



openHPI: **Web-Technologien** **Web-Frameworks**

Prof. Dr. Christoph Meinel

Hasso-Plattner-Institut

Universität Potsdam

Entwicklung serverseitiger Web-Programme

- **CGI:** HTML-Dokument wird vom CGI-Programm generiert, z.B.
 - `print("<h1>Titel</h1>");`
- **Serverseitiges Scripting:** Spezielles Skript-Markup innerhalb von (sonst statischen) HTML-Dateien, z.B.
 - `<ul>`
 `<?php foreach ($items as $item): ?>`
 `<li><?php echo $item->description ?></li>`
 `<?php endforeach ?>`
 `</ul>`
- **Problem:** Vermischung verschiedener Aufgabenbereiche in selber Datei
 - Unübersichtlicher, schwer wartbarer Code
 - Designer und Programmierer arbeiten an derselben Datei
 ➔ Konfliktpotential

Lösungsansätze:

■ Template-Engines:

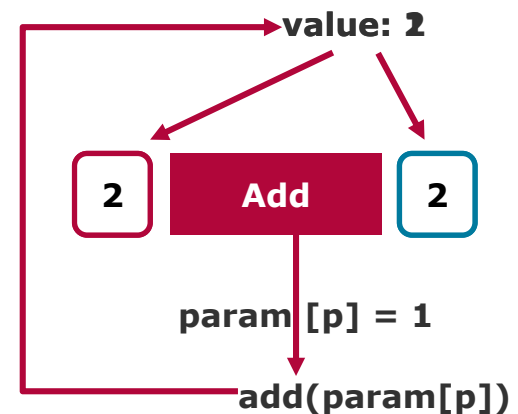
- Trennung von Programm-Logik und Darstellung

■ Moderne Web-Frameworks:

- weitergehende Aufsplitterung von Verantwortlichkeiten im Programm
- oft Anwendung von MVC-(ähnlichen)-Entwurfsmustern.

MVC: Model-View-Controller

- Model: Datenstruktur und Businesslogik
- View: Darstellung
- Controller: verbindet Model und View



Warum Frameworks? (1/2)

Frameworks ...

- Bieten Grundgerüst mit architektonischen Vorgaben für die Entwicklung von Anwendungen („framework“: englisch für *Gerüst*)
 - gute und bewährte Abstraktionen
- Stellen Lösungen bereit für Probleme, die immer wieder gelöst werden müssen
- Enthalten Bibliotheken für typische Anwendungsfälle
- Erlauben beschleunigtes Programmieren
 - Code-Generatoren
 - Konventionen

Warum Frameworks? (2/2)

Web-Frameworks ...

- Bieten Routing Pfade, die mit dem verarbeitenden *Controller* verbinden, z.B.
 - `/courses/webtech2017` → **CoursesController** mit ID `webtech2017`
- Bieten Persistenz-Layer für den Datenbank-Zugriff und für die Abbildung der Anwendungslogik
- Enthalten Template-Engines für die Generierung von HTML
- Gehen mit HTTP um, Request/Response-Zyklen, Headers, Middleware, ...
- Enthalten Bibliotheken für
 - Mail-Versand
 - Caching
 - Verarbeitung von langwierigen Prozessen im Hintergrund (*Queues*)
 - ...

Frameworks – Vorteile

Vorteile

- Frameworks ermöglichen **einfachere, schnellere** Programmierung
- Bauen auf der Erfahrung und den Tools anderer auf
- Bieten Lösungen für häufige Probleme
 - meist effizient, stabil und **sicher**
- Architektur wird vorgegeben
- Open-Source: Die meisten Frameworks sind frei verfügbar
 - viele Augen sehen mehr Fehler
 - Verbesserungen sind gern gesehen

Frameworks – Nachteile

Nachteile

- Frameworks sind nicht für die Ewigkeit und ändern sich oft
 - macht oft Veränderungen am eigenen Code nötig
- Eigener Programmcode oft an den Code des Frameworks **gekoppelt**
- Frameworks lösen viele Probleme, aber nicht alle
 - wer außerhalb der vorgegebenen Grenzen gerät, muss oft „gegen das Framework kämpfen“
- Frameworks und „Magie“:
 - ohne Kenntnis der Konventionen des Frameworks ist Code oft schwer nachvollziehbar (insbesondere für Neueinsteiger)
 - (insb. Ruby on Rails:) Was kann die Programmiersprache?
Was ist vom Framework?

Frameworks – Empfehlungen

Empfehlungen

- Erst die Programmiersprache lernen, dann ein Framework!
 - Magie des Frameworks von Funktionen der Sprache unterscheiden
 - **Probleme kennen, die das Framework löst, um dessen Lösungen wertschätzen zu können**
- Frameworks garantieren keinen sauberen Code
 - Sinn und Zweck der Architekturmuster kennen lernen
- Vor- und Nachteile abwägen
 - Frameworks beschleunigen die Entwicklung (insb. zu Beginn), aber erschweren Quereinstieg
 - Entscheidung für sehr lange Zeit: benutzt man ein Framework, wird man es nicht sehr schnell wieder los

Frameworks – Lizenzen (1/2)

- Open-Source-Software: Freie Verwendung unter durch **Lizenzen** geregelten Bedingungen
- **Flexible Lizenzen**
 - Beispiele: MIT, BSD, Apache
 - normalerweise untereinander kompatibel
- **Restriktive Lizenzen**
 - Beispiel: GPL
 - Abgeleitete Software muss ebenfalls unter GPL lizenziert werden
 - Verwendung von GPL-Bibliotheken kann dazu führen, dass eigener Code veröffentlicht wird
 - **AGPL**: Schließt Schlupfloch für serverseitige Applikationen

Frameworks – Lizenzen (2/2)

- Open-Source-Software-Lizenzen sind oft schwer zu verstehen
- Einfacherer Ansatz: Creative Commons:
 - <https://creativecommons.org/>
- Einsatzgebiete
 - weniger im Softwarebereich
 - weit verbreitet im Grafikbereich
 - Beispiel: <https://www.flickr.com/creativecommons/>
- Elemente
 - Attribution, Share-Alike, No-Derivs, Non-Commercial
 - beliebig kombinierbar
 - einfache Kennzeichnung

