



Kapitel 1: rails new

MScIS Peter Grosskopf
MScIS Thomas Hollstegge

Fachbereich Wirtschaft - Fachhochschule Münster
Bachelor of Science Wirtschaftsinformatik Wintersemester 2013/2014

Was ist Ruby?

- Dynamische, objektorientierte Programmiersprache
- Beinhaltet Prinzipien der funktionalen Programmierung
- Inspiriert von Lisp, Smalltalk, Perl, ...

„Ruby is designed to make programmers happy“
– Yukihiro Matsumoto

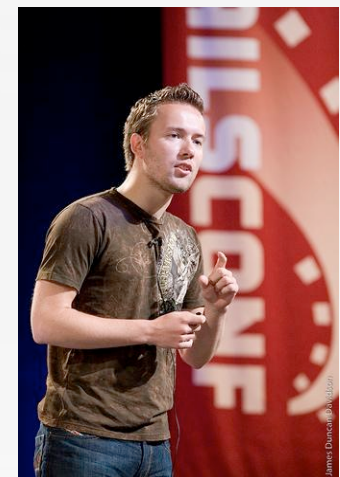
Einführung in Ruby



Programming Ruby 1.9 & 2.0 (4th edition):
The Pragmatic Programmers' Guide by Dave Thomas with Chad Fowler and Andy Hunt

Was ist Rails?

- Web-Entwicklungsframework
- Basiert auf der Programmiersprache Ruby
- Ermöglicht schnellen Einstieg in die Webentwicklung
- Ermöglicht schnelles Prototyping von Anwendungen
- 2004 vorgestellt von David Heinemeier Hansson



Prinzipien von Rails

"Don't repeat yourself" (DRY) -

Design-Prinzip für Software-Architekturen, das besagt, dass Informationen möglichst nicht redundant an mehreren Stellen im Quellcode vorgehalten werden sollen.

"Every piece of knowledge must have a single, unambiguous, authoritative representation within a system." (Hunt & Thomas, The Pragmatic Programmer, 1999)

Prinzipien von Rails

"Convention over Configuration" (CoC) -

Design-Prinzip für Software-Komponenten, das darauf abzielt, durch gut gewähltes Standardverhalten die Arbeit des Entwicklers im "Normalfall" zu minimieren, ohne jedoch die für besondere Fälle notwendige Flexibilität einzubüßen.

Rails trifft vernünftige Annahmen und benötigt keine aufwändige Konfiguration

Installation von Rails

- Rails wird als Rubygem ausgeliefert
- Installation mit `$ gem install rails`

Rubygems

Als Rubygem (oder kurz "**Gem**") bezeichnet man Softwarepakete, die Ruby-Applikationen oder -Bibliotheken enthalten. Das zentrale Verzeichnis aller verfügbaren Gems befindet sich unter <http://rubygems.org/>.

Erzeugen einer Rails-Anwendung

`$ rails new blog`

Erzeugt eine neue Rails-Anwendung mit Namen **blog** und installiert benötigte Pakete

Scaffolding

Scaffolding (deut. Gerüstbau) ist eine Technik der Meta-Programmierung, bei der automatisch ein **Quellcode-Grundgerüst** mit gewissen Basisfunktionalitäten generiert wird. Dieses kann und soll dann vom Entwickler angepasst und erweitert werden.

Grundgerüst einer Rails-Anwendung

app/	Enthält "Controller", "Models", "Views", "Helpers" und "Assets", Großteil der Funktionalität
config/	Enthält Konfigurationsdateien
config.ru	Rack-Konfiguration
db/	Datenbankschema, Migrationen
bin/	Binaries
Gemfile(.lock)	Abhängigkeiten der Anwendung von externen Paketen (Gems)
lib/	Erweiterte Module der Anwendung
log/	Logfiles der Applikation
public/	Statische Dateien, die ohne Weiterverarbeitung direkt ausgeliefert werden
Rakefile	Ermöglicht das Starten von einmaligen Tasks, die innerhalb der Anwendung ausgeführt werden
README.rdoc	Kurze Einleitung zur Anwendung
test/	Unit Tests, Integration Tests
tmp/	Temporäre Dateien
vendor/	Code von Drittanbietern

Starten einer Rails-Anwendung

WEBrick:

- Spezieller Webserver zur Entwicklung
- Im Lieferumfang der Rails-Konfiguration enthalten
- **Start:** `$ rails server`
- **Stopp:** `CTRL + C`



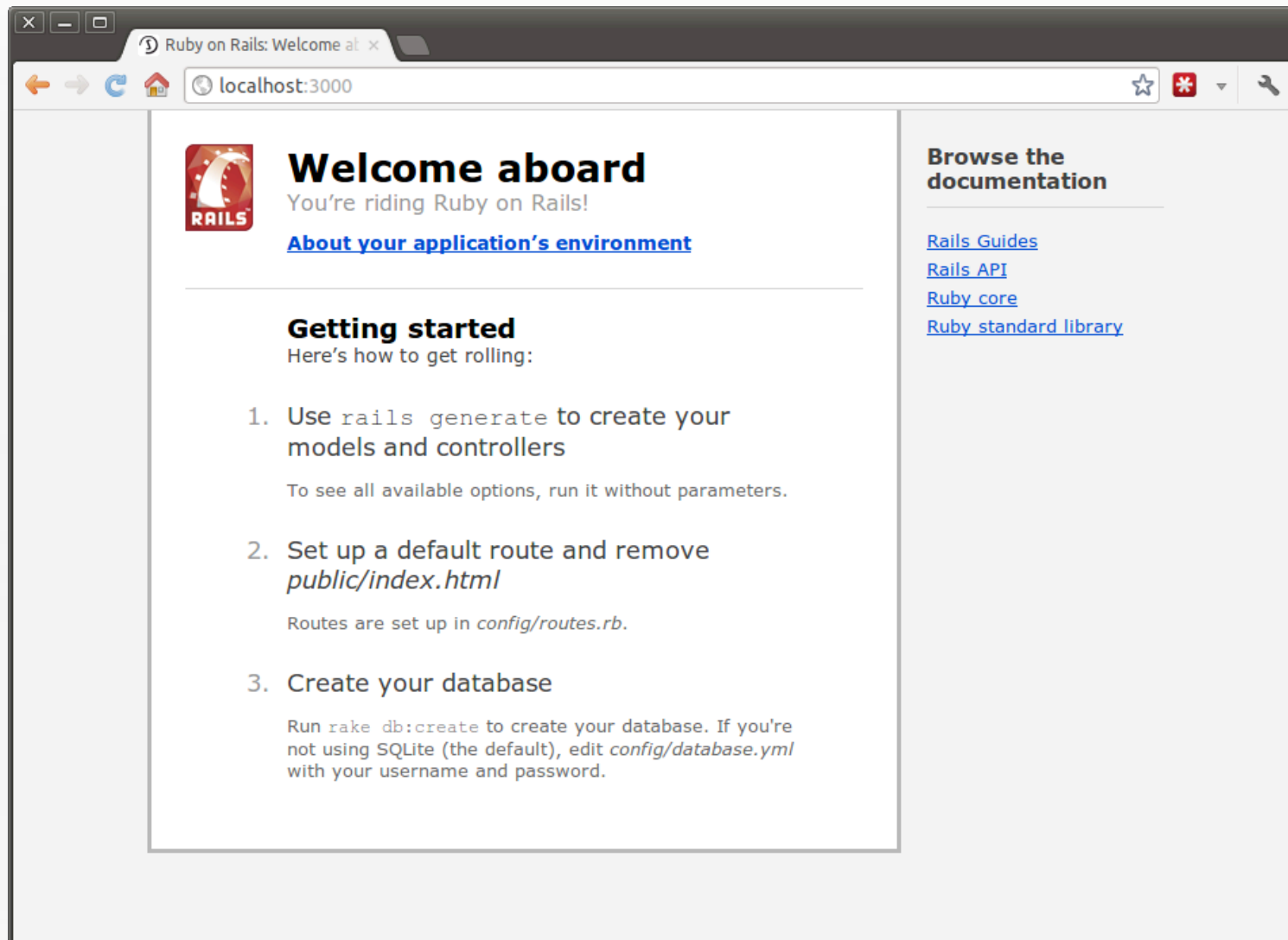
```
thomas@t420: rails server
thomas@t420 ~/blog % rails server
=> Booting WEBrick
=> Rails 3.2.8 application starting in development on http://0.0.0.0:3000
=> Call with -d to detach
=> Ctrl-C to shutdown server
[2012-10-13 16:39:27] INFO  WEBrick 1.3.1
[2012-10-13 16:39:27] INFO  ruby 1.9.3 (2012-02-16) [x86_64-linux]
[2012-10-13 16:39:27] INFO  WEBrick::HTTPServer#start: pid=30589 port=3000
```

Starten einer Rails-Anwendung

WEBrick Eigenschaften:

- Erreichbar unter <http://localhost:3000/>
- Keine aufwändige Konfiguration nötig
- Automatische Erkennung von Änderungen im Quellcode, daher meist kein Neustart nötig
- Nicht geeignet für Produktivsysteme!

Welcome!



Generieren eines Controllers

Controller

Zweck eines **Controllers** ist es, Anfragen an die Anwendung entgegen zu nehmen und Informationen für die Anzeige zu sammeln.

Syntax:

```
$ rails generate controller name action1 action2 [...]
```

name: Name des Controllers

action1, action2: Name von sog. Actions, die der Controller verarbeiten soll

Generieren eines Controllers

Beispiel:

```
thomas@t420: ~/blog
thomas@t420 ~/blog % rails generate controller welcome index
  create  app/controllers/welcome_controller.rb
  route  get "welcome/index"
        erb
  create  app/views/welcome
  create  app/views/welcome/index.html.erb
        test_unit
  create  test/functional/welcome_controller_test.rb
        helper
  create  app/helpers/welcome_helper.rb
        test_unit
  create  test/unit/helpers/welcome_helper_test.rb
        assets
        coffee
  create  app/assets/javascripts/welcome.js.coffee
        scss
  create  app/assets/stylesheets/welcome.css.scss
thomas@t420 ~/blog %
```

Anpassen einer View

View

Views bereiten von Controllern gesammelte Informationen für die Anzeige in einem lesbaren Format (z.B. HTML) auf.

- Views enthalten HTML-Code, der durch Vorlagen (Templates) erzeugt wird
- Views enthalten keine Programmlogik, sondern zeigen Informationen lediglich an!

Anpassen einer View

Beispiel:

A code editor window with a dark theme. The title bar at the top shows three small circles on the left and the file path 'app/views/welcome/index.html.erb' on the right. The main area of the editor is dark gray and contains the text '<h1>Hello, Rails!</h1>' in a light gray font. The opening and closing tags of the h1 element are highlighted in orange.

```
app/views/welcome/index.html.erb<h1>Hello, Rails!</h1>
```


Einbindung in die Applikation (Routing)

Routing

Das Routing einer Anwendung entscheidet, **welcher Controller** und **welche Action** für die Verarbeitung der Anfrage verantwortlich ist.

Das Routing wird in **config/routes.rb** definiert.

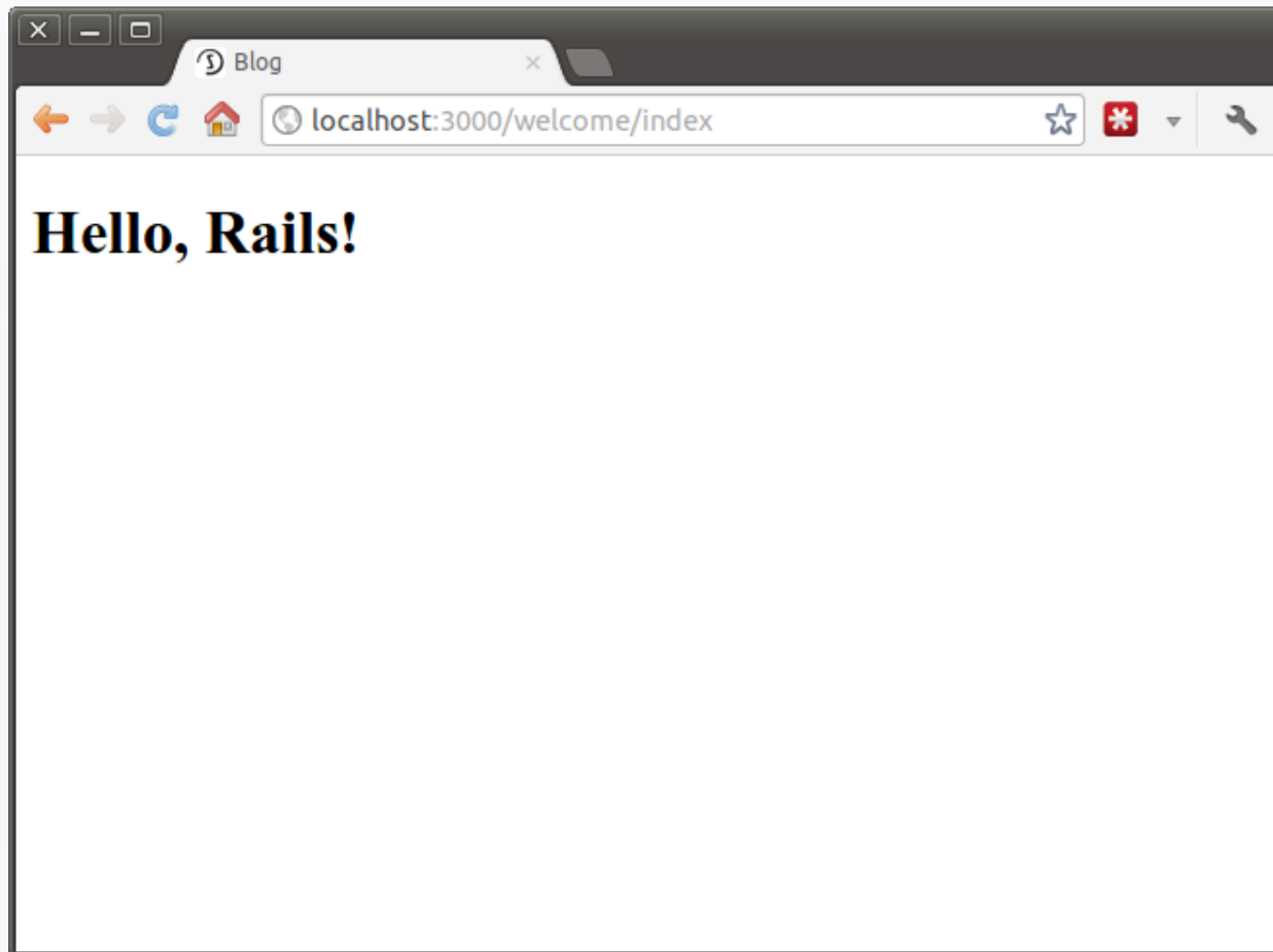
Einbindung in die Applikation (Routing)

```
config/routes.rb

Blog::Application.routes.draw
do
  get "welcome/index"
  root :to => "welcome#index"
end
```

- Anwendung soll auf **HTTP-GET**-Requests mit dem Pfad **welcome/index** reagieren
 - Standardverhalten:
Controller - welcome, **Action**: index (= "**welcome#index**")
- Die Startseite (/) wird auf "**welcome#index**" gemapped

Ergebnis





Architektur einer Rails-Anwendung

Model-View-Controller



Model-View-Controller (MVC)

Model-View-Controller (MVC) ist ein **Architekturmuster** zur Strukturierung von Software-Anwendungen mit Benutzer-Interaktion. Es wurde 1979 von dem norwegischen Informatiker Trygve M. H. Reenskaug vorgestellt und ist heute ein De-Facto-Standard für den Entwurf von komplexen Software-Systemen.

MVC - Allgemein

Model

- Verwaltung des internen Zustands der Anwendung
- Speicherung und Manipulation der Anwendungs-Daten

View

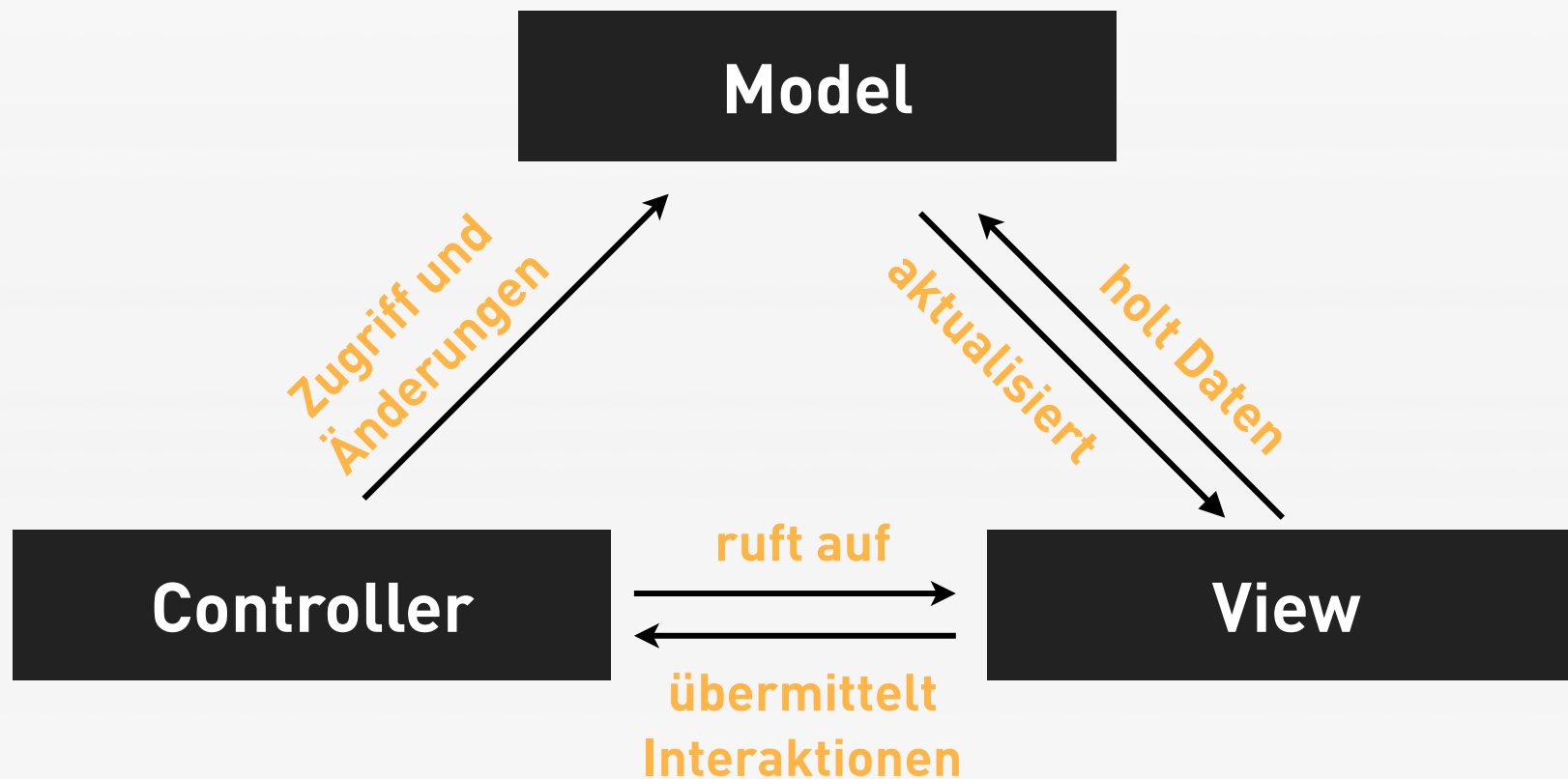
- Aufbreitung der Anwendungs-Daten zur Ansicht
- Bereitstellung der Benutzer-Oberfläche

MVC - Allgemein

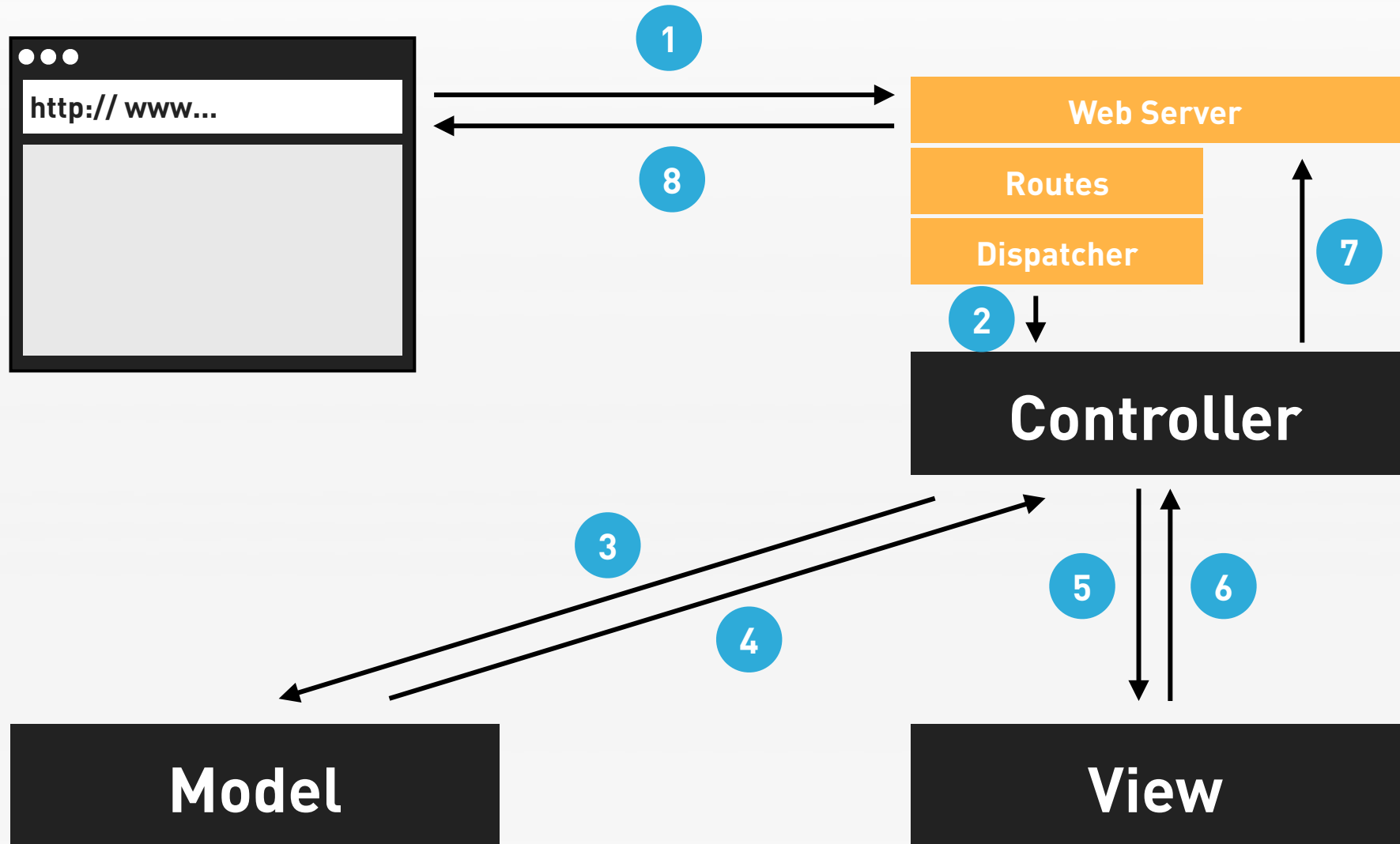
Controller

- Verarbeitung von Ereignissen (z.B. bei Nutzerinteraktion)
- Aufruf der Model-Funktionen
- Weiterleitung der Anwendung-Daten an den View

MVC - Allgemein



MVC - Realisierung in Rails





Rails-Prinzipien

"Don't repeat yourself" (DRY) -

Design-Prinzip für Software-Architekturen, das besagt, dass Informationen möglichst nicht redundant an mehreren Stellen im Quellcode vorgehalten werden sollen.

"Every piece of knowledge must have a single, unambiguous, authoritative representation within a system." (Hunt & Thomas, The Pragmatic Programmer, 1999)

Rails-Prinzipien

"Convention over Configuration" (CoC) -

Design-Prinzip für Software-Komponenten, das darauf abzielt, durch gut gewähltes Standardverhalten die Arbeit des Entwicklers im "Normalfall" zu minimieren, ohne jedoch die für besondere Fälle notwendige Flexibilität einzubüßen.

Rails trifft vernünftige Annahmen und benötigt keine aufwändige Konfiguration