git sub – ähmm – dings

Peter Hrenka

Linux Tag Tübingen 2016

11. Juni 2016

Über mich

- Linux Anwender seit 1995
- Studium Informatik und Mathematik in Tübingen
- Sofwareentwickler `C++`, `python`, OpenGL
- regelmäßig auf OpenSource Konferenzen anzutreffen

- Tolles Tool, mächtig, weit verbreitet, aber ...

- Tolles Tool, mächtig, weit verbreitet, aber ...
- Projekte mit Abhängigkeiten!

- Tolles Tool, mächtig, weit verbreitet, aber ...
- Projekte mit Abhängigkeiten!
- Nicht-standard Abhängigkeiten!!

- Tolles Tool, mächtig, weit verbreitet, aber ...
- Projekte mit Abhängigkeiten!
- Nicht-standard Abhängigkeiten!!
- Was tun?

Erster Versuch: `git submodule`

Erster Versuch: `git submodule`

- schon bei `git-core` dabei

Erster Versuch: `git submodule`

- schon bei `git-core` dabei
- Konfigurationsdaten in “`.gitsubmodules`”

Erster Versuch: `git submodule`

- schon bei `git-core` dabei
- Konfigurationsdaten in "`.gitsubmodules`"
- Abhängige Repositories werden nicht automatisch ausgecheckt

Erster Versuch: `git submodule`

- schon bei `git-core` dabei
- Konfigurationsdaten in "`.gitsubmodules`"
- Abhängige Repositories werden nicht automatisch ausgecheckt
- Manuelles "`git submodule init`" notwendig

Erster Versuch: `git submodule`

- schon bei `git-core` dabei
- Konfigurationsdaten in "`.gitsubmodules`"
- Abhängige Repositories werden nicht automatisch ausgecheckt
- Manuelles "`git submodule init`" notwendig
- Komische Semantik der `submodules`: "Detached Head"

Erster Versuch: `git submodule`

- schon bei `git-core` dabei
- Konfigurationsdaten in "`.gitsubmodules`"
- Abhängige Repositories werden nicht automatisch ausgecheckt
- Manuelles "`git submodule init`" notwendig
- Komische Semantik der `submodules`: "Detached Head"
- Was passiert beim Merge von Branches mit submodules?

`git submodule` – Fazit

`git submodule` – Fazit

- Für einfache Abhängigkeiten

`git submodule` – Fazit

- Für einfache Abhängigkeiten
- Wenn man nicht am abhängigen Projekt mitentwickelt

`git submodule` – Fazit

- Für einfache Abhängigkeiten
- Wenn man nicht am abhängigen Projekt mitentwickelt
- Verwirrt Gelegenheitsbenutzer durch kompliziertes Handling

`git submodule` – Fazit

- Für einfache Abhängigkeiten
- Wenn man nicht am abhängigen Projekt mitentwickelt
- Verwirrt Gelegenheitsbenutzer durch kompliziertes Handling
- Konzept fragwürdig

Zweiter Versuch: `git subtree`

Zweiter Versuch: `git subtree`

- jetzt auch bei `git-core` dabei

Zweiter Versuch: `git subtree`

- jetzt auch bei `git-core` dabei
- keine Konfigurationsdatei – keine Metadaten

Zweiter Versuch: `git subtree`

- jetzt auch bei `git-core` dabei
- keine Konfigurationsdatei – keine Metadaten
- Subprojekte werden automatisch mit ausgecheckt

Zweiter Versuch: `git subtree`

- jetzt auch bei `git-core` dabei
- keine Konfigurationsdatei – keine Metadaten
- Subprojekte werden automatisch mit ausgecheckt
- Split- und Merge-Operationen möglich

Zweiter Versuch: `git subtree`

- jetzt auch bei `git-core` dabei
- keine Konfigurationsdatei – keine Metadaten
- Subprojekte werden automatisch mit ausgecheckt
- Split- und Merge-Operationen möglich
- “Schattenrepository” für Subprojekt im gleichen git-Repo

Zweiter Versuch: `git subtree`

- jetzt auch bei `git-core` dabei
- keine Konfigurationsdatei – keine Metadaten
- Subprojekte werden automatisch mit ausgecheckt
- Split- und Merge-Operationen möglich
- "Schattenrepository" für Subprojekt im gleichen git-Repo
- `squash` Option: Änderungen in Subprojekt in einem Commit zusammenfassen

Zweiter Versuch: `git subtree`

- jetzt auch bei `git-core` dabei
- keine Konfigurationsdatei – keine Metadaten
- Subprojekte werden automatisch mit ausgecheckt
- Split- und Merge-Operationen möglich
- "Schattenrepository" für Subprojekt im gleichen git-Repo
- `squash` Option: Änderungen in Subprojekt in einem Commit zusammenfassen
- Jedes Subrepository muss einzeln gemerged werden

Zweiter Versuch: `git subtree`

- jetzt auch bei `git-core` dabei
- keine Konfigurationsdatei – keine Metadaten
- Subprojekte werden automatisch mit ausgecheckt
- Split- und Merge-Operationen möglich
- "Schattenrepository" für Subprojekt im gleichen git-Repo
- `squash` Option: Änderungen in Subprojekt in einem Commit zusammenfassen
- Jedes Subrepository muss einzeln gemerged werden
- Merge/Split-Roundtrip nicht atomar

`git subtree` – Fazit

`git subtree` – Fazit

- Auch für komplexere Situationen geeignet

`git subtree` – Fazit

- Auch für komplexere Situationen geeignet
- Aktive, verzahnte Mitentwicklung am Subprojekt machbar

`git subtree` – Fazit

- Auch für komplexere Situationen geeignet
- Aktive, verzahnte Mitentwicklung am Subprojekt machbar
- Gelegenheitsnutzer merkt nichts davon

`git subtree` – Fazit

- Auch für komplexere Situationen geeignet
- Aktive, verzahnte Mitentwicklung am Subprojekt machbar
- Gelegenheitsnutzer merkt nichts davon
- Merge/Split-Workflow gewöhnungsbedürftig, wenig Beispiele

`git subtree` – Fazit

- Auch für komplexere Situationen geeignet
- Aktive, verzahnte Mitentwicklung am Subprojekt machbar
- Gelegenheitsnutzer merkt nichts davon
- Merge/Split-Workflow gewöhnungsbedürftig, wenig Beispiele
- Keine Metadaten: Konfigurationsdatei fehlt

Dritter Versuch: `git subrepo`

Dritter Versuch: `git subrepo`

- **nicht** bei `git-core` dabei

Dritter Versuch: `git subrepo`

- **nicht** bei `git-core` dabei
- Sondern hier: `https://github.com/ingydotnet/git-subrepo`

Dritter Versuch: `git subrepo`

- **nicht** bei `git-core` dabei
- Sondern hier: `https://github.com/ingydotnet/git-subrepo`
- Manuelle Installation notwendig

Dritter Versuch: `git subrepo`

- **nicht** bei `git-core` dabei
- Sondern hier: `https://github.com/ingydotnet/git-subrepo`
- Manuelle Installation notwendig
- Metadaten pro Unterverzeichnis in “`.gitrepo`”

Dritter Versuch: `git subrepo`

- **nicht** bei `git-core` dabei
- Sondern hier: `https://github.com/ingydotnet/git-subrepo`
- Manuelle Installation notwendig
- Metadaten pro Unterverzeichnis in "`.gitrepo`"
- Subprojekte werden automatisch mit ausgecheckt

Dritter Versuch: `git subrepo`

- **nicht** bei `git-core` dabei
- Sondern hier: `https://github.com/ingydotnet/git-subrepo`
- Manuelle Installation notwendig
- Metadaten pro Unterverzeichnis in "`.gitrepo`"
- Subprojekte werden automatisch mit ausgecheckt
- Klare Rollendefinition: `owners`, `users`, `colaborators`

Dritter Versuch: `git subrepo`

- **nicht** bei `git-core` dabei
- Sondern hier: `https://github.com/ingydotnet/git-subrepo`
- Manuelle Installation notwendig
- Metadaten pro Unterverzeichnis in "`.gitrepo`"
- Subprojekte werden automatisch mit ausgecheckt
- Klare Rollendefinition: `owners`, `users`, `colaborators`
- Push, pull, merge und sogar rebase unterstützt

Dritter Versuch: `git subrepo`

- **nicht** bei `git-core` dabei
- Sondern hier: `https://github.com/ingydotnet/git-subrepo`
- Manuelle Installation notwendig
- Metadaten pro Unterverzeichnis in "`.gitrepo`"
- Subprojekte werden automatisch mit ausgecheckt
- Klare Rollendefinition: `owners`, `users`, `colaborators`
- Push, pull, merge und sogar rebase unterstützt
- automatisches "squash" beim pull, Historie bei push bleibt erhalten

Dritter Versuch: `git subrepo`

- **nicht** bei `git-core` dabei
- Sondern hier: `https://github.com/ingydotnet/git-subrepo`
- Manuelle Installation notwendig
- Metadaten pro Unterverzeichnis in "`.gitrepo`"
- Subprojekte werden automatisch mit ausgecheckt
- Klare Rollendefinition: `owners`, `users`, `colaborators`
- Push, pull, merge und sogar rebase unterstützt
- automatisches "squash" beim pull, Historie bei push bleibt erhalten
- Kann vorhandenes Unterverzeichnis zu subrepo umwandeln (vgl. `subtree split`)

Dritter Versuch: `git subrepo`

- **nicht** bei `git-core` dabei
- Sondern hier: `https://github.com/ingydotnet/git-subrepo`
- Manuelle Installation notwendig
- Metadaten pro Unterverzeichnis in "`.gitrepo`"
- Subprojekte werden automatisch mit ausgecheckt
- Klare Rollendefinition: `owners`, `users`, `colaborators`
- Push, pull, merge und sogar rebase unterstützt
- automatisches "squash" beim pull, Historie bei push bleibt erhalten
- Kann vorhandenes Unterverzeichnis zu subrepo umwandeln (vgl. `subtree split`)
- "`--all`" Option, um alle Subrepos gleichzeitig zu behandeln

`git subrepo` – Fazit

`git subrepo` – Fazit

- Fortschrittlichstes “sub”-Tool

`git subrepo` – Fazit

- Fortschrittlichstes “sub”-Tool
- hat versucht aus Defiziten und Fehlern von `submodule` und `subtree` zu lernen

`git subrepo` – Fazit

- Fortschrittlichstes “sub”-Tool
- hat versucht aus Defiziten und Fehlern von `submodule` und `subtree` zu lernen
- nicht in `git-core` Upstream

Vielen Dank!

Fragen?