Commits zusammenfassen (Squash) in PhpStorm □□

Du hast also fleißig gearbeitet und dabei mehrere kleine Commits erstellt – vielleicht mit Nachrichten wie *„WIP*", *„Fix typo*", *„Noch ein Fix*" oder *„Jetzt aber wirklich fertig*". Bevor du diese „Commits des Schreckens" auf GitHub pushst, möchtest du sie zu einem einzigen, sauberen Commit zusammenfassen. Das ist nicht nur guter Stil, sondern macht die Historie für dich und dein Team deutlich lesbarer. Lass mich dich Schritt für Schritt durch den gesamten Prozess führen.

Warum überhaupt Commits squashen?

Bevor wir in die Praxis einsteigen, kurz zum **Warum**: Eine saubere Commit-Historie ist wie ein gut geschriebenes Tagebuch deines Projekts. Wenn du später (oder ein Kollege) nachvollziehen möchtest, welche Änderung wann und warum gemacht wurde, helfen aussagekräftige Commits enorm. Niemand möchte sich durch zehn *„WIP*"-Commits wühlen, um zu verstehen, was eigentlich passiert ist.

Vorteile des Squashens:

- **Lesbarkeit:** Ein Commit pro logischer Änderung statt vieler Mikro-Commits
- **Einfacheres Debugging:** Mit `git bisect` findest du Bugs schneller, wenn jeder Commit einen sinnvollen Zustand repräsentiert
- **Saubere Pull Requests:** Reviewer können die Änderungen besser nachvollziehen
- **Professioneller Eindruck:** Zeigt, dass du dir Gedanken über deine Arbeit machst

---

Die Ausgangssituation

Nehmen wir an, du hast in den letzten Stunden an einem Login-Formular gearbeitet und dabei folgende Commits erstellt (vom ältesten zum neuesten):

```
a1b2c3d  WIP: Login-Formular angefangen
d4e5f6g  Passwort-Feld hinzugefügt
h7i8j9k  Typo im Label gefixt
l0m1n2o  CSS angepasst
p3q4r5s  Validation hinzugefügt
```

Diese fünf Commits sollen zu einem einzigen werden:

```
x9y8z7w  feat: Login-Formular mit Validierung implementiert
```

Methode 1: Der komfortable Weg über das Git-Log in PhpStorm

Dies ist die **empfohlene Methode** für die meisten Situationen, da sie visuell und intuitiv ist.

1. **Öffne das Git-Tool-Window**
   Navigiere zu **View → Tool Windows → Git** oder nutze die Tastenkombination `Alt + 9` (Windows/Linux) bzw. `⌘ + 9` (macOS). Du siehst jetzt das Git-Log mit allen Commits deines Repositories.
2. **Filtere auf deinen aktuellen Branch**
   In der Toolbar des Git-Fensters siehst du ein Dropdown-Menü für Branches. Stelle sicher, dass du nur die Commits deines Feature-Branches siehst. Du kannst auch im Suchfeld nach deinem Branch-Namen filtern.
3. **Identifiziere den Commit-Bereich**
   Scrolle durch die Liste und identifiziere die Commits, die du zusammenfassen möchtest. In unserem Beispiel sind das die fünf Commits von *„WIP: Login-Formular angefangen*" bis *„Validation hinzugefügt*".
4. **Wähle den Basis-Commit aus**
   Hier wird es wichtig: Du musst den Commit **vor** deinem ersten zu squashenden Commit finden. Das ist der Commit, auf dem deine Arbeit aufbaut – typischerweise der letzte Commit, bevor du mit dem Feature angefangen hast. Dieser Commit selbst wird **nicht** verändert, er dient nur als Referenzpunkt.
5. **Starte den Interactive Rebase**
   Klicke mit der **rechten Maustaste** auf diesen Basis-Commit und wähle im Kontextmenü: **Interactively Rebase from Here...** (auf Deutsch: *„Interaktiv Rebase von hier...*").
6. **Das Interactive Rebase-Fenster**
   PhpStorm öffnet jetzt ein Fenster mit dem Titel *„Rebasing Commits*" oder *„Git Interactive Rebase*". Du siehst eine Liste aller Commits, die nach deinem gewählten Basis-Commit kommen – also genau die Commits, die du bearbeiten möchtest. Die Liste zeigt:

- Die **Aktion** (standardmäßig „pick" für jeden Commit)
- Den **Commit-Hash** (verkürzt)
- Die **Commit-Nachricht**

7. **Commits zum Squashen markieren**
   Jetzt kommt der entscheidende Schritt. Du hast mehrere Möglichkeiten, die Aktion für jeden Commit zu ändern:
   - **Doppelklick** auf die Aktion öffnet ein Dropdown-Menü
   - **Rechtsklick** auf einen Commit zeigt alle verfügbaren Aktionen
   - Wähle mehrere Commits mit `Strg + Klick` (Windows/Linux) oder `⌘ + Klick` (macOS) und ändere die Aktion für alle gleichzeitig

   Für das Squashen gehst du so vor:
   - Der **erste Commit** (der älteste, auf den die anderen aufbauen) behält die Aktion **„pick"**
   - Alle **folgenden Commits**, die du zusammenfassen möchtest, änderst du auf **„squash"** oder **„fixup"**
8. **Der Unterschied zwischen „squash" und „fixup"**

Aktion	Beschreibung
squash	Kombiniert den Commit mit dem vorherigen. Die Commit-Nachricht wird beibehalten, und du kannst sie im nächsten Schritt bearbeiten.
fixup	Kombiniert den Commit mit dem vorherigen, aber **verwirft** die Commit-Nachricht. Nutze das für Commits wie *„Typo gefixt*", deren Nachricht nicht erhalten bleiben muss.

   Für unser Beispiel könntest du es so konfigurieren:

```
pick    a1b2c3d  WIP: Login-Formular angefangen
squash  d4e5f6g  Passwort-Feld hinzugefügt
fixup   h7i8j9k  Typo im Label gefixt
squash  l0m1n2o  CSS angepasst
squash  p3q4r5s  Validation hinzugefügt
```

9. **Starte den Rebase**
   Klicke auf **„Start Rebasing"** (oder den entsprechenden Button in deiner PhpStorm-Version). PhpStorm beginnt jetzt, die Commits neu zu schreiben.
10. **Bearbeite die kombinierte Commit-Nachricht**
    Nach dem Rebase öffnet PhpStorm ein Fenster, in dem du die **neue, kombinierte Commit-Nachricht** eingeben kannst. Du siehst dort alle Nachrichten der Commits, die mit „squash" markiert waren (nicht die mit „fixup" – deren Nachrichten wurden verworfen). Das sieht ungefähr so aus:

```
# This is a combination of 5 commits.
# The first commit's message is:
```

```
WIP: Login-Formular angefangen

# This is the 2nd commit message:
Passwort-Feld hinzugefügt

# This is the 4th commit message:
CSS angepasst

# This is the 5th commit message:
Validation hinzugefügt
```

Lösche diesen ganzen Text und schreibe stattdessen eine saubere, aussagekräftige Commit-Nachricht:

```
feat: Login-Formular mit Validierung implementiert


- Login-Formular mit Benutzername- und Passwort-Feld erstellt
- Client-seitige Validierung für beide Felder hinzugefügt
- CSS-Styling für konsistentes Design angepasst
```

11. **Bestätige die Nachricht**
    Klicke auf **„OK"** oder **„Commit"**, um den Prozess abzuschließen.

Methode 2: Der schnelle Weg mit „Squash Commits" direkt in PhpStorm

PhpStorm bietet auch eine **direkte Squash-Funktion**, die noch schneller ist, wenn du weißt, welche Commits du zusammenfassen möchtest.

1. **Öffne das Git-Log** wie in Methode 1 beschrieben (`Alt + 9` oder `⌘ + 9`).
2. **Wähle die Commits aus**, die du zusammenfassen möchtest. Halte `Strg` (Windows/Linux) oder `⌘` (macOS) gedrückt und klicke auf jeden Commit, den du squashen möchtest. Die Commits müssen **aufeinanderfolgend** sein.
3. **Rechtsklick auf die Auswahl** und wähle im Kontextmenü: **Squash Commits...** (auf Deutsch: *„Commits zusammenfassen..."*).
4. **Bearbeite die Commit-Nachricht** im erscheinenden Dialog und bestätige mit **„OK"**.

Diese Methode ist schneller, bietet aber weniger Kontrolle als der Interactive Rebase – du kannst zum Beispiel nicht einzelne Commits mit „fixup" statt „squash" markieren.

Methode 3: Über das Terminal in PhpStorm

Falls du lieber mit dem Terminal arbeitest oder die grafischen Methoden aus irgendeinem Grund nicht funktionieren, kannst du den Interactive Rebase auch direkt über die Kommandozeile durchführen.

1. **Öffne das Terminal in PhpStorm**
   Navigiere zu **View → Tool Windows → Terminal** oder nutze `Alt + F12` (Windows/Linux) bzw. `⌥ + F12` (macOS).
2. **Finde heraus, wie viele Commits du zusammenfassen möchtest**
   Mit `git log --oneline` siehst du eine kompakte Liste deiner letzten Commits:

   ```
   git log --oneline -10
   ```

   Das zeigt dir die letzten 10 Commits. Zähle, wie viele du squashen möchtest – in unserem Beispiel sind es 5.
3. **Starte den Interactive Rebase**

   ```
   git rebase -i HEAD~5
   ```

   Damit sagst du Git: „Ich möchte die letzten 5 Commits interaktiv bearbeiten."
4. **Bearbeite die Commit-Liste im Editor**
   Git öffnet einen Text-Editor (in PhpStorm normalerweise der integrierte Editor oder der von dir konfigurierte). Du siehst:

   ```
   pick a1b2c3d WIP: Login-Formular angefangen
   pick d4e5f6g Passwort-Feld hinzugefügt
   pick h7i8j9k Typo im Label gefixt
   pick l0m1n2o CSS angepasst
   pick p3q4r5s Validation hinzugefügt
   ```

   Ändere es zu:

   ```
   pick a1b2c3d WIP: Login-Formular angefangen
   squash d4e5f6g Passwort-Feld hinzugefügt
   fixup h7i8j9k Typo im Label gefixt
   squash l0m1n2o CSS angepasst
   ```

```
squash p3q4r5s Validation hinzugefügt
```

Du kannst auch die Kurzformen verwenden: `p` für pick, `s` für squash, `f` für fixup.

5. **Speichere und schließe den Editor**
   In PhpStorms integriertem Editor klickst du einfach auf den Bestätigungs-Button. Bei Vim-Editoren: `:wq` und Enter.
6. **Bearbeite die kombinierte Commit-Nachricht** wie bei Methode 1 beschrieben.

Was passiert, wenn etwas schief geht? □□

Keine Panik! Ein Rebase kann abgebrochen werden, solange er noch läuft:

Im Terminal:

```
git rebase --abort
```

In PhpStorm: Wenn der Rebase fehlschlägt (z.B. wegen Konflikten), zeigt PhpStorm eine Benachrichtigung an. Du kannst dort **„Abort Rebase"** wählen, um zum Zustand vor dem Rebase zurückzukehren.

Wenn du den Rebase schon abgeschlossen hast und merkst, dass etwas nicht stimmt, hilft dir das **Reflog**:

```
git reflog
```

Dort siehst du alle Zustände deines Repositories, auch die vor dem Rebase. Mit `git reset --hard HEAD@{n}` (wobei `n` die Nummer aus dem Reflog ist) kannst du zu einem früheren Zustand zurückkehren.

Wichtige Hinweise und Warnungen ⚠

1. **Nur unpushte Commits squashen!**

Die goldene Regel des Rebasings: **Schreibe niemals die Historie um, die bereits gepusht wurde** (es sei denn, du bist dir absolut sicher, was du tust, und hast das mit deinem Team abgesprochen). Wenn du bereits gepushte Commits squashst, müsstest du mit `git push --force` pushen, was die Historie für alle anderen Entwickler kaputt macht.

2. **Force-Push nur bei Feature-Branches**
   Wenn du an einem Feature-Branch arbeitest, den nur du nutzt, und du bereits gepusht hast, kannst du nach dem Squash einen Force-Push machen:

   ```
   git push --force-with-lease origin feature/login-formular
   ```

   `--force-with-lease` ist sicherer als `--force`, weil es prüft, ob jemand anders in der Zwischenzeit gepusht hat.
3. **Mache vorher ein Backup (optional)**
   Wenn du unsicher bist, erstelle vorher einen temporären Branch als Backup:

   ```
   git branch backup-vor-squash
   ```

   Falls etwas schief geht, kannst du immer zu diesem Branch zurückkehren.

Ein Beispiel-Workflow zusammengefasst

```
flowchart TD
    A["Du hast 5 WIP-Commits\nim Feature-Branch"] --> B["Oeffne Git-Log\nAlt+9 oder Cmd+9"]
    B --> C["Finde den Commit VOR\ndeiner Arbeit"]
    C --> D["Rechtsklick:\nInteractively Rebase from Here"]
    D --> E["Markiere Commits\nmit squash oder fixup"]
    E --> F["Klicke: Start Rebasing"]
    F --> G["Schreibe eine saubere\nkombinierte Commit-Nachricht"]
    G --> H["Bestaetigen"]
    H --> I["Fertig: 1 sauberer Commit\nstatt 5 WIP-Commits"]

    style A fill:#ffcccc,color:#000000
    style I fill:#ccffcc,color:#000000
```

Zusammenfassung

Schritt	Aktion
1	Git-Log öffnen (`Alt + 9` / `⌘ + 9`)
2	Basis-Commit identifizieren (der Commit **vor** deiner Arbeit)
3	Rechtsklick → *„Interactively Rebase from Here..."*
4	Commits mit „squash" oder „fixup" markieren
5	*„Start Rebasing"* klicken
6	Neue, saubere Commit-Nachricht schreiben
7	Bestätigen – fertig!

Mit etwas Übung wird das Squashen zur Routine und deine Commit-Historie wird es dir danken. Dein zukünftiges Ich (und deine Kollegen) werden dich dafür lieben, wenn sie durch eine saubere, nachvollziehbare Historie scrollen können, statt sich durch zwanzig *„WIP"*- und *„Fix"*-Commits zu kämpfen!

Revision #1
Created 2026-06-29 15:50:58 UTC by art10m
Updated 2026-06-29 15:51:23 UTC by art10m