Git- und Log-Werkzeugfenster erkunden

Das Git-Werkzeugfenster als Kommandozentrale

Während das Commit-Werkzeugfenster den *Weg in* die Historie beschreibt, ist das **Git-Werkzeugfenster** dein Blick *auf* die Historie. Es fasst alles zusammen, was mit dem aktuellen Zustand deines Repositorys zu tun hat: Branches, Commits, lokale Änderungen und die tatsächlich ausgeführten Git-Befehle. Du öffnest es über `Alt` + `9` oder über *View → Tool Windows → Git*.

Das Fenster ist in mehrere Reiter unterteilt, die je nach PhpStorm-Version leicht unterschiedlich benannt sind, inhaltlich aber stets dieselben drei Aufgaben abbilden:

Reiter	Aufgabe
Log	Commit-Graph, Branches, Tags, Details zu einzelnen Commits
Local Changes	Aktueller Stand von Arbeitsverzeichnis und Index (siehe Commit-Werkzeugfenster)
Console	Protokoll aller intern ausgeführten Git-Kommandozeilenbefehle

Der Log-Tab im Detail

Der **Log-Tab** ist das Herzstück des Fensters. Er visualisiert die Commit-Historie als Graph – vergleichbar mit `git log --graph --oneline --all`, nur interaktiv und deutlich übersichtlicher.

Der Graphbereich zeigt links die Commit-Linien mit farblich getrennten Branches. Jeder Knotenpunkt ist ein Commit; Verzweigungen und Zusammenführungen sind sofort erkennbar. Rechts daneben stehen Commit-Nachricht, Autor und Zeitpunkt.

Das Detailpanel öffnet sich, sobald du einen Commit anklickst. Es zeigt:

- die vollständige Commit-Nachricht,
- Autor, Committer und Zeitstempel,
- die betroffenen Dateien inklusive Diff-Vorschau per Doppelklick.

Die Branch-Spalte listet, welche lokalen und entfernten Branches auf den jeweiligen Commit zeigen – praktisch, um schnell zu erkennen, ob eine Änderung bereits gepusht wurde.

Filtern und Suchen

Über der Graphansicht befindet sich eine Filterleiste, mit der du die angezeigte Historie gezielt einschränkst:

- **Branch-Filter**: nur bestimmte Branches oder *alle* anzeigen.
- **Benutzer-Filter**: Commits eines bestimmten Autors isolieren.
- **Datumsfilter**: Zeiträume eingrenzen.
- **Textsuche**: Commit-Nachrichten oder – über die Option „Search in commit contents“ – auch geänderte Dateiinhalte durchsuchen.

Diese Filter lassen sich kombinieren, was besonders bei größeren Projekten mit vielen parallelen Branches wertvoll ist, um sich schnell einen Überblick über den Fortschritt eines Features zu verschaffen.

Kontextaktionen direkt im Graph

Ein Rechtsklick auf einen Commit öffnet ein Kontextmenü mit den wichtigsten Operationen, die du sonst über die Kommandozeile ausführen würdest:

- `Checkout Revision` – entspricht `git checkout <commit>`,
- `New Branch from Selected Commit`,
- `Cherry-Pick`,
- `Revert Commit`,
- `Reset Current Branch to Here`.

So lassen sich viele fortgeschrittene Git-Operationen ausführen, ohne den Befehl im Detail zu kennen – wichtig ist aber, dass du in späteren Kapiteln verstehst, *was* im Hintergrund tatsächlich passiert, um Fehlbedienungen zu vermeiden.

Der Console-Tab: Transparenz statt Blackbox

Ein zentraler Vorteil von PhpStorm gegenüber rein grafischen Git-Clients ist der **Console-Tab**. Hier protokolliert die IDE jeden intern ausgeführten Git-Befehl im Klartext – inklusive Parametern und Rückgabewerten.

Dieser Reiter ist didaktisch besonders wertvoll: Klickst du beispielsweise in der Oberfläche auf *Pull*, kannst du in der Console exakt nachlesen, dass PhpStorm etwa

```
git fetch origin
git merge origin/main
```

ausgeführt hat. So verknüpfst du grafische Aktionen direkt mit dem zugrunde liegenden Kommandozeilenwissen, das dir in den folgenden Kapiteln immer wieder begegnet.

Zusammenspiel der Reiter

```
flowchart LR
    A["Log-Tab: Commit-Historie und Branches"] --> D["Git-Werkzeugfenster"]
    B["Local Changes: Arbeitsverzeichnis und Index"] --> D
    C["Console: Ausgeführte Git-Befehle"] --> D
    D --> E["Vollständiger Überblick über Repository-Zustand"]
```

Praktische Hinweise für den Alltag

- **Doppelklick auf einen Commit** öffnet direkt die Diff-Ansicht der geänderten Dateien – der schnellste Weg, eine fremde Änderung zu verstehen.
- **„Show Diff Preview on Double Click“** in den Einstellungen beschleunigt wiederkehrende Prüfungen.
- Nutze den Log-Tab regelmäßig *vor* riskanten Operationen wie Rebase oder Reset, um dir den aktuellen Branch-Zustand bewusst zu machen.
- Die **Console** solltest du dir angewöhnen zu beobachten, gerade in der Lernphase – sie ist dein Fenster zur eigentlichen Git-Mechanik hinter der Oberfläche.

Mit einem sicheren Umgang mit Log- und Console-Tab hast du die Grundlage geschaffen, um im nächsten Schritt Änderungen und Diffs gezielt zu untersuchen – ein Werkzeug, das dich durch den gesamten weiteren Kurs begleiten wird.

Revision #1
Created 2026-07-26 17:07:24 UTC by art10m
Updated 2026-07-26 17:07:45 UTC by art10m