Lecture 03/16:
Fundamental parts of open-source projects

Jeongkyu Shin
Sep 28, 2016

Development procedure

From basement

Project type

- The *“lone wolf”*
  - Easily found at many open-source projects
  - Small / The ‘power of one’ effect
  - Becomes team when it is needed for crowds
- OSS team
  - Most OSS projects are
  - Consists of various positions
- Enterprise-based OSS
  - Well-structured codebase
  - Fixed tech stacks

The importance of OSS topic

- Hobby → Response → Digging →Community →Ecosphere
- Personal story #1 (2004~2008)

Development positions

So many positions for the goal

Tech stacks

- Backbone / server developer
- Front-end developer
- App developer
  - Android / iOS / ETC
- Designer
- Full-stack developer?

Backbone / server developer (1)

- Very important when you want to make opensource-based ‘service’
- When front-end developer also develops backbone/server:
  - Before scaling
  - Better for prototyping
  - Will need *refactoring*
- Database master
  - Independent position when developing server-side

Backbone / server developer (2)

- Server-side development language
  - Korea: Java+Spring
  - Start-up companies
    - Uses various stacks
    - Node.js / Python / Go
    - RoR / Scala (sometimes)

Front-end developer (1)

- Develops “viewable” part
- Need understanding about ‘designing’
- Platform-dependency
  - “Web” or “App” ?
  - What devices?
- Also server-side?
  - When the project is based on APM / NPM

Front-end developer (2)

- Front-end development languages
  - Not the much far from server-side
  - Web (HTML5 / ES6)
    - jQuery / Bootstrap
    - Angular.js / backbone.js (for SPA)
    - React / React Native / Polymer

App developer (1)

- Mobile frontend development
- Can be with front-end developer
  - Depends on the framework
    - React Native / Polymer / Cordova
- Considerations
  - Applications with lightening-speed responses?
  - Applications those need the real-time communications?

App developer (2)

- Needs understanding about native language of app platform
  - Android: Java+Android framework
  - iOS: Objective-C + Swift
  - Windows / macOS / Linux
    - Various languages
- Cross-platform?
  - Java
  - Binding : QT / GTK / Mono?

The *'team'*

- (Frontend+backend) developer + app developer
- When the project grows:
  - Divide frontend and backend development positions
  - App developer -> platform-specific developer
  - Designer
    - Artworks

Structure and architecture

After gathering building blocks

Interests / tech stacks

- OSS: from the language I can
  - Of course it usually starts from author's need
- Stand-alone or service application?
- Desktop or mobile application?
- Local or network application?

DevOps

- Development + Operation
  - Usually it is separated
    - Customer / operation part
    - Development part
  - Faster tech stack changes lead startups into DevOps
- OSS-based projects choose DevOps
  - Team size effect
  - Ultra-fast develop / production cycle

Collaboration tools (real-time)

- IRC
  - Old but still working method
  - Servers / channels
- Slack (not OSS. It is service)
  - Almost similar to IRC
  - Team-specific chatting channels
  - Many IRC channels support *“relay”* between IRC and Slack

Test suite

- Automatically check / test codes
- Language-specific test suites
- Travis CI
  - Easy: Add configuration to the project on Github
  - Free: for OSS projects
  - You will need it for your CI

Documentation

The most important part to be *‘usable software’*

Documentation from source code

- JavaDoc (http://www.oracle.com/technetwork/articles/java/index-137868.html)
  - Tool for generating API documentation in HTML format from doc comments in source code
  - Traditional and strong / can use with other programming languages
- Doxygen
  - The *de facto standard* tool for generating documentation from annotated C++ sources,
- Sphinx
  - Tool that makes it easy to create intelligent and beautiful documentation
  - reST (reStructuredText) (http://docutils.sourceforge.net/docs/user/rst/quickref.html)
- Docco (http://jashkenas.github.io/docco/)
  - Quick-and-dirty documentation generator

Documentation format

- JavaDoc
  - https://www.tutorialspoint.com/java/java_documentation.htm
- Doxygen
  - http://xerces.apache.org/xerces-c/apiDocs-3/classes.html
- reStructuredText
  - http://docutils.sourceforge.net/docs/user/rst/quickref.html
- Docco
  - https://davidwalsh.name/javascript-documentation
- Only for docs: Markdown (https://daringfireball.net/projects/markdown/)
  - Easy **text-to-HTML** conversion tool/specification for web writers
  - Many parser exists

Learn by run today:

Learn by run today

- Discuss about your team objective and project on Slack
  - Finalize your project name / goal
  - Fix the dev. position. (Of course you can be a full-stack developer! Why not?)
  - Set the rule
- Create team github project page
  - With language setup / license / initalization
- Create Jekyll github wiki page for project

Learn by run today

- Write the plan about the project
  - Name
  - Purpose
  - Goal
  - Milestone
- Add *'link to github project page'* to your personal github page

Assignment #1

- Explore 10 or more github projects related to your project
- Get the statistics
  - Tech stacks (language, database, OS, environment) / members / code length / contributions / ETC.
- Summarize the result statistics using LibreOffice Calc
- Send results to Teaching Assistant using Slack message
- Due date: Oct. 4 00:00 (Oct. 3 24:00)

Next is...

4/16: OSS License issues

@inureyes
Questions? inureyes@gmail.com
OR
https://www.codeonweb.com/circle/@oss-basics-hu