab.com](#) является [открытым вопросом](#). Решение проблемы смотрите в интеграции с GitHub.

Использовать PlantUML в GitLab очень весело. Все, что вам нужно сделать, это [настроить](#) сервер рендеринга для использования, и вы можете просто зафиксировать файлы Markdown со встроенными диаграммами PlantUML, и они будут отображаться для всех, кто посещает веб-интерфейс GitLab.

Замечательно то, что это работает не только с файлами Markdown, переданными в репозиторий git, но и со всеми другими полями в GitLab, которые отображают markdown - практически со всем. У вас также могут быть небольшие диаграммы, помогающие проиллюстрировать некоторые моменты в выпусках.

New Issue

Title

PlantUML

Add [description templates](#) to help your contributors communicate effectively!

Description

Write Preview

```
```plantuml
@startuml
node app
database db
app -- db
@enduml
```
```

[Markdown](#) and [quick actions](#) are supported

New Issue

Title

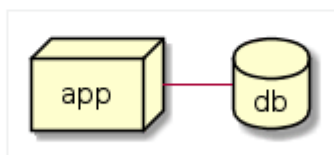
PlantUML

Add [description templates](#) to help your contributors communicate

Description

Write

Preview



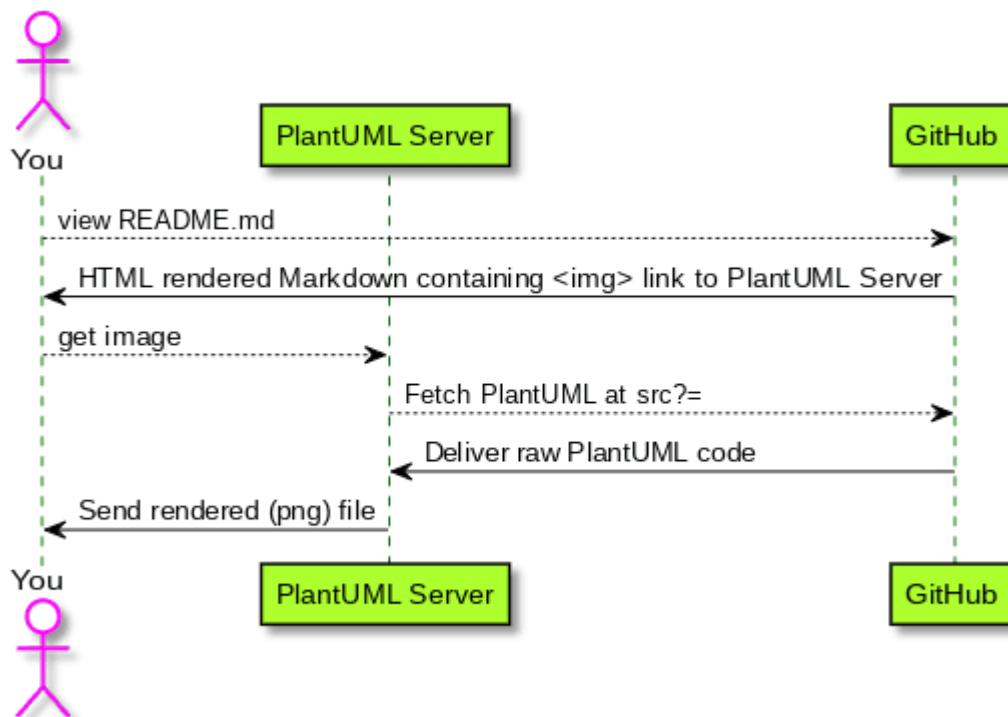
Интеграция с GitHub

Для GitHub нет встроенной интеграции PlantUML и [gitlab.com](#) доступно. Для сохранения перечисленных выше преимуществ, очевидно, что просто визуализировать файлы локально и фиксировать их в `git` - недопустимый обходной путь.

Вместо этого используйте [прокси-службу](#) PlantUML, как описано в [этом обсуждении stackoverflow](#). Это работает так: вместо передачи серверу PlantUML содержимого диаграммы в URL-адресе мы определяем *удаленный URL-адрес*, откуда содержимое может быть извлечено, например, `http://www.plantuml.com/plantuml/proxy?src=https://raw.githubusercontent.com/plantuml/plantuml-server/master/src/main/webapp/resource/test2diagrams.txt`. Этот URL-адрес может быть встроен в HTML `` тег или в синтаксис изображения Markdown ``. Чтобы использовать эту функцию при использовании GitHub, просто укажите *удаленный URL* на необработанную ссылку на диаграмму PlantUML в вашем репозитории.

Вы можете увидеть этот метод в действии в проектах [plantbuddy](#) и [techradar](#) в моей учетной записи на GitHub.

На следующей диаграмме показано, что произойдет, когда вы откроете страницу Markdown, размещенную на GitHub и содержащую такую ссылку:



[Этот пример](#) показывает, что добавление `?cache=no` может быть хорошей идеей из-за [стратегии кэширования](#) Само на GitHub, которая предотвращает обновление ваших изображений при изменении исходного кода.

Недостатком этого подхода является то, что он всегда будет отображать последнюю коммиту в вашем репозитории, даже если вы просматриваете старые версии. Если просмотр старых версий является для вас *строгим* требованием при использовании интеграции с GitHub, то вам может потребоваться создать свой собственный плагин / средство визуализации или оптимизировать локальную среду разработки, потому что, в конце концов, правильная версия `diagrams` всегда будет с исходным кодом, который вы проверили.

Чтобы воспользоваться интеграцией с прокси-сервисом, просто используйте:

```
! [cached image](http://www.plantuml.com/plantuml/proxy?src=https://raw.git
```

```
! [uncached image](http://www.plantuml.com/plantuml/proxy?cache=no&src=http
```

Краткие сведения

Существует два основных способа сохранения диаграмм PlantUML


1. встроены в Markdown

2. сохранять в виде отдельных .puml файлов

В зависимости от вашего набора инструментов, один из них должен быть предпочтительным вариантом для работы с диаграммами в вашем репозитории. Настоятельно рекомендуется размещать диаграммы как можно ближе к коду и не создавать искусственные хранилища документации.

В этом посте рассказывалось о том, как писать и отображать файлы локально в **VS Code**, используя контейнеры **Docker** и как интегрироваться в **GitLab на prem**, а также в общедоступные экземпляры **GitHub** и **GitLab**.

Можно еще многое рассказать о PlantUML, но я надеюсь, что в этой статье вы получили достаточно информации, чтобы начать работу на любой используемой вами платформе. Я рекомендую эту [справочную таблицу по PlantUML](#), которая поможет вам охватить еще более широкий спектр вариантов использования.

Расскажите мне о своем опыте работы с PlantUML или альтернативными интеграциями в [Twitter](#) 

👋 Последний шанс, прежде чем вы уйдете!

...

Регистрация в DEV занимает *одну минуту*, и это того стоит для вашей карьеры.

Вы получаете в 3 раза больше стоимости, входя в систему, а не скрываясь

Начало работы

Лучшие комментарии (0)

[Кодекс поведения](#) • [Сообщить о нарушении](#)



dchow@goldenmothchemic...
ID: 7036



Chrome
Version: 80.0.3987



Mac OS X
Version: 10.15.4

TAGS

| | | | | | | | |
|-------------|--|--------------|------------------|---------|----------------|---------------|------------|
| browser | Chrome 80.0.3987 | browser.name | Chrome | device | Mac | device.family | Mac |
| environment | prod | handled | yes | level | error | mechanism | generic |
| os | Mac OS X | os | Mac OS X 10.15.4 | release | 41c74c2ea9f2 ⓘ | transaction | /checkout/ |
| url | https://empowerplant.io/shop/checkout/ ⓘ | user | id:7036 | | | | |

EXCEPTION (most recent call first)

Full Raw

ReferenceError

getCardInfo is not defined

| | | | |
|-----------|---------|---------|------|
| mechanism | generic | handled | true |
|-----------|---------|---------|------|

JS

./app/components/Form.jsx in onSubmit at line 13:4

```
09.   let { cart } = this.form;
10.
11.   cart = {
12.     ...cart,
13.     creditCard: getCardInfo()
14.   };
15.
16.   App.confirmPurchase({ cart }).then(({ invoice }) => {
17.     this.onPurchaseComplete({ invoice });
19.   }
```

./app/components/Checkout.jsx in onFormSubmit at line 126:9

./app/components/EmpowerPlant.jsx in confirmPurchase at line 94:40

Next.js Мониторинг ошибок ⚡

Смотрите ошибку и Next.js трассировка стека, ранее видимая только в консоли отладки вашего пользователя. Автоматически применяйте исходные карты для преобразования уменьшенного, скомпилированного или перенесенного кода обратно в его исходную форму. Сохраняйте конфиденциальность своих Next.js исходных карт, загружая их непосредственно на Sentry.

[Попробуйте Sentry](#)



Андреас Оффенхайзер

Днем я разрабатываю и внедряю подключенные сервисы, а в свободное время возлюсь с забавными штуками из операционной системы. Пытаюсь помочь людям поиграть с NodeJS и освоить программирование.

РАСПОЛОЖЕНИЕ

Штутгарт, Германия

ОБРАЗОВАНИЕ

Английский.

РАБОТА

разработчик решений Robert Bosch GmbH

ПРИСОЕДИНИЛСЯ

13 мар. 2017 г.

Подробнее от Андреаса Оффенхайзера

Пять основных изменений при переходе на vuetify 2.1

[#vue](#) [#typescript](#) [#webdev](#) [#javascript](#)

Предварительный просмотр AsciiDoc со встроенным PlantUML в VS Code

[#производительность](#) [#учебное пособие](#) [#документация](#) [#webdev](#)

RBAC с помощью Google Firestore

[#firebase](#) [#бессерверный](#) [#webdev](#)

Сообщество разработчиков

...



Become a Moderator

[Ознакомьтесь с этим опросом](#) и помогите нам модерировать наше сообщество, став модератором тегов здесь, на DEV.
