Design Patterns und GUI-Architekturen

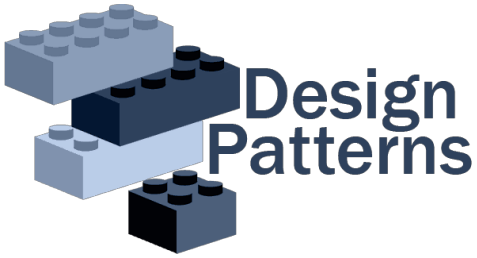
Design Patterns in der Programmierung

Ein Beispiel:

Jedes Design hat sein Pattern, alles eine Vorlage: Sei es eine Tasse, ein Wohnhaus oder ein Kleid. Niemand würde auf die Idee kommen, die Henkel einer Tasse innen anzubringen. Es hat sich für die praktische Nutzung einfach bewährt, dass dieser Bestandteil außen angebracht wird. Wenn Sie einen Töpferkurs besuchen und einen Henkeltopf töpfern möchten, ist Ihnen die grundsätzliche Form schon bekannt, sie ist sozusagen als Entwurfsmuster in Ihrem Kopf hinterlegt.

Ähnlich verhält es sich mit Computerprogrammen. Bestimmte Abläufe wiederholen sich immer wieder, sodass es nur ein kleiner Schritt zu der Idee war, auch hier so etwas wie Schnittmusterbögen anzulegen. In der Softwareentwicklung wird dabei von Design Patterns gesprochen.

Was genau sind Design Patterns?



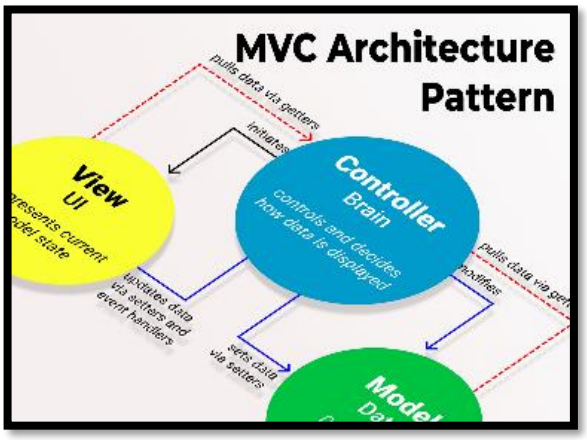
Um das oben gezeigte Beispiel aufzugreifen: Für jede Tasse werden immer wieder die gleichen Grundelemente benötigt: Boden, Wand und Henkel – egal ob es eine Kaffee-, Espresso- oder Teetasse werden soll. Ganz ähnlich verhält es sich beim Programmieren. Zum Beispiel: Zu durchlaufende Schleifen sind immer an Start- und Endvorgaben geknüpft; eine Bedingung erfordert immer eine Entscheidung, was bei Übereinstimmung und was bei Nichtübereinstimmung passiert; eine Berechnung gibt stets das Ergebnis der Kombination von Variablen aus usw. Aus vielen einzelnen Programmierschritten entsteht ein Programmablauf, der für bestimmte Aufgaben immer die gleichen Züge trägt. Entwurfsmuster (= Design Patterns) sind eine Beschreibung, wie ein Problem zu lösen ist.

Gleiches gilt für größere Zusammenhänge und Abläufe, die immer wieder eingesetzt werden, um bestimmte Aufgaben in Programmabläufen zu lösen. Somit sind Entwurfsmuster als Schemata zu verstehen, die sich bereits in der Praxis bewährt haben.

Die Möglichkeit, auf bewährte Lösungsansätze zurückzugreifen, geht mit einer Zeit- und Kostenersparnis einher. Entwicklerteams müssen nicht ständig das Rad neu erfinden, um eine bereits vielfach gelöste Teilaufgabe in einem neuen Programmablauf zu lösen.

Der Umgang mit Entwurfsmustern setzt sehr umfangreiches Wissen voraus. Die Verfügbarkeit von Design Patterns kann auch zu der Auffassung verführen, dass mit vorliegenden Entwurfsmustern scheinbar alle Probleme gelöst werden können. Kurz gesagt: die Kreativität kann eingeschränkt werden, die Neugier darauf, neue (bessere) Lösungsansätze zu finden könnte schwinden.

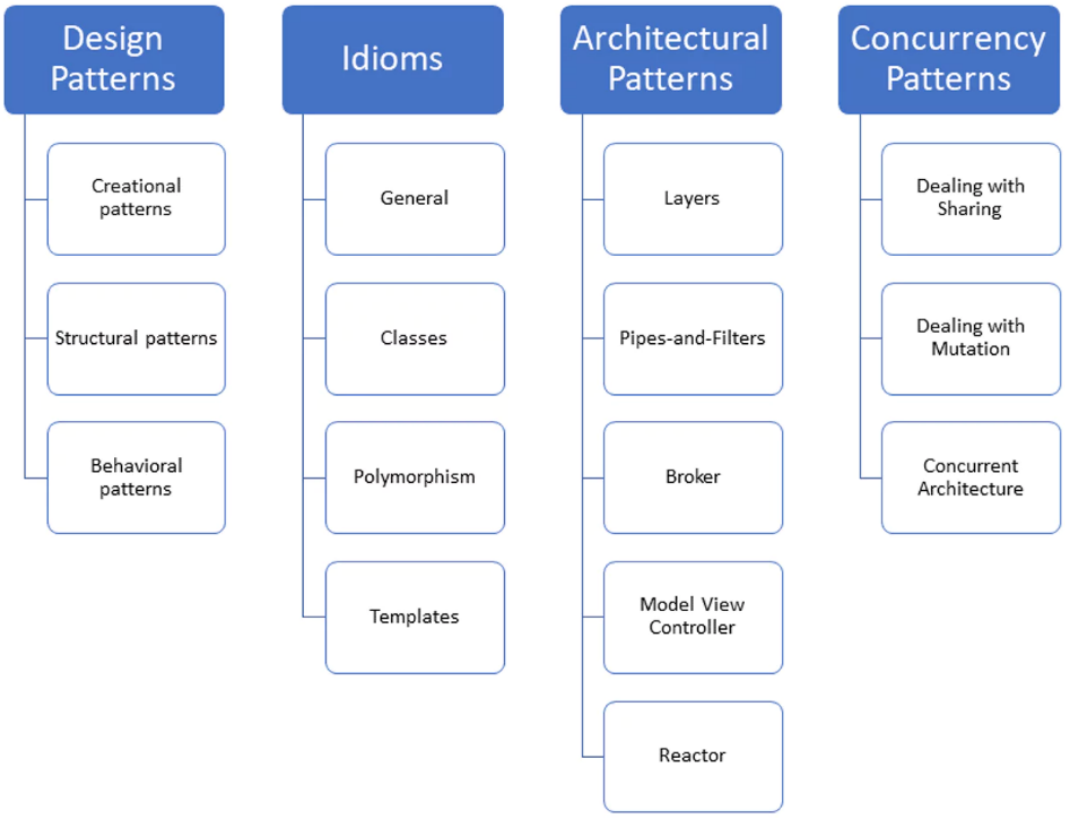
Das MVC-Pattern und seine Funktionsweise



Das Model-View-Controller (MVC) Pattern ist ein bewährtes Architekturmuster in der Softwareentwicklung, das dazu dient, die Trennung von Daten, Benutzeroberfläche und Steuerungslogik in einer Anwendung zu fördern. Dieses Muster ermöglicht eine effiziente Entwicklung, Wartung und Skalierbarkeit von Softwareanwendungen.

Hinweis:

Im Bereich der Softwareentwicklung sind **Architekturmuster** (englisch architectural patterns) anders einzuordnen als Entwurfsmuster. Im Gegensatz zu **Entwurfsmustern (Design-Patterns)** bestimmen sie nicht ein konkretes (meist kleines oder lokales) Teilproblem, sondern die grundlegende Organisation und Interaktion zwischen den Komponenten einer Anwendung. Bei MVC spricht man oft von einem Design-Pattern. Streng genommen ist es jedoch ein Architektur-Pattern.



https://www.heise.de/blog/Klassifikation-von-Mustern-in-der-Softwareentwicklung-7203402.html

Die MVC-Komponenten:

Das Model

Das Model repräsentiert die Daten und die Geschäftslogik einer Anwendung. Hier werden Datenbankabfragen, Berechnungen und jegliche Verarbeitung der Anwendungsdaten durchgeführt. Das Model ist unabhängig von der Benutzeroberfläche und dem Controller und bietet Schnittstellen, um auf die Daten zuzugreifen oder sie zu aktualisieren.

Die View

Die View ist für die Präsentation der Daten und die Benutzeroberfläche verantwortlich. Sie ist dafür zuständig, die Informationen aus dem Model anzuzeigen und ermöglicht dem Benutzer die Interaktion mit der Anwendung. Views können Textfelder, Schaltflächen, Listen, Diagramme und mehr enthalten. Eine wichtige Eigenschaft der View ist, dass sie passiv ist, was bedeutet, dass sie nicht direkt mit dem Model interagiert.

Der Controller

Der Controller ist die Schaltzentrale der Anwendung und fungiert als Vermittler zwischen dem Model und der View. Er empfängt Benutzereingaben aus der View, verarbeitet sie und leitet die entsprechenden Anfragen an das Model weiter. Der Controller enthält die Anwendungslogik, die die Daten im Model manipuliert und die View aktualisiert. Er sorgt für die Trennung von Daten und Benutzeroberfläche.

Grundsätzliche Funktionsweise des MVC-Musters:

1. Benutzereingabe: Der Prozess beginnt, wenn der Benutzer mit der Benutzeroberfläche interagiert, beispielsweise durch Klicken auf eine Schaltfläche.

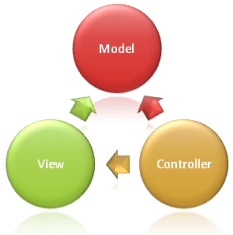
2. Controller-Verarbeitung: Der Controller fängt die Benutzereingabe ab und verarbeitet sie. Basierend auf der Art der Eingabe wird der Controller relevante Aktionen im Model auslösen.

3. Model-Aktualisierung: Das Model wird aktualisiert, um die angeforderte Aktion auszuführen. Dies kann das Aktualisieren von Daten, Berechnungen oder Datenbankabfragen beinhalten.

4. View-Aktualisierung: Nachdem das Model aktualisiert wurde, informiert der Controller die View darüber, dass eine Aktualisierung erforderlich ist. Die View liest die Daten aus dem Model und zeigt sie entsprechend an.

5. Benutzerfeedback: Der aktualisierte Zustand wird dem Benutzer in der View angezeigt, und der Benutzer erhält Feedback über seine Aktion.

Das MVC-Designmuster fördert die Trennung von Software-Komponenten, indem es die Daten, die Darstellung und die Steuerung voneinander isoliert. Diese Trennung erleichtert die Wartung und Erweiterbarkeit von Anwendungen, da Änderungen in einem Bereich nur minimale Auswirkungen auf die anderen Bereiche haben. Es ermöglicht auch die Wiederverwendung von Komponenten und eine verbesserte Testbarkeit der Software. Obwohl das Model-View-Controller (MVC) Design Pattern viele Vorteile bietet, hat es auch potenzielle Nachteile.



Mögliche Nachteile sind:

Komplexität: MVC kann die Struktur einer Anwendung komplexer machen, insbesondere für kleinere Projekte. Die Aufteilung in drei separate Komponenten erfordert zusätzlichen Code und Verwaltungsaufwand.

Overhead: In einigen Fällen kann MVC einen gewissen Overhead verursachen, da die Kommunikation zwischen Model, View und Controller koordiniert werden muss. Dies kann die Leistung beeinträchtigen, wenn es nicht sorgfältig implementiert wird.

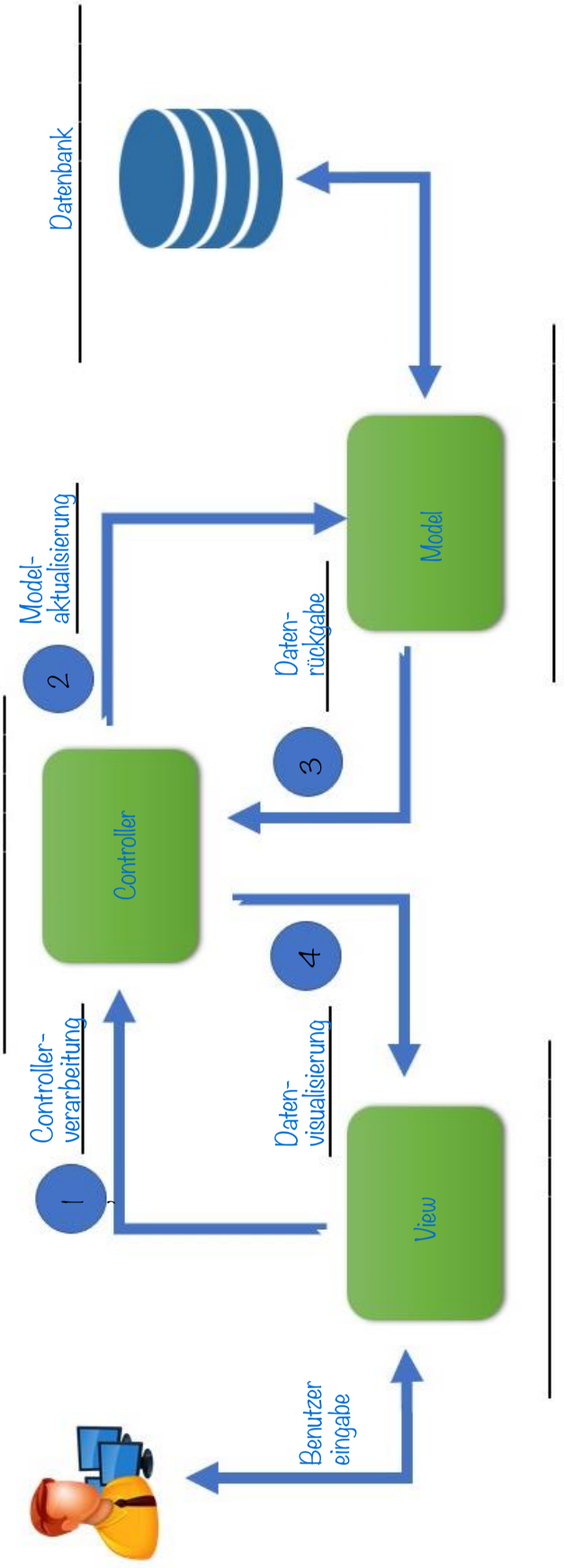
Trotz dieser potenziellen Nachteile bleibt das MVC-Designmuster ein nützliches Werkzeug, insbesondere in größeren und komplexeren Softwareprojekten, da es die Wartbarkeit, Skalierbarkeit und die Trennung von Software-Komponenten fördert. Insgesamt ist das MVC-Designmuster ein leistungsfähiges Werkzeug in der Softwareentwicklung, das dazu beiträgt, strukturierte und gut organisierte Anwendungen zu erstellen, die leichter zu warten und zu erweitern sind.

Füllen Sie die leeren Felder in der Grafik zum MVC-Pattern auf der nächsten Seite vollständig aus.

Runde Markierungen: Schritt-Nummer 1 bis 4 in der richtigen Reihenfolge eintragen

Rechtecke: Bezeichnungen der MVC-Komponenten eintragen

Linien: Generelle Aufgabe der jeweiligen Komponente eintragen


Datenbank
Model
Model-
aktualisierung
2
Daten-
rückgabe
3
Controller
4
Controller-
verarbeitung
1
Daten-
visualisierung
View
Benutzer
eingabe

Welche wesentlichen Eigenschaften haben die drei Komponenten des MVC-Patterns? Füllen Sie die Tabelle mit Stichpunkten.

Model	View	Controller
• Enthält Daten • Ist von View und Controller unabhängig • Kann Geschäftslogik enthalten	• Präsentationsebene • Sorgt für Darstellung der Daten des Models • Realisierung der Benutzerinteraktionen • Ist nicht für die Verarbeitung von Daten zuständig • Ist vom Controller unabhängig	• Verwaltet View und Model • Wird von View über Benutzeraktionen informiert

Beantworten Sie die folgenden Fragen zu MVC.

Warum ist die Trennung von Daten, Benutzeroberfläche und Steuerungslogik in einer Anwendung wichtig, wie sie im MVC-Designpattern umgesetzt wird?

- Wartung und Erweiterbarkeit der Anwendung wird erleichtert
- Mehrere Entwickler können gleichzeitig/unabhängig an unterschiedlichen Komponenten arbeiten
- Änderungen in einem der Bereiche hat minimale bis keine Auswirkungen auf andere Bereiche
- Modulartigkeit wird gefördert
- Verschiedene Teile der Anwendung können unabhängig voneinander entwickelt und getestet werden

Welche potenziellen Nachteile bringt das MVC-Pattern mit sich?

- Komplexität
- Overhead (bei unsaubererImplementierung)
- ...