Куда класть исходники

чтобы потом не было мучительно больно

Что сейчас будет?

Время	45 минут
Ведущий	Григорий Петров
Специализация	Руководство разработкой
Чем занимается	Частный консультант
Опыт	Более 15 лет
Вопросы	В конце вебинара

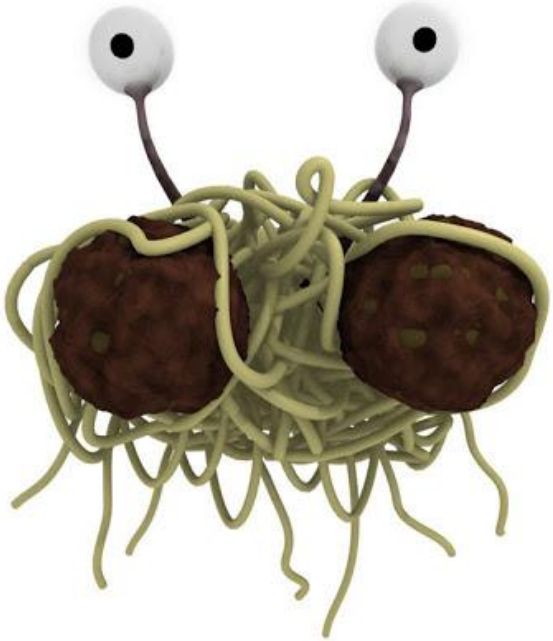
ОСТОРОЖНО: много англицизмов

В начале программа маленькая

В ней меньше тысячи строк кода, несколько файлов, и ничто не предвещает проблем.

Через год она становится большой

В ней уже десятки, если не повезет - сотни тысяч строк кода и много-много файлов. Но пока оно все еще лежит в одном месте и выглядит безвредно...



А затем программ становится две

- Второй сайт "на базе" первого.
- Утилита, использующая часть функциональности.

И возникает вопрос

А что мы будем делать с кодом, который присутствует в обоих програмах?

Простое решение

А давайте сделаем папку 'common'!

Простое решение

А давайте сделаем папку 'common'!

Ну и заодно папки 'logger', 'database', 'jqueryex', 'billing', 'partners', 'ad', ...

Это "монорепозиторий"

Для того, чтобы он возник, ничего особенного делать не надо - достаточно оставить новый проект в той же VCS что и старый, выделив общий код в отдельные директории.

Это "монорепозиторий"

Для того, чтобы он возник, ничего особенного делать не надо - достаточно оставить новый проект в той же VCS что и старый, выделив общий код в отдельные директории.

У монорепозитория есть плюсы и минусы.

Прост в использовании

Прост в использовании

- Не требует времени и усилий для использования.

Прост в использовании

- Не требует времени и усилий для использования.
- Не приходится объяснять новым сорудника что такое distribute и как именно змеи взамодействуют с pth файлами.

Прост в использовании

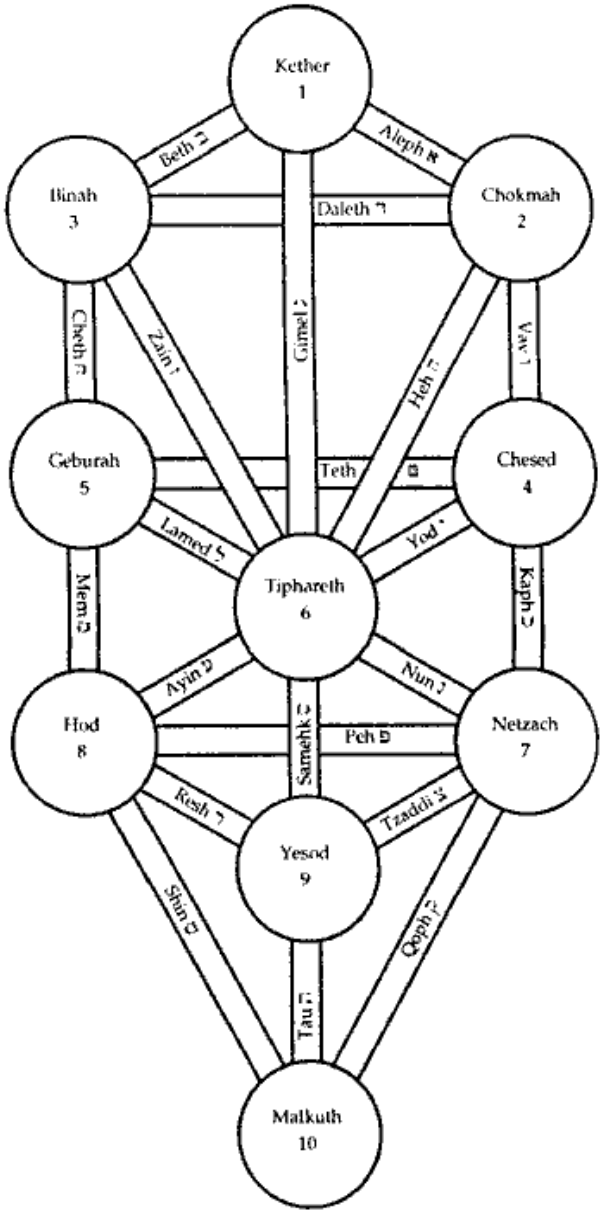
- Не требует времени и усилий для использования.
- Не приходится объяснять новым сорудника что такое distribute и как именно змеи взамодействуют с pth файлами.
- Часто достается вместе с legacy проектами.

Сквозные метки

Если у нас один репозиторий и мы выпускаем релиз "2.8.1463", то все, что нам нужно, - это установить одну метку в этом репозитории.

древо сефирот

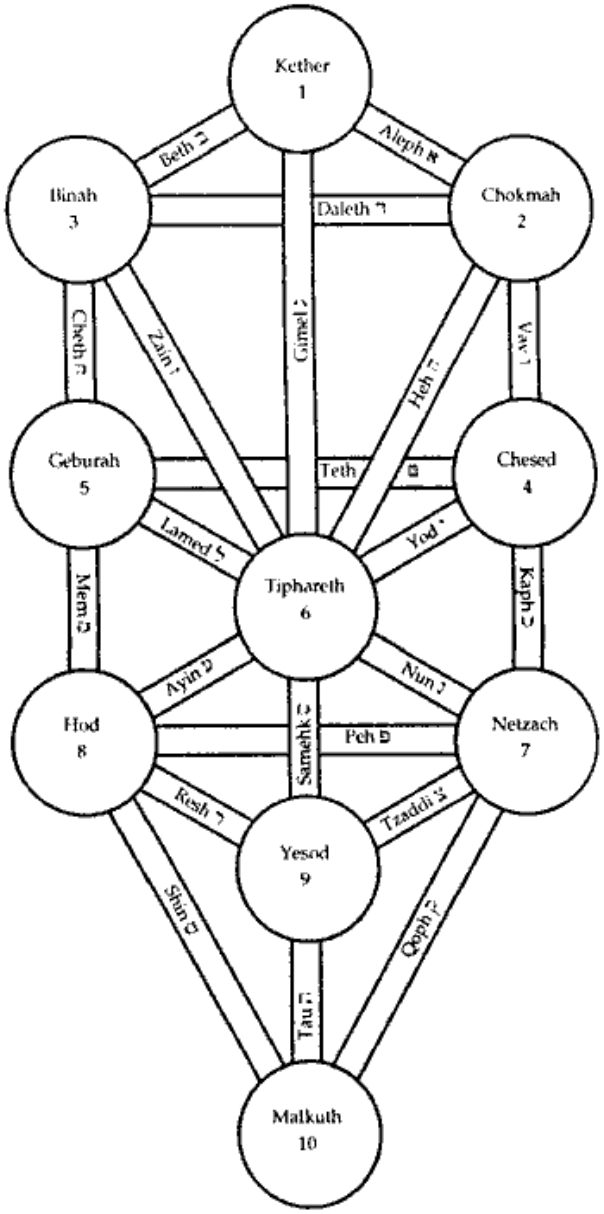
Все связано со всем.



древо сефирот

Все связано со всем.

Поменяли интерфейс в common - у нас не собрались все проекты. Вообще все.



связи не видно

Приходит Вася и говорит: "а давайте я библиотеку 'foo' переделаю!". А мы смотрим на эту библиотеку - и не видим какие именно из наших проектов ее используют. И так **каждый** раз.

соблазн приварить contrib

Если у нас все наши модули лежат в одном репозитории - то возникает огромный соблазн в этот же репозиторий положить нужную нам версию чужой библиотеки.

соблазн приварить contrib

Если у нас все наши модули лежат в одном репозитории - то возникает огромный соблазн в этот же репозиторий положить нужную нам версию чужой библиотеки.

После чего неявные зависимости от нее расползутся по коду, и сделать ее отключаемой будет чудовищно сложной задачей.

проблемы разделеня доступа

Все существующие VCS рассчитаны на установку прав для репозитория целиком. Поэтому когда в репозитории у нас лежат несколько проектов и нужно для них установить разные права - мы получим маленький филиал ада и маленькое транспортное средство проблем.

сложно отделить проект

"Мы бы хотели отдать это в open source/подрядчику/продать вам исходники - но у нас **все** наши проекты лежат в одном репозитории и зависят друг от друга"

Что же делать?

Альтернатива - мультирепозиторий, когда под каждый проект или библиотеку выделяется свой отдельный репозиторий.

Мультирепозиторий - это сложно

Мультирепозиторий - это сложно

- "Директории" в монорепозитории становятся "программами" и "библиотеками" в своих собственных репозиториях - нужно изучать как это работает.

Мультирепозиторий - это сложно

- "Директории" в монорепозитории становятся "программами" и "библиотеками" в своих собственных репозиториях - нужно изучать как это работает.
- Пропадает возможность "взять последнюю версию всего".

Мультирепозиторий - это сложно

- "Директории" в монорепозитории становятся "программами" и "библиотеками" в своих собственных репозиториях - нужно изучать как это работает.
- Пропадает возможность "взять последнюю версию всего".
- Deploy перестает быть "просто скопировать папку на сервер".

Мультирепозиторий - это сложно

- "Директории" в монорепозитории становятся "программами" и "библиотеками" в своих собственных репозиториях - нужно изучать как это работает.
- Пропадает возможность "взять последнюю версию всего".
- Deploy перестает быть "просто скопировать папку на сервер".
- И многое другое ...

Зависимости от версий

Программы зависят от определенных стабильных версий библиотек в репозиториях. Мы можем смело менять общую библиотеку для одной программы - остальные не сломаются.

Явные зависимости

С помощью простейших инструментов мы можем быстро определить какие программы какой общий код используют.

Это позволяет принимать решения о трудоемкости новых фичей и рефакторинга.

Слабо зависимый contrib

Если мы копируем сторонний код в монорепозиторий - у разработчика огромный соблазн вносить в него изменения и намертво связывать со своим кодом. Если сторонний код берется из определенной ревизии своего репозитория, то такого соблазна нет.

Управление доступом

Разделение программ и их частей по отдельным репозиториям позволяет легко ограничивать доступ к критичным участкам кода и, наоборот - максимально открывать некритичные, не боясь открыть "что-нибудь не то".

Приятные мелочи

Приятные мелочи

- Легко отдать какую-нибудь часть в open source, если у бизнеса возникла такая потребность

Приятные мелочи

- Легко отдать какую-нибудь часть в open source, если у бизнеса возникла такая потребность
- Для работы можно клонировать себе только нужные репозитории, а не все пятьдесят гигабайт кодовой базы с историей и бинарниками.

Приятные мелочи

- Легко отдать какую-нибудь часть в open source, если у бизнеса возникла такая потребность
- Для работы можно клонировать себе только нужные репозитории, а не все пятьдесят гигабайт кодовой базы с историей и бинарниками.
- Бинарники наконец-то лежат отдельно. Совсем отдельно. И это никому не мешает.

... и крупные неприятности

Серебряные пули опять не завезли.

Непростое управление

Непростое управление

- Необходимость использовать virtualenv

Непростое управление

- Необходимость использовать virtualenv
- setup.py develop вручную

Непростое управление

- Необходимость использовать virtualenv
- setup.py develop вручную
- deprecated **dependency_links** для внутренних библиотек

Непростое управление

- Необходимость использовать virtualenv
- setup.py develop вручную
- deprecated **dependency_links** для внутренних библиотек
- а это все еще надо развертывать

Непростое управление

- Необходимость использовать virtualenv
- setup.py develop вручную
- deprecated **dependency_links** для внутренних библиотек
- а это все еще надо развертывать
- а еще есть IDE

Метки

Нельзя просто взять (с) и "Поставить метку релиза 2.3.654", а через полгода по этой метке восстановиться. Требуется соблюдать соглашения о ручном управлении зависимостями и создавать собственные механизмы получения и деплоя программы с зависимостями.

Эволюция

От одноклеточных файлов - к экосистеме программ, библиотек и фреймворков.

Это все :)

Теперь можно задавать вопросы.

Рассказывал и показывал Григорий Петров
grigory.v.p@gmail.com
facebook.com/grigoryvp

Френдитесь