Tooling for Java EE applications

PA165

Jiří Uhlíř, Martin Kotala

26. 9. 2017

Contents

Contents

History

- Usnadnění tvorby příkladů pro použití v testech a domácích úkolech
- Generování sad otázek pro odpovědníky

Comparison with other tools

- Jednotná aplikace pro generování příkladů
- Opravy chyb a vylepšení algoritmů existujících generátorů

Comparison with other tools

- Jednotná aplikace pro generování příkladů
- Opravy chyb a vylepšení algoritmů existujících generátorů
- Generátor příkladů pro algoritmus C-Y-K

Comparison with other tools

- Jednotná aplikace pro generování příkladů
- Opravy chyb a vylepšení algoritmů existujících generátorů
- Generátor příkladů pro algoritmus C-Y-K
- Rozšiřitelnost vzniklé aplikace pro další typy příkladů (i mimo okruh formálních jazyků)

Comparison with other tools

- Jednotná aplikace pro generování příkladů
- Opravy chyb a vylepšení algoritmů existujících generátorů
- Generátor příkladů pro algoritmus C-Y-K
- Rozšiřitelnost vzniklé aplikace pro další typy příkladů (i mimo okruh formálních jazyků)
- Odlehčení GUI

Comparison with other tools

- Jednotná aplikace pro generování příkladů
- Opravy chyb a vylepšení algoritmů existujících generátorů
- Generátor příkladů pro algoritmus C-Y-K
- Rozšiřitelnost vzniklé aplikace pro další typy příkladů (i mimo okruh formálních jazyků)
- Odlehčení GUI
- Lokalizace do češtiny a angličtiny

Contents

Cíle práce

- Jednotná aplikace pro generování příkladů
- Opravy chyb a vylepšení algoritmů existujících generátorů

Cíle práce

- Jednotná aplikace pro generování příkladů
- Opravy chyb a vylepšení algoritmů existujících generátorů
- Generátor příkladů pro algoritmus C-Y-K

Cíle práce

- Jednotná aplikace pro generování příkladů
- Opravy chyb a vylepšení algoritmů existujících generátorů
- Generátor příkladů pro algoritmus C-Y-K
- Rozšiřitelnost vzniklé aplikace pro další typy příkladů (i mimo okruh formálních jazyků)

Cíle práce

- Jednotná aplikace pro generování příkladů
- Opravy chyb a vylepšení algoritmů existujících generátorů
- Generátor příkladů pro algoritmus C-Y-K
- Rozšiřitelnost vzniklé aplikace pro další typy příkladů (i mimo okruh formálních jazyků)
- Odlehčení GUI

Cíle práce

- Jednotná aplikace pro generování příkladů
- Opravy chyb a vylepšení algoritmů existujících generátorů
- Generátor příkladů pro algoritmus C-Y-K
- Rozšiřitelnost vzniklé aplikace pro další typy příkladů (i mimo okruh formálních jazyků)
- Odlehčení GUI
- Lokalizace do češtiny a angličtiny

Contents

Cíle práce

- Jednotná aplikace pro generování příkladů
- Opravy chyb a vylepšení algoritmů existujících generátorů

solarwinds

Cíle práce

- Jednotná aplikace pro generování příkladů
- Opravy chyb a vylepšení algoritmů existujících generátorů
- Generátor příkladů pro algoritmus C-Y-K

Cíle práce

- Jednotná aplikace pro generování příkladů
- Opravy chyb a vylepšení algoritmů existujících generátorů
- Generátor příkladů pro algoritmus C-Y-K
- Rozšiřitelnost vzniklé aplikace pro další typy příkladů (i mimo okruh formálních jazyků)

Cíle práce

- Jednotná aplikace pro generování příkladů
- Opravy chyb a vylepšení algoritmů existujících generátorů
- Generátor příkladů pro algoritmus C-Y-K
- Rozšiřitelnost vzniklé aplikace pro další typy příkladů (i mimo okruh formálních jazyků)
- Odlehčení GUI

Cíle práce

- Jednotná aplikace pro generování příkladů
- Opravy chyb a vylepšení algoritmů existujících generátorů
- Generátor příkladů pro algoritmus C-Y-K
- Rozšiřitelnost vzniklé aplikace pro další typy příkladů (i mimo okruh formálních jazyků)
- Odlehčení GUI
- Lokalizace do češtiny a angličtiny

Contents

Git Branching - Overview

- A branch represents an independent line of development.
- Lightweight implementation of branching – Git stores a branch as **a reference to a commit**.
- Keeps history as a tree, where **each commit is a node** in the tree, and has one or more parents.
- History is **extrapolated through the commit relationships**.
- It's a good practice to **spawn a new branch to encapsulate your changes** no matter how big the changes are.

Git Branching - Local Branches

- **Non-tracking local branches**
  - Exist on user's machine.
  - Not associated with any other branch.
  - User needs to specify which upstream branch when running push or pull commands.
- **Tracking local branches**
  - Exist on user's machine.
  - Tracking branch is a branch that has a direct relationship to another branch.
  - Local tracking branches in most cases track a remote tracking branch.
  - Allow user to run git pull and git push without specifying which upstream branch to use.

Git Branching - remote-tracking branches

- **Remote**
  - Remote connection (bookmark) into other repository.
- **Remote branch**
  - Branch on a remote location.
- **Remote-tracking branch**
  - Local cache for what the remote repositories contain.
  - (remote)/(branch)
    - origin/master
    - origin/test-branch
- **Note:**
  - “origin” and “master” are not special.

Git Branching - merge

- Way of putting a forked **history back together** again.
- **Non-destructive** operation.
- All the operations always **merge into the current** branch.
- Git has **several distinct algorithms** to accomplish the merge.

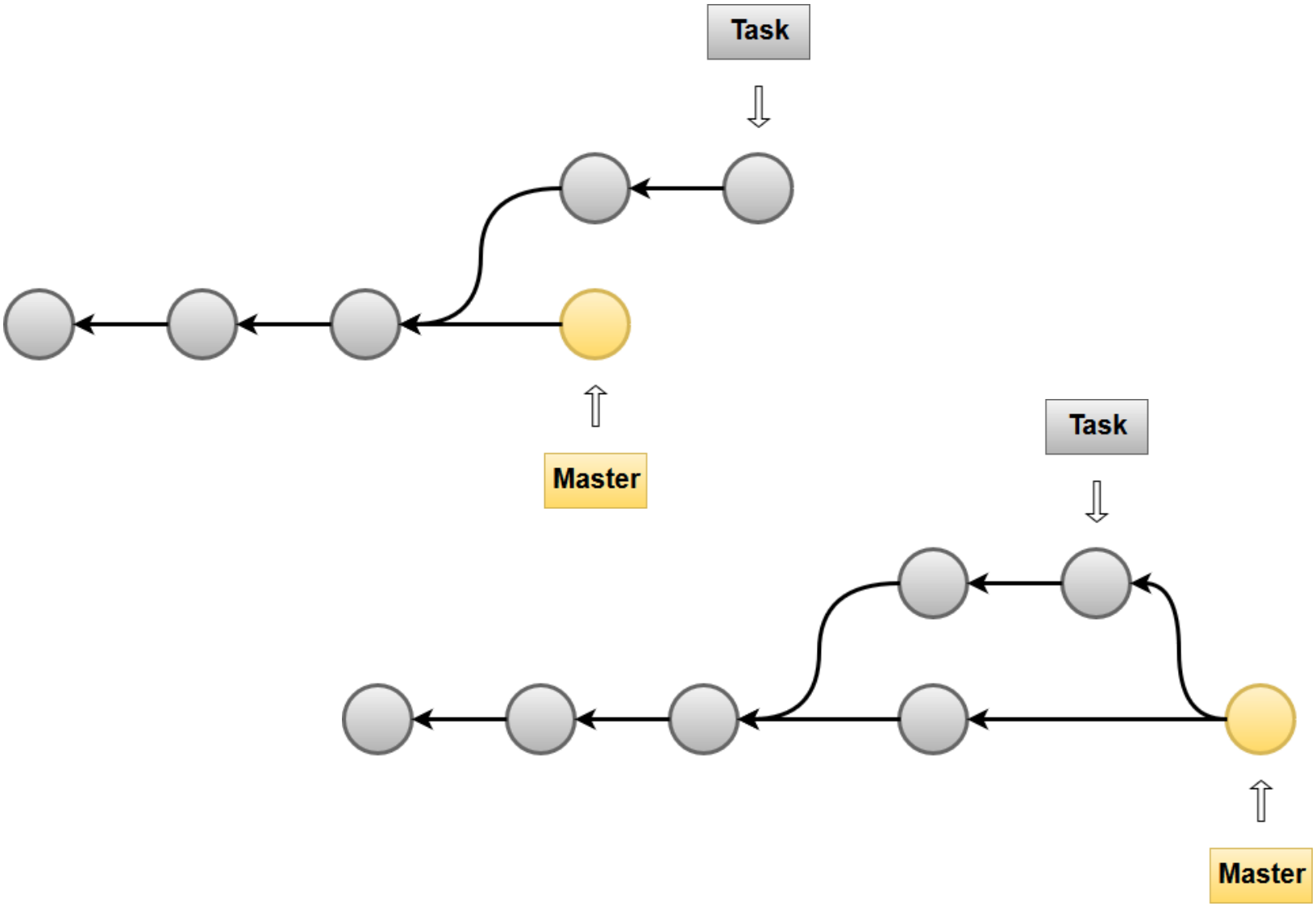
Note:

- git pull command effectively runs git fetch and git merge.

Git Branching - merge

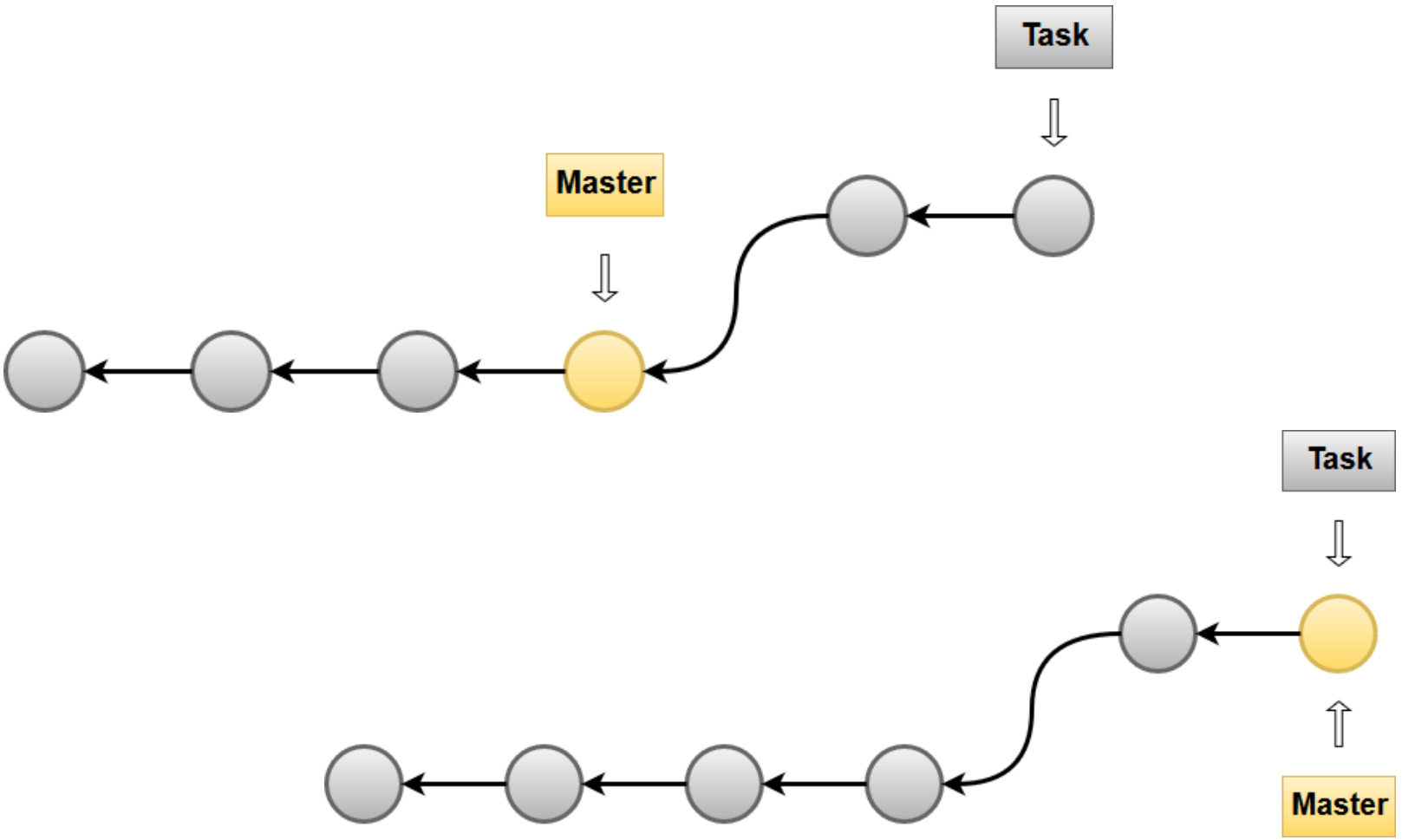
- **3-Way Merge**
  - Creates **merge commit** that ties together the histories of both branches.
  - Merge commit as a **symbolic joining** of the two branches.
  - Original **context is maintained**.
- **Fast-Forward Merge**
  - Requires **linear path** from the current branch tip to the target branch.
  - Usually **facilitated through rebasing** – suitable for small tasks and fixes.
  - Context of the affected commits as part of an earlier feature branch is lost.

Git Branching - 3-way merge



solarwinds

Git Branching - fast-forward merge

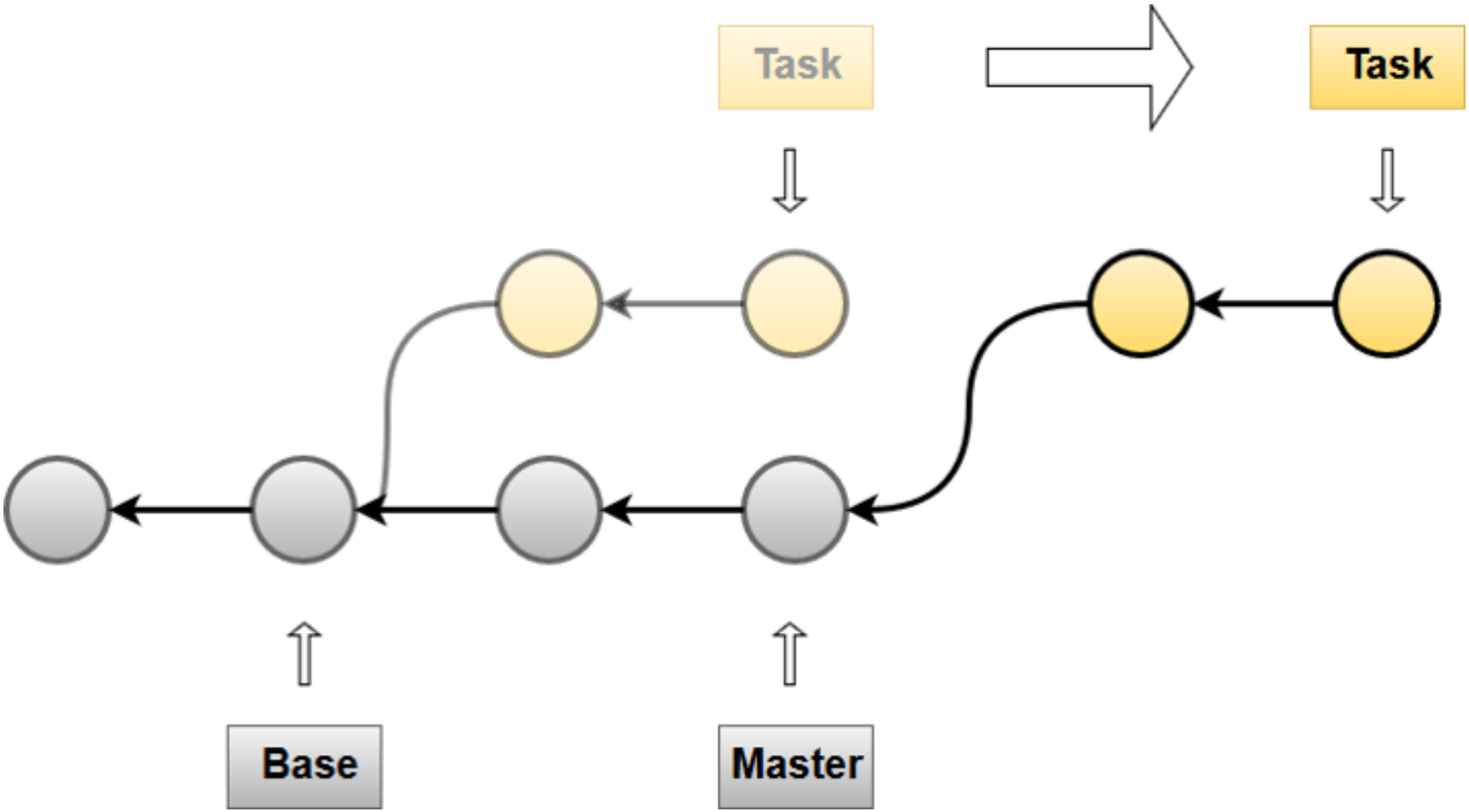


solarwinds

Git Branching - rebase

- Process of **moving or combining** a sequence of commits to a new base commit – alternative to merge.
- Makes the branch appear as if you'd created it from a different commit.
- Git takes changes from your branch and **replays them** on top of the destination branch.
- Result branch **looks the same** but it's composed of entirely new commits.
- Do not rebase commits that exist **outside your repository** (unless you have a good reason to do so).

Git Branching - rebase



solarwinds

Git Branching - rebase

- Rebase helps to maintain a **linear project history** – that allows an easier investigation of regression issues.
- **Workflow example:**
  1. User creates the new task branch from master and starts working in it.
  2. There is an active development on a master branch.
  3. User wants to get the latest updates from master to task branch.
  4. User performs regular rebase operation to move his commits on top of latest master commits.
  5. User is done with his task and performs the final rebase and merge to master.
  6. Git is able to apply fast-forward algorithm for the merge.

Git Branching - rebase

- **Interactive rebase**
  - Allows user to **alter individual commits** in the process of rebasing.
  - Support for powerful **history rewriting** features.
  - Useful for history cleanup: reword, edit, squash, fixup
  - Always amend commits that have **not been pushed** yet to avoid confusion.

Git Branching - conflicts

- Conflicts may occur during merge and rebase operations.
- Use the suitable **merge strategy** to avoid conflicts.
- When the conflict occurs:
  - **Abort** the merge with.
  - **Work through** the conflict and **continue**.
- **Visualize and resolve** conflict in merge tool.

Note:

- Rebase has an option to **skip/bypass** conflicting commit.

Git Branching - commands

git branch

- List all of the branches in your repository.

git branch <new-branch-name>

- Create a new branch called <new-branch-name>.

git branch -d <existing-branch-name>

- Delete the existing branch, safely. Use –D to force it.

git checkout <existing-branch-name>

- Navigate between the existing branches.

git checkout -b <new-branch-name> <remote>/<branch-name>

- Create and checkout a new local tracking branch.
- Or simply use the previous command if there is only one remote tracking branch called <branch-name>. This applies to Git 1.6.6+.

Git Branching - commands

git remote

- List the connections to remote repositories. Use -v to get URLs.

git remote add <remote-name> <url>

- Add a new connection to a remote repository.

git remote rm <remote-name>

- Remove the connection to a remote repository.

git fetch <remote-name>

- Fetch all of the branches from the remote repository.

git pull <remote-name>

- Fetch the remote's copy of the current branch and merge it into the local copy.

git push <remote-name> <branch-name>

- Push the specified branch to remote repository

Git Branching - commands

git merge <existing-branch-name>

- Merge the specified branch into current one and let Git to choose an algorithm.

git merge –no-ff <existing-branch-name>

- Merge the specified branch into current one and generate merge commit.

git merge –ff-only <existing-branch-name>

- Merge the specified branch into current one and refuse to merge when fast-forward is not possible.

git checkout task git rebase master

- Move the entire task branch to begin on the tip of the master branch, effectively incorporating all of the new commits in master.

Contents

Pull Requests - overview

- Mechanism for a developer to **notify coworkers.**
- **Not a Git feature**, functionality provided by e.g. Bitbucket or GitHub.
- Interface for **discussing proposed changes** before integrating them into the official project code base.
- **4 pieces** of information:
  - source repository
  - source branch
  - destination repository
  - destination branch

Pull Requests - workflow

1. **Developer creates task/feature branch** to deliver the code.
2. Once done the **developer pushes his dedicated local branch** to public repository.
3. **Pull requests is created** (it's good idea to rebase first).
4. Coworkers **review the code** and provide feedback.
5. Once approved the project maintainer **merges the changes.**