Software Engineering I (IB)

Blatt 0

Prof. Dr. Oliver Braun

Fakultät für Informatik und Mathematik
Hochschule München

Letzte Änderung: 09.10.2018 12:52

Die Abgaben in diesem Semester erfolgen über Git-Repositories die auf GitHub gehostet werden. Sie bekommen für jede Aufgabe ein eigenes Repository. Für die Abgabe ist jeweils eine Deadline vorgesehen, zu der Sie einen Pull-Reuest zur Abgabe erzeugt haben müssen.

Der GitHub-Account

Sie benötigen zur Verwendung der Git-Repos auf GitHub einen (kostenlosen) Account auf GitHub. Wenn Sie auf den weiter unten angegebenen GitHub Classroom Link klicken, müssen Sie sich einloggen oder einen neuen Account erstellen. Sie können für den Account einen beliebigen Username nutzen. Dieser muss nichts mit der Hochschule zu tun haben. Wenn Sie dann über den Link das erste Repository erzeugen, finden Sie eine Liste mit Namen. Dort müssen Sie Ihren GitHub-Account ihrem echten Namen zuordnen.

Datenschutz: Die Zuordnung zu Ihrem echten Namen sehen nur die Tutoren und ich. Die Repositories, die Sie in diesem Semester auf GitHub für die Abgaben nutzen, sind sog. private Repositories. Das heisst nur Sie selbst, die Tutoren und ich sehen die Repositories. Der Rest der Welt sieht nicht einmal, dass es diese Repositories gibt.

Tipp: Sie können, solange Sie studieren, am GitHub Education Programm teilnehmen und bekommen z.B. die Möglichkeit kostenlos private Repositories zu erzeugen und zu nutzen.

Gemeinsames Repository

Es gibt ein gemeinsames Repository für die Veranstaltung. Damit ich weiß wer von Ihnen am Praktikum teilnimmt um einen Schein zu bekommen, gibt es zwei Möglichkeiten

darauf zuzugreifen:

1. Wenn Sie **keinen** Schein benötigen, greifen Sie einfach direkt auf das Repository zu.
2. Wenn Sie einen Schein bekommen wollen, treten Sie über den GitHub-Classroom-Link `https://classroom.github.com/g/xzmfCBQ9` dem **einzigen** Team `ws18` bei. Erzeugen Sie **auf keinen Fall** ein anderes Team. Wählen Sie dabei Ihre E-Mail-Adresse aus, damit ich Ihnen Ihren GitHub-Account richtig zuordnen kann. Wenn Ihre E-Mail-Adresse nicht auftaucht, heisst das Sie sind im ZPA nicht für diese Veranstaltung eingetragen und können keine Leistungen ablegen.

Erste Schritte mit Git…

…finden Sie auf Blatt 0 der Veranstaltung Softwareentwicklung I (IB).

Aufgabe 1 — GitHub Flow und Pull-Request

In dieser Aufgabe sollen Sie lernen und ausprobieren, wie die Praktikumsaufgaben alleine zu bearbeiten und abzugeben sind. Lesen Sie sich dazu zunächst die Seite Bearbeitung und Abgabe der Praktikumsaufgaben, insbesondere im Bereich `Arbeiten, Abgabe und Feedback` den Teil über den `GitHub Flow`, durch.

Das Repository für diese Aufgabe bekommen Sie unter `https://classroom.github.com/a/dllpHsdK`. Dieses Repository ist leer. Erzeugen Sie über die Website eine Datei `README.md` (über den Link unter Quick Setup zum Beispiel) und schreiben Sie in die Datei Ihren Namen. Nachdem Sie die Datei committed haben, können Sie das Repository auf Ihren Rechner oder einen Laborrechner clonen.

Zunächst gehen Sie nach dem GitHub-Flow vor. Erzeugen Sie einen neuen Branch mit dem Namen `webdevinfo` (Web Development Information) und wechseln Sie in diesen:

```
git branch webdevinfo
git checkout webdevinfo
```

Ergänzen Sie Ihre Datei anschließend mit Infos über Ihre Kenntnisse in der Webentwicklung, so dass die Datei von der Struktur her so aussieht:

```
# Oliver Braun

## Kenntnisse in der Webentwicklung

Leider habe ich noch keine. Ich habe aber schon mit HTML und CSS gearbeitet.
```

Committen Sie die Änderung **mit einer vernünftigen Commitmessage**:

```
git add README.md
git commit -m "Infos über Webentwicklung eingefügt"
```

Pushen Sie anschließend auf GitHub. Ein `git push` alleine reicht dabei noch nicht, weil der Branch `webdevinfo` bisher nur bei Ihnen lokal vorliegt. Um Git zu sagen, dass der neue Branch gepusht werden soll, geben Sie ein:

```
git push origin webdevinfo
```

Damit wird der Branch `webdevinfo` auf den Server gepusht von dem Sie ursprünglich geclont haben (`origin`).

Bei einer zukünftigen Abgabe würden Sie jetzt weiter arbeiten und immer mal wieder committen und pushen. Dabei reicht dann ein `git push` zum Pushen aus, weil der `webdevinfo`-Branch ja jetzt auch auf GitHub liegt.

Nehmen wir an, Sie sind jetzt fertig und wollen abgeben. Dazu gehen Sie auf die Webpage Ihres GitHub-Repositorys. Wenn Sie alles richtig gemacht haben, zeigt Ihnen GitHub eh schon ein gelbes Feld mit dem Sie einen Pull-Request erzeugen können. Also tun Sie dies in dem Sie auf `compare & pull request` klicken.

So ein Pull-Request sieht ähnlich aus wie ein Issue. Er stellt quasi die Anfrage (Request) die Änderungen eines Branches in einen anderen zu bringen. Die beiden Branches, die in dem Fall schon richtig ausgewählt sein sollten, sind als `base`-Branch **in** den gepullt wird der `master`-Branch und als `compare`-Branch **von** dem gepullt wird der `webdevinfo`-Branch.

Achten Sie darauf, dass der Titel des Pull-Request sinnvoll ist. Ändern Sie in in diesem Fall einfach in `Abgabe`. Wenn Sie mir eine zusätzliche Nachricht zukommen lassen wollen, schreiben Sie einfach einem Text dazu in das Feld `Write`. Nutzen Sie dazu Markdown!

Wenn Sie zufrieden sind, klicken Sie auf `Create Pull Request`. Auf der dann sichtbaren Seite tragen Sie mich (`obcode`) **und** Ihren Tutor als `Reviewer` ein. Wenn Sie diesen Schritt nicht machen, bekomme ich nichts von Ihrem Pull-Request mit und das heisst, Sie haben **nicht** abgegeben. Bei den zukünftigen Aufgaben führt das zum Nichtbestehen des Scheines.

Ihr Tutor und ich werde mir dann Ihre Abgabe ansehen. Wir können dann eine von drei Aktionen ausführen:

- *Comment*: einfach nur kommentieren
- *Approve*: dann haben Sie die Abgabe bestanden
- *Request changes*: dann müssen Sie noch etwas ändern vor Sie bestanden haben

Wenn Sie Änderungen vornehmen sollen, können Sie einfach in dem Branch weiter arbeiten. Sobald Sie neue Commits pushen, werden diese automatisch dem Pull-Request zugeordnet. Sobald Sie alles erledigt haben, tragen Sie uns erneut als `Reviewer` ein. Damit werden wir wieder per E-Mail benachrichtigt, denn ansonsten bekommen wir das nicht mit. Und dann beginnt das Ganze von vorne…

Wenn Sie gleich oder nach einer Änderung ein `Approve` bekommen haben, können Sie den Pull-Request mergen. Dazu klicken Sie einfach auf den `Merge Pull Request`-Button und dann auf `Merge`. Den `webdevinfo`-Branch können Sie dann auch wieder löschen. Die gesamte Info ist jetzt im `master`-Branch.

Wenn Sie den Pull-Request nicht selbst mergen, sehe ich am Ende des Semesters nicht, dass Sie diese Abgabe bestanden haben. Daher unbedingt noch mergen!

Aufgabe 2 — Git Flow und Pull-Request

In dieser Aufgabe sollen Sie lernen und ausprobieren, wie die Praktikumsaufgaben gemeinsam zu bearbeiten und abzugeben sind. Lesen Sie sich dazu die Seite Bearbeitung und Abgabe der Praktikumsaufgaben, insbesondere im Bereich `Arbeiten, Abgabe und Feedback` den Teil über den `Git Flow`, durch. Um zunächst das Arbeiten mit dem Git Flow zu lernen, bearbeiten Sie diese Aufgabe dennoch jeder alleine.

Das Repository für diese Aufgabe bekommen Sie unter `https://classroom.github.com/a/Lhvrxstc`. Dieses Repository ist leer. Erzeugen Sie über die Website eine Datei `README.md` (über den Link unter Quick Setup zum Beispiel) und schreiben Sie in die Datei Ihren Namen. Nachdem Sie die Datei committed haben, können Sie das Repository auf Ihren Rechner oder einen Laborrechner clonen.

Viele Git-GUIs bringen bereits eine Git Flow-Unterstützung mit, z.B. SourceTree welches in den Laboren installiert sein sollte. Auf dem Aufgabenblatt erläutere ich das analoge Vorgehen auf der Kommandozeile, nachdem Sie es installiert haben.

Zuerst müssen Sie Ihr Repository für Git Flow initialisieren:

```
git flow init
```

wobei Sie die angegebenen Defaults übernehmen können.

Pushen Sie den `develop`-Branch auf GitHub:

```
git push origin develop
```

Um nun an einem Feature zu arbeiten, geben Sie ein

```
git flow feature start expectations
```

wobei `expectations` der Name des Branches sein soll (Erwartungen).

Veröffentlichen Sie Ihren Branch mit Git Flow auf GitHub:

```
git flow feature publish expectations
```

Editieren Sie nun die `README.md` und schreiben Sie, in Markdown formatiert, Ihre Erwartungen an diese Lehrveranstaltung hinein. Committen Sie und pushen Sie. Jetzt könnten Sie einfach weiter arbeiten, committen und pushen.

Je nach späterer Aufgabenstellung können Sie Ihr Feature selbst in den `develop`-Branch mergen oder müssen ein Review von einem Gruppenmitglied anfordern. Im Fall, dass Sie ein Review anfordern müssen, erstellen Sie einen entsprechenden Pull-Request auf GitHub. Den **genehmigten** Pull Request mergen Sie dann am Besten lokal mit Git Flow. Genauso gehen Sie vor, wenn Sie das Feature selbst mergen:

```
git flow feature finish expectations
```

Wenn es einen Pull-Request gibt, wird dieser automatisch geschlossen. So wie mit dem Expectations-Feature gehen Sie jetzt mit jedem neuen Feature vor.

Um dann (gemeinsam) abzugeben, erzeugen Sie ein Release mit Git Flow **und** einen Pull-Request. Mergen Sie den Pull-Request dann aber nicht auf der GitHub-Seite, sobald die Abgabe okay ist, sondern mit Git Flow.

Erster Schritt ist das Erzeugen eines Release-Branches. Der Name des Release-Branches kann bei einer einfachen Abgabe auch einfach `abgabe` sein. Üblicherweise gibt man aber einem *echten* Release eine Nummer. Wir nutzen hier jetzt einfach `abgabe`:

```
git flow release start abgabe
```

Damit befinden wir uns jetzt in einem Branch `abgabe`. Dort könnten jetzt noch Release-spezifische Commits hinzugefügt werden, wie z.B. das Hochsetzen der Versionsnummer in irgendwelchen Dateien.

Um dann auch den Pull-Request zur Abgabe zu erzeugen, müssen Sie den Release-Branch auf GitHub veröffentlichen:

```
git flow release publish abgabe
```

Dann können Sie auf GitHub einen neuen Pull-Request öffnen und zwar dieses mal vom Release-Branch direkt in den `master`-Branch. Schreiben Sie wieder etwas Sinnvolles in die Überschrift, eventuell noch Zusatzinfo in das Feld darunter und tragen Sie Ihren Tutor und mich als Reviewer ein.

Wenn der Pull-Request akzeptiert ist, mergen Sie **auf keinen Fall** direkt auf der GitHub-Seite. Sie haben sonst zu vieles noch nachzuarbeiten. Nutzen Sie lokal

```
git flow release finish abgabe
```

Geben Sie eine sinnvolle Message für den Merge und auch für den Tag an. Mit dem Tag taggen Sie den Commit als Release-Commit. Damit kann auch später wieder genau zu dieser Version zurück gesprungen werden.

Der Release ist jetzt im `master`- und im `develop`-Branch. Sie befinden sich wieder im `develop`-Branch. Jetzt müssen Sie nur noch alle beide Branches auf GitHub pushen und vor allem auch den Tag:

```
git push --all
git push --tags
```

Wenn Sie das Repository jetzt auf GitHub ansehen, stellen Sie fest, dass der Pull-Request automatisch geschlossen wurde und es gibt jetzt auch ein Release.