From Zero to Open Source Hero

Contributing to Spring projects

Agenda

Vedran Pavić

- Software Development Engineer at Kapsch CarrierCom d.o.o.
- Open Source Software enthusiast
  - Java, Spring, Linux
- Active contributor to multiple Spring projects
- Spring Session committer

What it means to contribute?

There’s more to contributing than just code

- Helping other users matters
  - Issue tracker, Gitter, Stack Overflow
- Reporting issues matters
  - stackoverflow.com/help/mcve
- Documentation matters - a LOT

What are the prerequisites?

- Knowledge of Git, related workflows and GitHub
- Willingness to discuss, elaborate and rework your proposals
- Contributor License Agreement (CLA)
  - cla.pivotal.io
- Patience :)

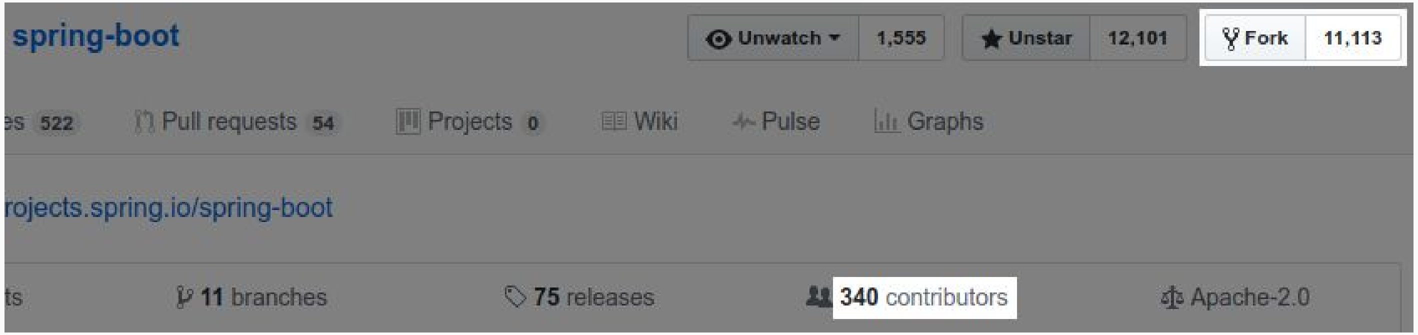
Why contribute?

Spring embraces your contributions

- Move to GitHub made contributing much easier
- Projects are well managed with contributors in mind
  - Easy to build, easy to import in IDE
- Contributions are properly attributed
  - Commits, @author tags

Spring embraces your contributions

- Numbers are also telling:



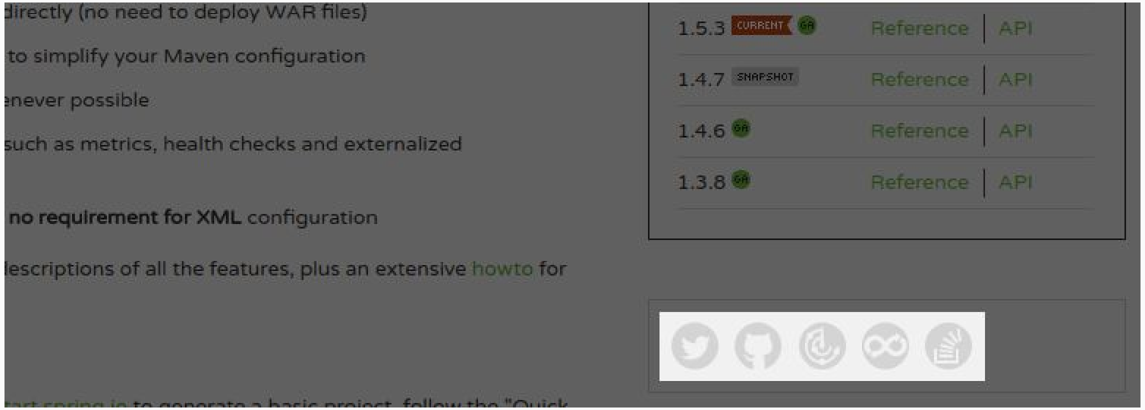
What do you get out of it?

- Learn new skills, or enhance existing ones
    - Apply the ideas from Spring projects to your own projects
- Meet the people behind Spring and collaborate with them
- Grow your reputation
- Contributing is an empowering experience

Where to start?

Use spring.io as *service discovery*

- spring.io/projects contains pointers to all relevant project's resources
  - issue tracker, source repository, CI server, Stack Overflow tag



directly (no need to deploy WAR files)

to simplify your Maven configuration

enever possible

such as metrics, health checks and externalized

no requirement for XML configuration

lescriptions of all the features, plus an extensive howto for

1.5.3 CURRENT GA	Reference	API
1.4.7 SNAPSHOT	Reference	API
1.4.6 GA	Reference	API
1.3.8 GA	Reference	API

Get familiar with the project

- Source repositories contain resources for contributors
    - README, CONTRIBUTING, CODE_OF_CONDUCT
- Note the project's active branches
- Check out the issues marked for contribution
    - JIRA roadmap, GitHub labels

Get familiar with the project

- Project build: Gradle or Maven
- Single-click builds that are easy on the newcomers
  - As simple as `./gradlew build` or `./mvnw clean install`
- Check out resources for contributors for more details
  - Some projects have special build profiles, for example documentation builds

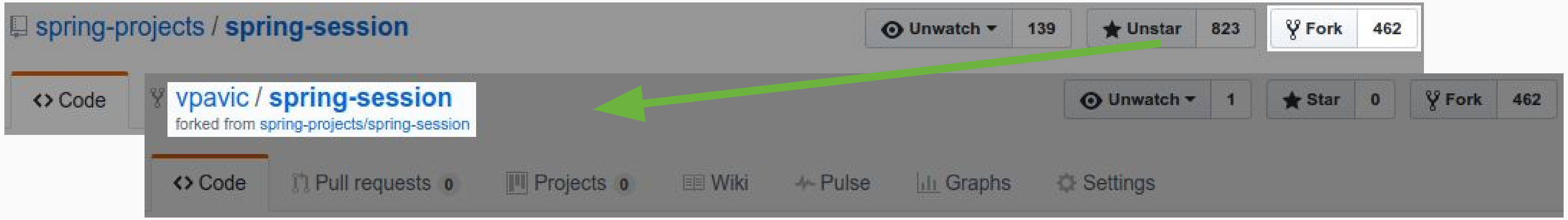
Get familiar with the project

- Note the preferred Git workflows
  - Merge vs rebase
- Note the preferred code style
  - Check source repository for IDE config files
- Use other people’s contributions as a reference
- Reach out to the project maintainers or community
  - Gitter or Stack Overflow

Managing your forks

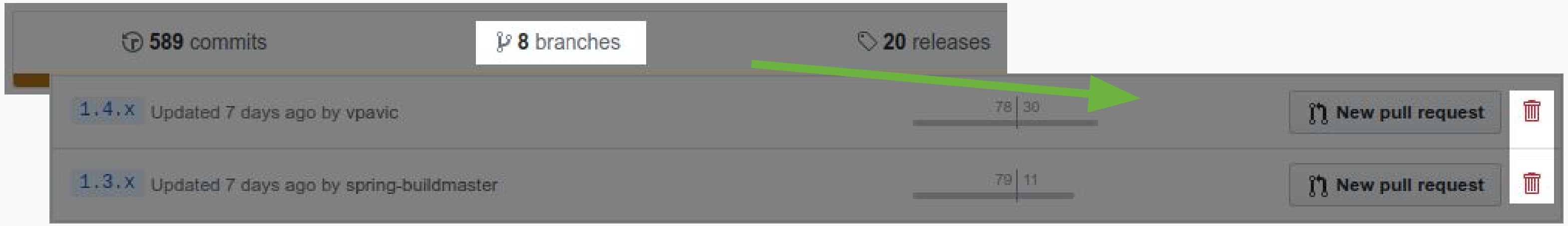
Creating a fork

- A fork is a copy of a repository
- Serves as a base for contributing activities



Keep your forks lean

- Forking creates a copy with all branches of the original repository
  - Some of them are not active, or not relevant for contributor
- Deleting needless branches makes your fork easier to maintain



Keep your forks up to date

- Configure a remote that points to original repository

```
$ git remote add upstream git@github.com:spring-projects/spring-session.git

$ git remote -v
origin     git@github.com:vpavic/spring-session.git (fetch)
origin     git@github.com:vpavic/spring-session.git (push)
upstream   git@github.com:spring-projects/spring-session.git (fetch)
upstream   git@github.com:spring-projects/spring-session.git (push)
```

Keep your forks up to date

- Fetch and merge the changes from the upstream repository

```
$ git fetch upstream
...
From github.com:spring-projects/spring-boot
   216506d20f..e236b71615  1.5.x      -> upstream/1.5.x
   3abd8d3adf..269cea291c  master     -> upstream/master

$ git checkout 1.5.x && git merge upstream/1.5.x && git push

$ git checkout master && git merge upstream/master && git push
```

Keep your forks up to date

- Tags need to be handled separately

```
$ git fetch upstream --tags
...
From github.com:spring-projects/spring-boot
 * [new tag]               v1.5.3.RELEASE -> v1.5.3.RELEASE

$ git push --tags
```

Clean up your local branches

- Clean up after deleting branches on GitHub

```
$ git remote prune origin
Pruning origin
URL: git@github.com:spring-projects/spring-integration
 * [pruned] origin/INT-4248

$ git branch -vv | grep gone
  INT-4248 a2458f78f [origin/INT-4248: gone] Use StringRedisTemplate

$ git branch -d INT-4248
```

Add new upstream branches

- As development of the upstream goes on, new branches

```
$ git checkout --track upstream/4.2.x

$ git branch -vv | grep upstream
* 4.2.x  f166bd1bd [upstream/4.2.x] Groovy test: Fix format for `MM` instead of `mm`

$ git push --set-upstream origin/4.2.x
```

Anatomy of a good Pull Request

Before you start

- If the issue ticket exists, drop a note you’re working on it
    - To help prevent duplicating efforts
- Otherwise opening issue might be required
- Pick the appropriate target branch
    - Semantic versioning matters - semver.org
- If in doubt about target branch consult the maintainers

Working on your changes

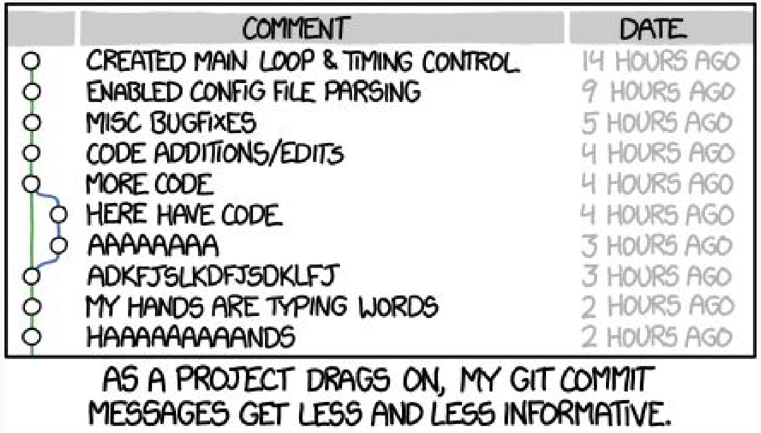
- Configure your IDE to use appropriate code style
    - Most projects contain Eclipse formatter configuration files
    - IntelliJ IDEA users will find *Eclipse Code Formatter* plugin useful
- Create a dedicated feature branch for your changes - use target branch as base
- Initially make your changes a single commit unless there’s a good reason to do otherwise

Tests or it didn’t happen

- Unit tests are a **must** if you change the code
- If you’re fixing a bug add a unit test that reproduces the problem
  - Check out the contributors resources for any policies on unit tests
- If you’re adding a new functionality a substantial set of tests is expected
  - Check the existing unit tests for similar/related functionalities

Write good commit messages

- Try avoiding lazy commit messages :)



	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

xkcd.com/1296

Write good commit messages

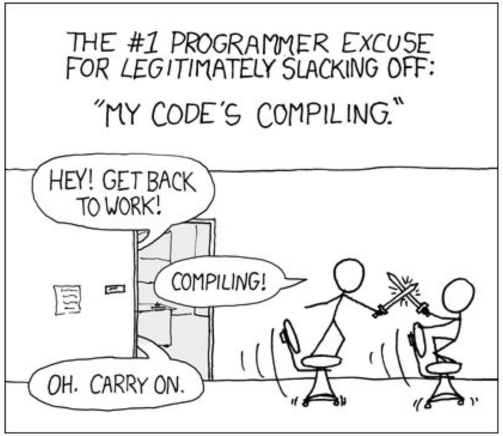
- There are some excellent resources on writing good commit messages
    - chris.beams.io/posts/git-commit
- Good commit message does you a favor when opening the PR
    - Commit message is automatically used for PR description on GitHub

Build the project before submitting PR

- Builds are single-click and easy to get running
    - Check contributor resources for info on additional build profiles, like documentation
- Contains additional checks, such as Checkstyle
    - Remember to import the IDE code style config
    - Use Checkstyle plugin for your IDE to discover errors early
- Tests the impact of your changes on entire project

Build the project before submitting PR

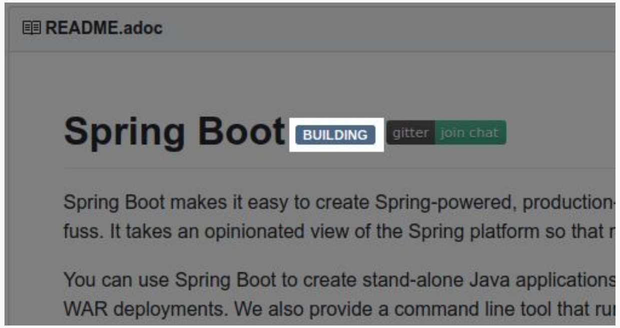
- Full project build takes some time however so you can get creative :)



xkcd.com/303

Build the project before submitting PR

- If the build fails for you for reasons unrelated to your changes check the project’s source repository and/or CI server for info



README.adoc

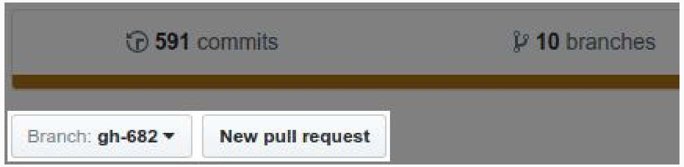
Spring Boot BUILDING gitter join chat

Spring Boot makes it easy to create Spring-powered, production-
fuss. It takes an opinionated view of the Spring platform so that r

You can use Spring Boot to create stand-alone Java applications
WAR deployments. We also provide a command line tool that ru

Submitting the PR

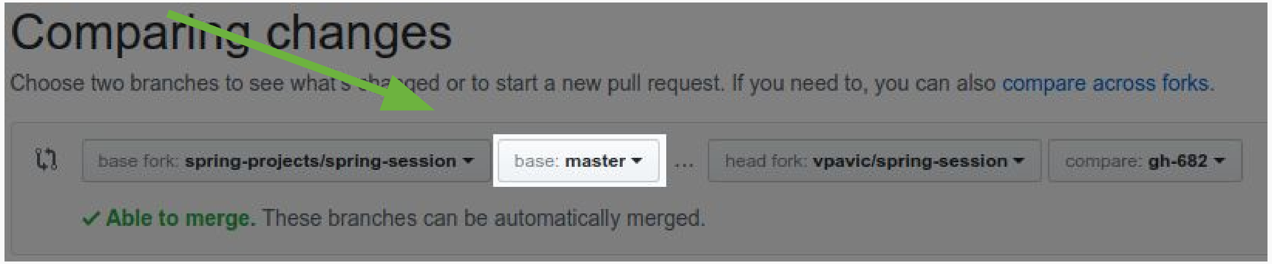
- Remember to select the target branch



591 commits

10 branches

Branch: gh-682



Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks.

base fork: spring-projects/spring-session

base: master

head fork: vpavic/spring-session

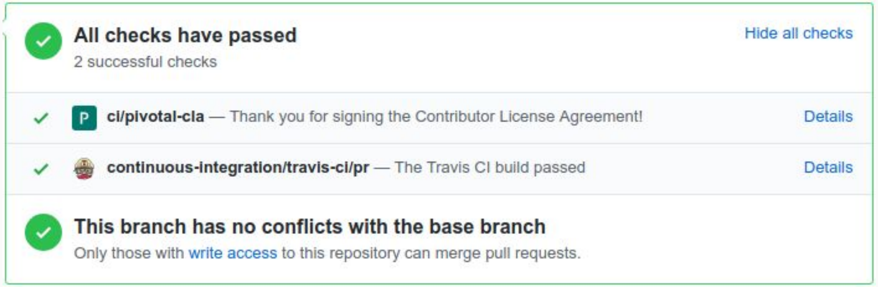
compare: gh-682

✓ Able to merge. These branches can be automatically merged.

Lifecycle a Pull Request

Pull Request checks

- Submitting a PR will usually trigger some actions
  - Contributor License Agreement (CLA) check
  - PR branch build on Travis CI



All checks have passed
2 successful checks

Hide all checks

✓ **ci/pivotal-cla** — Thank you for signing the Contributor License Agreement! Details

✓ **continuous-integration/travis-ci/pr** — The Travis CI build passed Details

This branch has no conflicts with the base branch
Only those with write access to this repository can merge pull requests.

Pull Request checks

- If you’re first time contributor you’ll be asked to sign CLA
  - cla.pivotal.io has all the details
  - The process in nearly automatic these days
- Minor changes (e.g. typos) can skip some checks
  - CLA not required - add “Obvious Fix” to the PR description
  - Skip the Travis CI build - include “[ci skip]” in commit message

Pull Request checks

- Travis CI builds can sometime get stuck or fail for transient reasons
  - You can trigger the build again by closing and reopening the PR
  - Or more elegantly using Git

```
$ git commit --amend --no-edit && git push --force
```

Discussion and reviews

- Expect discussion on your proposals, especially if your PR is introducing new features
- Often times you’ll be asked to rework your proposal
- Don’t open a PR and walk away
  - If unsure how to rework your proposal ask for help
  - If you have no time to rework let the maintainers know

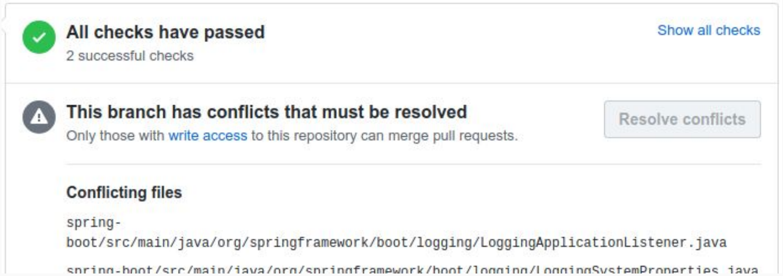
Updating your Pull Request

- Requested changes are done on the existing PR - no need to close existing and open new one
- You can simply push more commits to your PR branch
- You can update the existing commit (force push is needed)

```
$ git add . && git commit --amend --no-edit && git push --force
```

Updating your Pull Request

- While reworking the PR it might be a good idea to rebase your PR branch on the current state of base branch
  - Remember the tips for managing forks
  - This especially matters is your PR has been on the shelf for some time



All checks have passed
2 successful checks

This branch has conflicts that must be resolved
Only those with write access to this repository can merge pull requests.

Conflicting files

```
spring-
boot/src/main/java/org/springframework/boot/logging/LoggingApplicationListener.java
spring-boot/src/main/java/org/springframework/boot/logging/LoggingSystemProperties.java
```

In the end

- You didn’t receive any response - be patient
    - It might get some time for maintainers to get to your PR
- Your contribution was not accepted - don’t get discouraged
    - If you’re active in the open source this will happen sooner or later :)
- Your contribution was accepted - welcome to the club!

Conclusion

Spring ♥ contributions

- Spring and entire ecosystem around it wouldn’t be what it is today without contributors
- Significant efforts have been made to make Spring projects contributor friendly

The time is right to start contributing

- With Spring 5 around the corner there’s a lot of movement across the Spring ecosystem
    - Move to Java 8 as baseline, introduction of reactive programming model
- Most Spring projects are moving to new major release as a consequence
    - Chance to make significant changes

Questions?

Thanks!

@vedran_pavic
github.com/vpavic