Commit-Werkzeugfenster konfigurieren

Das *Commit*-Werkzeugfenster ist die zentrale Anlaufstelle für alle Änderungen, die du in ein Repository einpflegen möchtest. Es zeigt dir auf einen Blick, welche Dateien bearbeitet, neu erstellt oder gelöscht wurden, und erlaubt dir, diese gezielt zu gruppieren, zu prüfen und mit einer aussagekräftigen Nachricht zu versehen, bevor sie als Commit in die Historie wandern.

Du öffnest es über:

```
View → Tool Windows → Commit
```

oder mit dem Tastaturkürzel `Alt + 0`. Standardmäßig erscheint es links am Bildschirmrand, kann aber wie jedes andere Werkzeugfenster angedockt, verschoben oder als eigenständiges Fenster „abgerissen“ werden.

Modaler und nicht-modaler Commit

PhpStorm bietet zwei grundlegend unterschiedliche Arbeitsweisen für den Commit-Vorgang, die sich in ihrem Verhalten deutlich unterscheiden:

Modus	Verhalten	Empfehlung
Nicht-modal *(Standard seit neueren Versionen)*	Der Commit-Bereich ist dauerhaft im Werkzeugfenster sichtbar; du kannst währenddessen weiterarbeiten, Dateien wechseln und andere Fenster nutzen.	Für den Alltag empfehlenswert, da flüssiger und weniger blockierend.
Modal *(klassischer Commit-Dialog)*	Ein separates Dialogfenster öffnet sich, blockiert andere Aktionen und muss explizit bestätigt oder abgebrochen werden.	Sinnvoll, wenn du einen klaren, unterbrechungsfreien Prüfungsschritt bevorzugst.

Du steuerst dieses Verhalten unter:

```
Settings → Version Control → Commit
```

über die Option **„Use non-modal commit interface“**. Deaktivierst du sie, kehrst du zum klassischen, modalen Dialog zurück – eine Einstellung, die besonders Umsteiger von älteren PhpStorm-Versionen oder anderen IDEs oft bevorzugen.

Wichtige Grundeinstellungen

Unter `Settings → Version Control → Commit` findest du die zentralen Schalter, mit denen sich der Commit-Vorgang an deinen Arbeitsstil anpassen lässt:

- **„Show unversioned files“** – zeigt nicht verfolgte Dateien direkt in der Commit-Liste an, statt sie zu verstecken.
- **„Before Commit“-Checks** – eine Reihe automatischer Prüfungen, die *vor* jedem Commit ausgeführt werden können (siehe nächster Abschnitt).
- **„Clear initial commit message“** – legt fest, ob das Nachrichtenfeld bei jedem neuen Commit leer beginnt oder die letzte Eingabe behält.
- **„Move Focus to the Commit Message Field“** – springt der Cursor nach dem Öffnen automatisch in das Nachrichtenfeld, was den Arbeitsfluss beschleunigt.

Diese Optionen wirken unscheinbar, summieren sich aber über Hunderte von Commits zu einem spürbaren Unterschied in der täglichen Effizienz.

Automatische Prüfungen vor dem Commit

Eine der nützlichsten Funktionen des Commit-Werkzeugfensters sind die **„Before Commit“-Prüfungen**. Sie laufen automatisch ab, sobald du auf *Commit* klickst, und verhindern, dass unsaubere oder fehlerhafte Änderungen versehentlich in die Historie gelangen.

Zu den wichtigsten gehören:

- **Reformat code** – wendet den konfigurierten Code-Style automatisch auf die geänderten Dateien an.
- **Optimize imports** – entfernt ungenutzte und sortiert vorhandene Imports, besonders relevant in PHP-Klassen mit vielen `use`-Anweisungen.
- **Rearrange code** – ordnet Klassenmitglieder gemäß den definierten Regeln neu an.

- **Perform code analysis** – führt eine Inspektion durch und warnt vor Fehlern, bevor sie festgeschrieben werden.
- **Check TODO** – weist auf offene `TODO`-Kommentare in den betroffenen Dateien hin.

> *Praxistipp:* Aktiviere zu Beginn nicht alle Prüfungen gleichzeitig. Beginne mit „Reformat code“ und „Optimize imports“, um ein Gefühl für die automatischen Eingriffe zu entwickeln, bevor du strengere Analysen hinzufügst.

Jede Prüfung lässt sich einzeln ein- oder ausschalten und kann so konfiguriert werden, dass sie bei Verstößen den Commit blockiert oder lediglich eine Warnung anzeigt.

Commit-Nachrichten strukturieren

Direkt im Werkzeugfenster befindet sich das Nachrichtenfeld für die Commit-Beschreibung. PhpStorm unterstützt dich dabei auf mehreren Ebenen:

- **Vorlagen (Commit Message Templates):** Unter `Settings → Version Control → Commit Message` kannst du eine feste Vorlage hinterlegen, etwa im Sinne der *Conventional Commits*:

```
<type>(<scope>): <subject>


<body>
```

- **Zeilenlängen-Hinweis:** Eine vertikale Linie im Eingabefeld markiert die empfohlene maximale Zeilenlänge der ersten Zeile (üblicherweise 72 Zeichen), was der gängigen Git-Konvention entspricht.
- **Verlauf früherer Nachrichten:** Über das Uhrsymbol im Nachrichtenfeld rufst du zuvor verwendete Commit-Nachrichten ab – praktisch bei wiederkehrenden Formulierungen innerhalb eines Features.

Changelists gezielt nutzen

Das Commit-Werkzeugfenster gruppiert Änderungen standardmäßig in der *Default*-Changelist. Du kannst jedoch beliebig viele weitere Changelists anlegen, um thematisch getrennte Änderungen parallel vorzubereiten, ohne sie sofort zu committen.

- Über das Kontextmenü einer Datei wählst du **„Move to Another Changelist“**, um sie einer neuen Gruppe zuzuweisen.
- Jede Changelist besitzt einen eigenen Namen und optional einen Kommentar, was besonders hilfreich ist, wenn du an mehreren logisch getrennten Themen gleichzeitig arbeitest.
- Beim Commit wählst du gezielt aus, welche Changelist eingecheckt werden soll – die übrigen bleiben unberührt im Arbeitsverzeichnis erhalten.

Diese Funktion ersetzt in vielen Alltagssituationen das manuelle Stashen und erlaubt eine deutlich übersichtlichere Arbeitsweise, wenn mehrere kleine Aufgaben nebeneinander existieren.

Diff-Vorschau direkt im Fenster

Ein oft übersehenes, aber äußerst hilfreiches Detail ist die integrierte Diff-Vorschau. Aktivierst du unter `Settings → Version Control → Commit` die Option **„Show diff preview on double click“**, öffnest du mit einem Doppelklick auf eine Datei sofort die Änderungsansicht, ohne das Commit-Fenster verlassen zu müssen.

Alternativ lässt sich die Vorschau dauerhaft als eingebettetes Panel neben der Dateiliste anzeigen – so behältst du beim Formulieren der Commit-Nachricht stets den inhaltlichen Kontext im Blick, was die Qualität deiner Beschreibungen merklich verbessert.

Zusammenfassung

Das Commit-Werkzeugfenster ist weit mehr als ein einfacher „Speichern“-Knopf. Durch die Kombination aus **automatischen Prüfungen**, **strukturierten Nachrichtenvorlagen**, **flexiblen Changelists** und **integrierter Diff-Ansicht** wird es zu einem Werkzeug, das Disziplin und Geschwindigkeit gleichzeitig fördert. Wer diese Einstellungen bewusst an den eigenen Arbeitsstil anpasst, reduziert Flüchtigkeitsfehler erheblich und legt damit den Grundstein für eine saubere, nachvollziehbare Projekthistorie – ein Thema, das im weiteren Verlauf des Kurses noch vertieft wird.

Revision #1
Created 2026-07-26 17:06:56 UTC by art10m
Updated 2026-07-26 17:07:18 UTC by art10m