MVC

Model – View – Controller

Ein Referat zur Vorlesung Softwaretechnische Grundlagen an der HHUD im SoSe 2012

Stefan Upietz

Übersicht

- MVC – WTF?
- Komponenten
- Zusammenspiel
- Vor- und Nachteile
- Weiterentwicklung und Umsetzung
- Fazit

MVC – WTF?

- Model – View – Controller
- Entwurfsmuster in der Softwareentwicklung
- Auch: MVC – Paradigma
- Erste Erwähnung in Smalltalk
- Entwicklung von Smalltalk-Spezifischem Ansatz zu allgemeinem Muster

MVC – WTF?

- Grundidee: Unterteilung der Software in
  - Daten
  - Präsentation
  - Interaktion
- Die Software wird in Module aufgeteilt, die je einer Domäne innerhalb des Musters zuzuweisen sind

Komponenten

- Model:
  - Spezifische, relevante und eigentliche Programmdaten
  - Kann je nach Ansatz auch die domain logic implementieren
  - Idealerweise komplett unabhängig von den anderen Programmteilen
  - Verschiedene Arten der Implementierung

Komponenten

- View:
  - Darstellung der Model-Daten sowie der UI-Elemente
  - Auf ein bestimmtes Medium spezialisiert
  - Einstiegspunkt für den Benutzer
  - Kommuniziert mit Model und Controller
  - Kann je nach Implementation z.B. als Listener des Models agieren

Komponenten

- Controller:
  - Umsetzung und Definition von Benutzeraktionen auf Model-Daten
  - Steuerung und Koordination der verschiedenen Views (Application Logic)
  - Ein (Sub)Controller pro View
  - Im Idealfall nur Programmsteuerung, d.h. Model und View kommunizieren direkt

Zusammenspiel

- Views stellen Daten dar und bieten Aktionen auf sie an
- Die Aktionen sowie die Navigation werden vom Controller ausgeführt
- Das Model führt Änderungen der Daten durch und propagiert sie

Vor- und Nachteile

- Vorteile
  - Klare Trennung von Daten, Präsentation und Programmlogik
  - Hohe Modularität
  - Hohe Wiederverwendarkeit
  - Medienunabhängige Datenbasis
  - Möglichkeit, in verschiedenen Medien gleiche Datenbasis einzusetzen (Parallelität)

Vor- und Nachteile

- Nachteile
  - Listener-Modell schwierig zu debuggen, da viele Aktionen nur zur Laufzeit passieren
  - Unterschied Programmlogik – Domänenlogik nicht immer trennscharf
  - View und Controller sollten eigenständig sein, dies ist aber selten klar zu trennen

Weiterentwicklung und Umsetzung

- Trennung von View – Controller immer weniger
- Daraus resultierend die Trennung von Inhalt – Präsentation z.B. in CMS
- Ruby on Rails
- Android-Programmierung (Java)

Fazit

- Zugrundeliegende Idee MVC bald von Smalltalk entkoppelt und als Designmuster etabliert
- Nach Bedarf muss gegen Prinzipien verstoßen werden im Sinne der Praktikabilität
- OO-Sprachen begünstigen Umsetzung
- Immer noch moderne Ansatz, gerade mit steigender Anzahl von Medien

Fragen?

Vielen Dank für eure Aufmerksamkeit.