Экосистема open source проектов

Полезные инструменты и сервисы

Юрий Литвинов
yurii.litvinov@gmail.com

06.02.2019г

Комментарии по домашке

- Надо закрывать потоки в тестах
- Папку .idea можно не выкладывать
- Можно инициализировать поля прямо в месте их объявления
- Надо ли закрывать стрим в serialize/deserialize?
- Аннотации @NotNull, @Nullable должны работать и при консольной сборке
  - https://github.com/JetBrains/gradle-intellij-plugin
  - intellij-annotations-instrumenter-maven-plugin

Новые штрафы

- Неиспользование nullability-аннотаций там, где они явно полезны — -0.5 балла
- Не static вложенные классы не по делу — -0.5 балла
- Отсутствие комментария ко всему классу, интерфейсу, enum-у и т.д. — -0.5 балла

Continuous Integration

Непрерывная интеграция — практика слияния всех изменений по нескольку раз в день, сборки их в известном окружении и запуска юнит-тестов.

- Автоматический билд
  - Всё, что нужно для сборки, есть в репозитории, может быть получено на чистую (ну, практически) машину и собрано одной консольной командой
- Большое количество юнит-тестов, запускаемых автоматически
- Выделенная машина, слушающая репозиторий и выполняющая билд
  - Чаще всего каждый билд запускается на заранее настроенной виртуалке

Continuous Integration

- Извещение всех разработчиков о статусе
  - Если билд не прошёл, разработка приостанавливается до его починки
- Автоматическое выкладывание
- Пока билд не прошёл, задача не считается сделанной
  - Короткие билды (<10 мин.)
  - deployment pipeline
    - Отдельная машина для сборки, для коротких тестов, для длинных тестов, для выкладывания

Travis

- https://travis-ci.org/ — пример бесплатной для open source-проектов облачной CI-системы
- Собирает на чистой виртуальной машине под Ubuntu 12.04 (или Ubuntu 14.04) или OS X (есть экспериментальная поддержка Windows)
- Интегрируется с GitHub-ом, Slack-ом, умеет деплоить
- Окружение настраивается конфигурационным файлом или “вручную” из скрипта сборки (некоторые конфигурации разрешают sudo)
- Сборка выполняется либо автоматически, либо указанным скриптом сборки
- Build Matrix
  - Разные конфигурации сборки, выполняемые на разных виртуальных машинах

Travis, настройка сборки

- Установить commit hook на гитхабе
  - Travis умеет это делать сам, надо залогиниться под своим GitHub-аккаунтом на Travis и выбрать нужный репозиторий в профиле
- Добавить .travis.yml в корень репозитория
  - Это конфигурационный файл, говорящий, что и как собирать
- Закоммитить и запушить, это инициирует процесс сборки
  - Коммит, где в комментарии есть подстрока “[ci skip]”, игнорируется Travis-ом, остальные он собирает
- Результаты будут видны у каждого коммита в истории и в пуллреквесте

Travis, конфигурационный файл

language: java

— всё :) Если у вас проект прямо в корне репозитория.
Жизненный цикл сборки:

- Install apt addons
- before_install
- install
- before_script
- script
- after_success или after_failure
- [OPTIONAL] before_deploy
- [OPTIONAL] deploy
- [OPTIONAL] after_deploy
- after_script

Travis, примеры

- Веб-приложение из нескольких сервисов на Java:
  - https://github.com/qreal/wmp/blob/master/.travis.yml
- Относительно большое десктопное приложение на C++:
  - https://github.com/qreal/qreal/blob/master/.travis.yml
- Билд в Docker-окружении (все зависимости носим с собой):
  - https://github.com/trikset/trikRuntime/blob/master/.travis.yml

Travis, конфиг для домашки

```
language: java
os:
  - linux
env:
  - PROJECT_DIR=hw1
  - PROJECT_DIR=hw2
  - PROJECT_DIR=hw3
script: cd $PROJECT_DIR && ./gradlew check
```

AppVeyor

- CI с поддержкой Windows и Linux, прежде всего для сборки .NET-приложений, но умеет много чего ещё (в т.ч. Java)
  - Windows Server 2016 или Windows Server 2012 R2
  - Ubuntu 16.04.4 LTS или Ubuntu 18.04 LTS
- Тоже бесплатен для open source
- Конфигурируется через appveyor.yml
- Собирает по умолчанию MSBuild
  - Можно переубедить
- Синтаксис .yml-файла: https://www.appveyor.com/docs/appveyor-yml/
- Пример: https://github.com/qreal/qreal/blob/master/appveyor.yml
  - Собирать двумя CI-серверами один проект не зазорно

Анализ тестового покрытия, CodeCov

- https://codecov.io/
- Визуализатор для функциональности компиляторов или специальных инструментов по слежению за исполнявшимися строчками
- Чем больше операторов было исполнено во время тестового прогона, тем меньше вероятность пропустить баг
  - 100% покрытие не гарантирует работоспособность программы
- Для Java (Kotlin, Scala) используется библиотека JaCoCo, для C++/gcc — ключ “-coverage”
- Интегрируется с гитхабом (комментит пуллреквесты информацией о тестовом покрытии)

CodeCov, пример конфигурации

Travis:

language: java

after_success:
  - bash <(curl -s https://codecov.io/bash)

Пример для Android-приложения:

- https://github.com/codecov/example-android/blob/master/.travis.yml

Статический анализ, Codacy

- https://www.codacy.com/
- Ищет типичные ошибки: потенциальные баги, стайлгайд, мёртвый код, производительность и т.д.
- Поддерживает много языков (в том числе C#, C++, Java, Kotlin, Python, Scala)
- Не требует дополнительных манипуляций с репозиторием
- Очень настраиваема

Инструменты планирования, Trello

- https://trello.com/
- Интерактивная доска с карточками, организованными в списки
- Карточки легко редактируются и перетаскиваются между списками
  - Типичные списки: TODO, In Progress, Done (возможны варианты)
- Поддерживает дедлайны, чеклисты, вложения, комментарии, голосования, метки
- Легковесный инструмент планирования, подходящий, тем не менее, и для больших проектов

Инструменты планирования, Pivotal Tracker

- https://www.pivotaltracker.com
- Более “тяжеловесный” инструмент, ориентированный на Scrum
- Всего три списка
  - Icebox — что было бы неплохо сделать
  - Backlog — запланированные задачи
  - Current — задачи на текущую итерацию
- Задачи можно оценивать, задачи имеют тип и статус
  - По оценкам задач и статистике работы команды считается team velocity, позволяющая предсказать линейные сроки
- Есть релизы с дедлайнами, метки, epic-и, чеклисты, вложения, комментарии
- Умеет считать статистику, рисовать графики (burndown charts)

Средства коммуникации, Slack и Gitter

- Instant messenger-ы, ориентированные на команды и интегрированные со средствами разработки
  - Информация о коммитах и пуллреквестах
  - Статус CI
  - Другие тулы
- Синтаксическая подсветка (markdown), вложения, отображение картинок, ...
- Gitter интегрирован с гитхабом и “более открыт” (предназначается прежде всего для общения сообщества)
- Slack интегрирован с чем угодно, предназначается прежде всего для общения внутри команды

GitHub: Issues, Projects, Wiki, Pages

- GitHub сам многое умеет
- Issues — довольно удобный багтрекер
  - Майлстоуны, дедлайны, метки на багах, возможность закрывать баги автоматически (если в сообщении коммита есть “close” или “fix” и #<номер бага>)
  - Пуллреквест тоже считается Issue
- Projects — представляет Issues в виде набора списков, между которыми их можно перетаскивать в духе Trello
- Wiki — викистраницы, куда можно выкладывать полезную информацию о проекте
  - Тоже git-репозиторий
- Pages — хостинг для статических сайтов <имя проекта>.github.io

Авторское право

- Open source-кодом можно пользоваться, только если автор явно это разрешил, так что просто код на GitHub — не совсем open source
- Бывают исключительные и личные неимущественные права
  - Личные неимущественные права неотчуждаемы
  - Исключительные права можно передать
  - Права появляются в момент создания произведения и принадлежат автору
    - Если произведение создано по служебному заданию — работодателю
    - Знак копирайта служит только для информирования, регистрация прав не требуется
  - Соавторы владеют произведением в равной степени
- Идея не охраняется, охраняется её физическое выражение

Open source-лицензии

- Лицензия — способ передачи части прав на произведение
- Пример — “Do what the **** you want to public license”
  - “Want to” может включать в себя патентование произведения и подачу в суд на автора за нарушение патента, поэтому обычно лицензии более длинны и унылы
  - В России и Европе программы не патентуют, в США — да
- Каждый нормальный open source-проект должен иметь лицензию

Open source-лицензии

- Часто используемые open source-лицензии:
  - GPL, LGPL (GPL вирусная, поэтому использовать её, внезапно, плохая практика)
  - MIT License
  - Apache License 2.0 (может применяться пофайлово)
  - BSD License (в разных вариантах)
  - The Unlicense — явная передача произведения в Public Domain
  - Семейство лицензий Creative Commons — не для софта, но хорошо подходит для ресурсов (картинок, текстов и т.д.)