Netzwerk-Ökonomie

Effekte und Strategien

https://iuk.one/1066-1914

Clemens H. **Cap**
ORCID: 0000-0003-3958-6136

Department of Computer Science
University of **Rostock**
Rostock, Germany
clemens.cap@uni-rostock.de

Version 1

1. Patterns

Muster bei der Entwicklung von Frontends

Model View Controller

Bestandteile:

- **Model**: Enthält Anwendungslogik
- **View(s)**: Enthalten Visualisierung oder abstraktere Darstellung
- **Controller**: Verändern das Model

Vorteile:

- Separation of Concerns
- Konsistenz von Model und View leicht zu gewährleisten
- Multi-View Darstellungen einfach zu entwickeln

Model View Controller in Web-Anwendungen

Das Model lebt am Server:

- Problem der Round trip Latenz
- View kann (Server-side) Model nicht direkt beobachten, da remote
- Benötige Protokoll zwischen Server und Client
- Wo wir gerendert? (Server oder Client)
- Was wird übertragen? (Html oder Json oder separate PDUs)

Das Model lebt am Client:

- Zusätzliches Konsistenz-Problem Client-Model zu DB-Model
- Oft keine genügend klare Trennung von View und Controller

Generell:

- Viele offene Aspekte (Bsp: Rolle der Template Engine)
- Oft unterschiedliches Verständnis von der genauen Funktionsweise
- Oft unklare Formalisierung

Model View Controller Vorstellung (1)

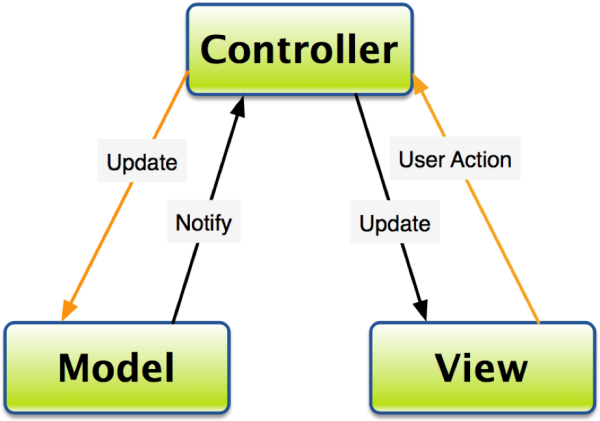

Abb. 1: 1. Art von MVC

Model View Controller Vorstellung (2)

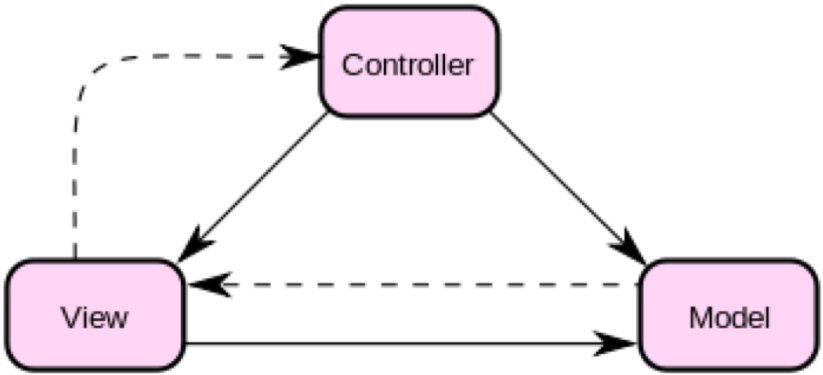

Abb. 2: 2. Art von MVC

Model View Controller Vorstellung (3)

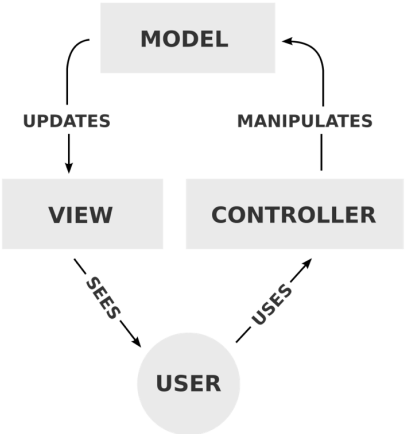

Abb. 3: 3. Art von MVC

Model View Controller Vorstellung (4) – Beste Darstellung

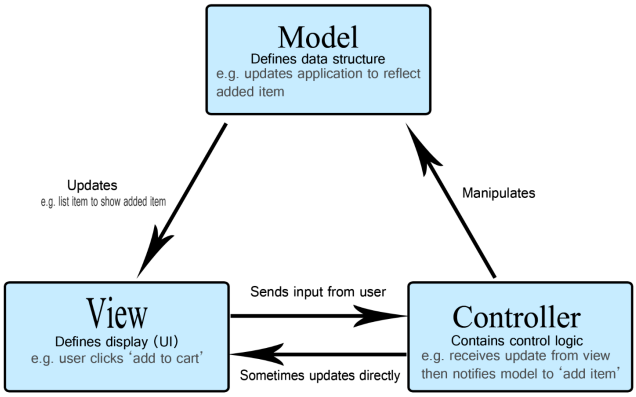

Abb. 4: 4. Art von MVC: Beste Variante

Model View Presenter

Vorgehensweise:

- Stärkere Trennung von Model und View
- Presenter: Ist die Koppelung von M und V
- Verbessert separate Testbarkeit

Problem:

- Die starke Entkoppelung ist nicht immer wünschenswert
- Bsp: Implementierung der Listen im TWBK Chat auf Mobiles
- Der Umweg über das Model kann zu aufwendig sein

Beispiele: Google Web Toolkit, Swing, Laravel

Model View Presenter

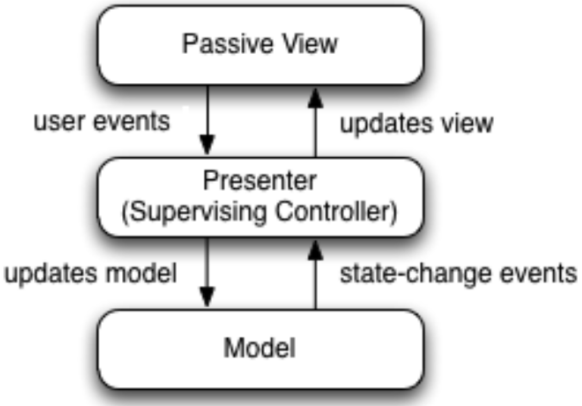

Abb. 5: Model View Presenter

Model View ViewModel

Vorgehensweise:

- Verdoppelung des Model Konzepts
  - **Model**: Model für Geschäftslogik
  - **ViewModel**: Model für die Bedienlogik
- ViewModel vermittelt zwischen Model und View
- Model und View sind nicht direkt verbunden
- Erlaubt bessere Anpassung an die Situation im Web
- Häufig in Web-Frameworks zu finden

Beispiele: Angular, Ember, React, Ext JS

2. Data Binding

Deskriptive Koppelung von Model und View

Data Binding (1)

Ziel: Deskriptive Koppelung von Model und View

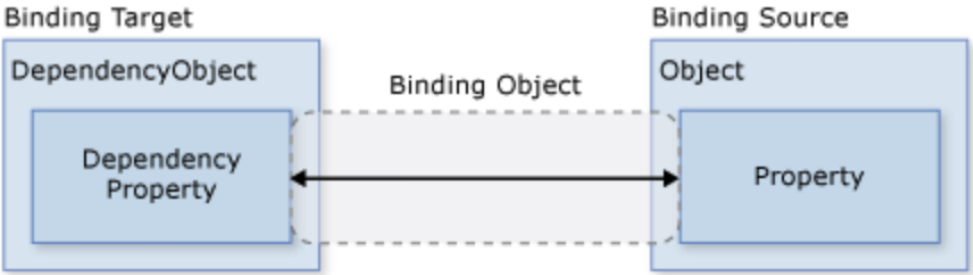

Abb. 6: Bestandteile einer Data-Binding Relation

Data Binding (2)

- **Unidirektional**: Model Änderung propagiert in View
- **Bidirektional**: Änderung im View propagiert in Model

Je nach Struktur des Models:

- Synchronisation View mit client-seitigen Caches
- Synchronisation View mit server-seitiger DB

Guter Übersichtsartikel:
https://learn.microsoft.com/en-us/dotnet/desktop/wpf/data/

Umsetzung von Data Binding (1)

Variante 1: Dirty Checking

- Regelmäßiger Vergleich alter und neuer Model Objekte
- Bei hoher Änderungsrate aber effizient

Variante 2: Objekt Beobachtung:

- Setzen von Traps auf Objektveränderung
- Bei hoher Änderungsrate wird ev. zu oft nachgeführt

Umsetzung von Data Binding (2)

Variante 3: Datenfluß-Analyse

- Dynamisches Implementieren einer Datenfluß-Analyse
- Kann gut durch Proxy Objekte unterstützt werden
- Punktgenaues Nachführen des jeweils angekoppelten Models

Server-seitige Frameworks mit Data Binding

Express:

- Schlankes Framework für Node.js
- Template Engine Jade und Pug
- Standard-Anbindung für weitere Template Engines

Sail.js:

- Modulares Framework, das Express-kompatibel ist
- Sehr flexibles DB Mapping über den Mapper Waterline
- DB-Zugang und Sicherheit über deklarativen Policies Ansatz
- Beliebige Frontends können eingekoppelt werden

Client-seitige Frameworks mit Data Binding

Angular

- Databinding über regelmäßiges Dirty Checking
- Templating über ng-* Attribute in die DOM eingebunden
- Scaffolding für einige Teile der Anwendung

Ember

- Databinding durch direkte Anbindung an das Objekt
- Handlebar Template Engine
- Stark opinionated
- Scaffolding für viele Teile der Anwendung

Sehr ausführlicher Vergleich: http://angularjs-emberjs-compare.bguiz.com/

3. Weitere Libraries und Frameworks

Hohe Anzahl

Erinnert an den Turmbau zu Babel

Frontend Libaries

Bootstrap

- Responsives Design für unterschiedliche Formfaktoren
- Ursprünglich von Twitter eingesetzt
- Nutzt neuere CSS Features

Framework7

- Emuliert natives Aussehen von iOS und Android mit HTML
- Sehr gute gelungene optische Umsetzung

jQuery UI

- Geringere Plattform-Treue als Framework7.
- Gute Modularisierung via Plug In Architektur

Sehr viele weitere Libraries...

Welches Framework einsetzen?

Hohe Anzahl von Frameworks:

https://en.wikipedia.org/wiki/Comparison_of_JavaScript-based_web_frameworks

Kriterien zur Auswahl

http://angularjs-emberjs-compare.bguiz.com/intro/questions.html

4. UI und DOM Anpassung

Wie gleichen wir praktisch den View an das Model an?

UI / Dom Anpassung

Problem:

- Model ändert sich – wie führe ich rasch das UI nach
- Im Web: Wie führe ich die DOM nach?
- DOM kann schwergewichtig sein (Layout Mechanismus)

Direktes Rendern

Vorgehensweise:

- DOM komplett neu rendern

Problem: Server-Roundtrip Time (wenn am Server gerendert)

Frage: Wann rendere ich neu?

- Sofort nach jeder Model-Änderung: Ev. zu oft
- Nur periodisch: Sehe gewisse Zeit ältere Daten

Frage: Wie detektiere ich Erfordernis, das UI neu zu rendern?

- Modell kann anders sein, View kann aber gleich sein

Übergänge

Bei größeren Übergängen oder langsameren Endgeräten mögliches Problem.

Lösung: Double Buffering

- Einen zweiten Layer anlegen
- Jeweils in den unsichtbaren Layer rendern
- Sichtbarkeit der Layer wechseln

Virtual Dom

Vorgehensweise:

- DOM außerhalb des Dokuments (virtuell) nachbilden
- DOM virtuell rendern
- Bei Änderung:
  - neues Dokument ebenso virtuell rendern
  - die Differenz der virtuellen DOMs ermitteln
  - Nur die Differenzoperatoren tatsächlich im DOM ausführen

Beispiele:

- React

Incremental Dom

Vorgehensweise:

- Variante der Virtual Dom
- Ausgangspunkt für die Differenzbildung ist die echte DOM

Bewertung:

- Etwas langsamer bei geringerem Speicherbedarf

Beispiele:

- Google incremental-dom

Anhang

Übersicht

Verzeichnis aller Abbildungen

Rechtliche Hinweise (1)

Die hier angebotenen Inhalte unterliegen deutschem Urheberrecht. Inhalte Dritter werden unter Nennung der Rechtsgrundlage ihrer Nutzung und der geltenden Lizenzbestimmungen hier angeführt. Auf das Literaturverzeichnis wird verwiesen. Das Zitatrecht in dem für wissenschaftliche Werke üblichen Ausmaß wird beansprucht. Wenn Sie eine Urheberrechtsverletzung erkennen, so bitten wir um Hinweis an den auf der Titelseite genannten Autor und werden entsprechende Inhalte sofort entfernen oder fehlende Rechtsnennungen nachholen. Bei Produkt- und Firmennamen können Markenrechte Dritter bestehen. Verweise und Verlinkungen wurden zum Zeitpunkt des Setzens der Verweise überprüft; sie dienen der Information des Lesers. Der Autor macht sich die Inhalte, auch in der Form, wie sie zum Zeitpunkt des Setzens des Verweises vorlagen, nicht zu eigen und kann diese nicht laufend auf Veränderungen überprüfen.

Alle sonstigen, hier nicht angeführten Inhalte unterliegen dem Copyright des Autors, Prof. Dr. Clemens Cap, ©2020. Wenn Sie diese Inhalte nützlich finden, können Sie darauf verlinken oder sie zitieren. Jede weitere Verbreitung, Speicherung, Vervielfältigung oder sonstige Verwertung außerhalb der Grenzen des Urheberrechts bedarf der schriftlichen Zustimmung des Rechteinhabers. Dieses dient der Sicherung der Aktualität der Inhalte und soll dem Autor auch die Einhaltung urheberrechtlicher Einschränkungen wie beispielsweise Par 60a UrhG ermöglichen.

Die Bereitstellung der Inhalte erfolgt hier zur persönlichen Information des Lesers. Eine Haftung für mittelbare oder unmittelbare Schäden wird im maximal rechtlich zulässigen Ausmaß ausgeschlossen, mit Ausnahme von Vorsatz und grober Fahrlässigkeit. Eine Garantie für den Fortbestand dieses Informationsangebots wird nicht gegeben.

Die Anfertigung einer persönlichen Sicherungskopie für die private, nicht gewerbliche und nicht öffentliche Nutzung ist zulässig, sofern sie nicht von einer offensichtlich rechtswidrig hergestellten oder zugänglich gemachten Vorlage stammt.

Use of Logos and Trademark Symbols: The logos and trademark symbols used here are the property of their respective owners. The YouTube logo is used according to brand request 2-9753000030769 granted on November 30, 2020. The GitHub logo is property of GitHub Inc. and is used in accordance to the GitHub logo usage conditions https://github.com/logos to link to a GitHub account. The Tweedback logo is property of Tweedback GmbH and here is used in accordance to a cooperation contract.

Rechtliche Hinweise (2)

Disclaimer: Die sich immer wieder ändernde Rechtslage für digitale Urheberrechte erzeugt ein nicht unerhebliches Risiko bei der Einbindung von Materialien, deren Status nicht oder nur mit unverhältnismäßig hohem Aufwand abzuklären ist. Ebenso kann den Rechteinhabern nicht auf sinnvolle oder einfache Weise ein Honorar zukommen, obwohl deren Leistungen genutzt werden.

Daher binde ich gelegentlich Inhalte nur als Link und nicht durch Framing ein. Lt EuGH Urteil 13.02.2014, C-466/12 (Pressemitteilung, Blog-Beitrag, Urteilstext). ist das unbedenklich, da die benutzten Links ohne Umgehung technischer Sperren auf im Internet frei verfügbare Inhalte verweisen.

Wenn Sie diese Rechtslage stört, dann setzen Sie sich für eine Modernisierung des völlig veralteten Vergütungs- und Anreizsystems für urheberrechtliche Leistungen ein. Bis dahin klicken Sie bitte auf die angegebenen Links und denken Sie darüber nach, warum wir keine für das digitale Zeitalter sinnvoll angepaßte Vergütungs- und Anreizsysteme digital erbrachter Leistungen haben.

Zu Risiken und Nebenwirkungen fragen Sie Ihren Rechtsanwalt oder Gesetzgeber.

Weitere Hinweise finden Sie im Netz hier und hier oder hier.

Zitierweise dieses Dokuments

Wenn Sie Inhalte aus diesem Werk nutzen oder darauf verweisen wollen,
zitieren Sie es bitte wie folgt:

Clemens H. Cap: Netzwerk-Ökonomie. Electronic document. https://iuk.one/1066-1914 12. 6. 2024.

Bibtex Information: https://iuk.one/1066-1914.bib

```
@misc{doc:1066-1914,
    author       = {Clemens H. Cap},
    title        = {Netzwerk-Ökonomie},
    year         = {2024},
    month        = {6},
    howpublished = {Electronic document},
    url          = {https://iuk.one/1066-1914}
}
```

Typographic Information:
Typeset on ?today?
This is pdfTeX, Version 3.14159265-2.6-1.40.21 (TeX Live 2020) kpathsea version 6.3.2
This is pgf in version 3.1.5b
This is preamble-slides.tex myFormat©C.H.Cap

Verzeichnis aller Folien

Legende:
Fortsetzungsseite
Seite ohne Überschrift
Bildseite