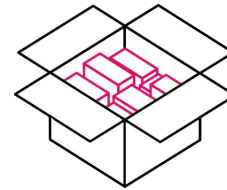


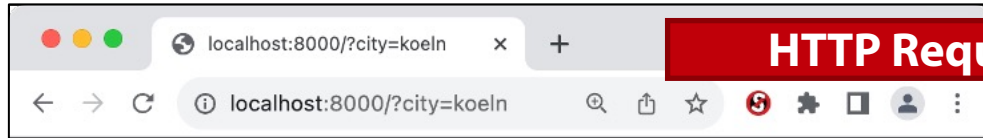
Einführung in die Server-seitige Webprogrammierung mit Spring

Hoai Viet Nguyen – TH Köln

Technology
Arts Sciences
TH Köln



Dynamische Webinhalte - SimpleWeatherServer



HTTP Request

```
fun main(args: Array<String>) {  
    val server = HttpServer.create(InetSocketAddress(8000), 0)  
    server.createContext("/", MyHandler())  
    server.start()  
}  
  
class MyHandler : HttpHandler {  
    override fun handle(httpExchange: HttpExchange) {  
        val city = getQueryByName(httpExchange.requestURI.query, "city")  
        val degree = getDegreeByCity(city)  
        val payload = "$degree Degree in $city"  
        httpExchange.responseHeaders.add("Content-Type", "text/html")  
        httpExchange.sendResponseHeaders(200, payload.length.toLong())  
        val os = httpExchange.responseBody  
        os.write(payload.toByteArray())  
        os.close()  
    }  
    ...  
}
```



HTTP Response

<https://git-ce.rwth-aachen.de/dws/simpleweatherserver>

Hoai Viet Nguyen

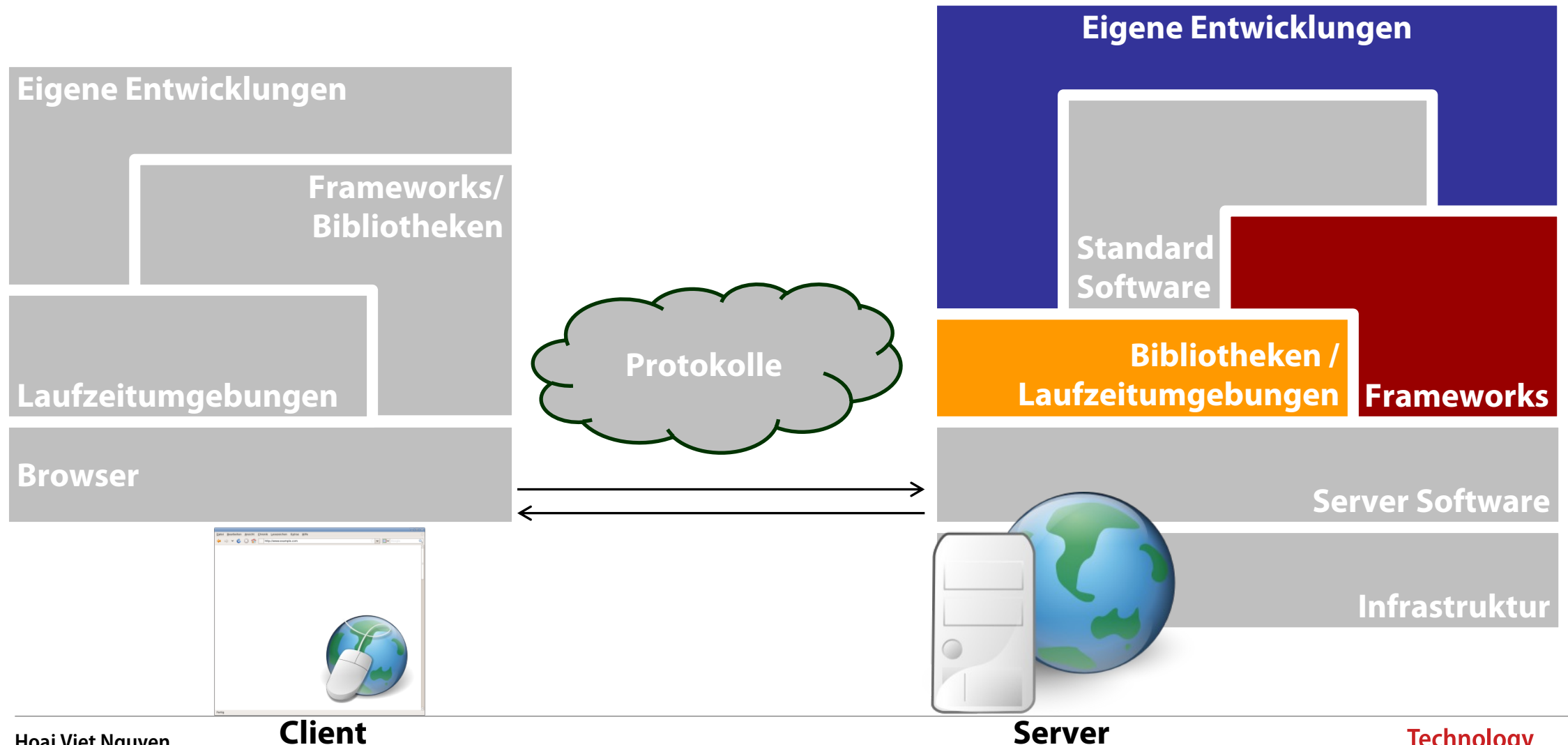
Einführung in die Server-seitige Webprogrammierung mit Spring

Technology
Arts Sciences
TH Köln

Ist es sinnvoll komplexere Webanwendungen auf dieser Basis zu weiterzuentwickeln?



Wo befinden wir uns gerade?



Hoai Viet Nguyen

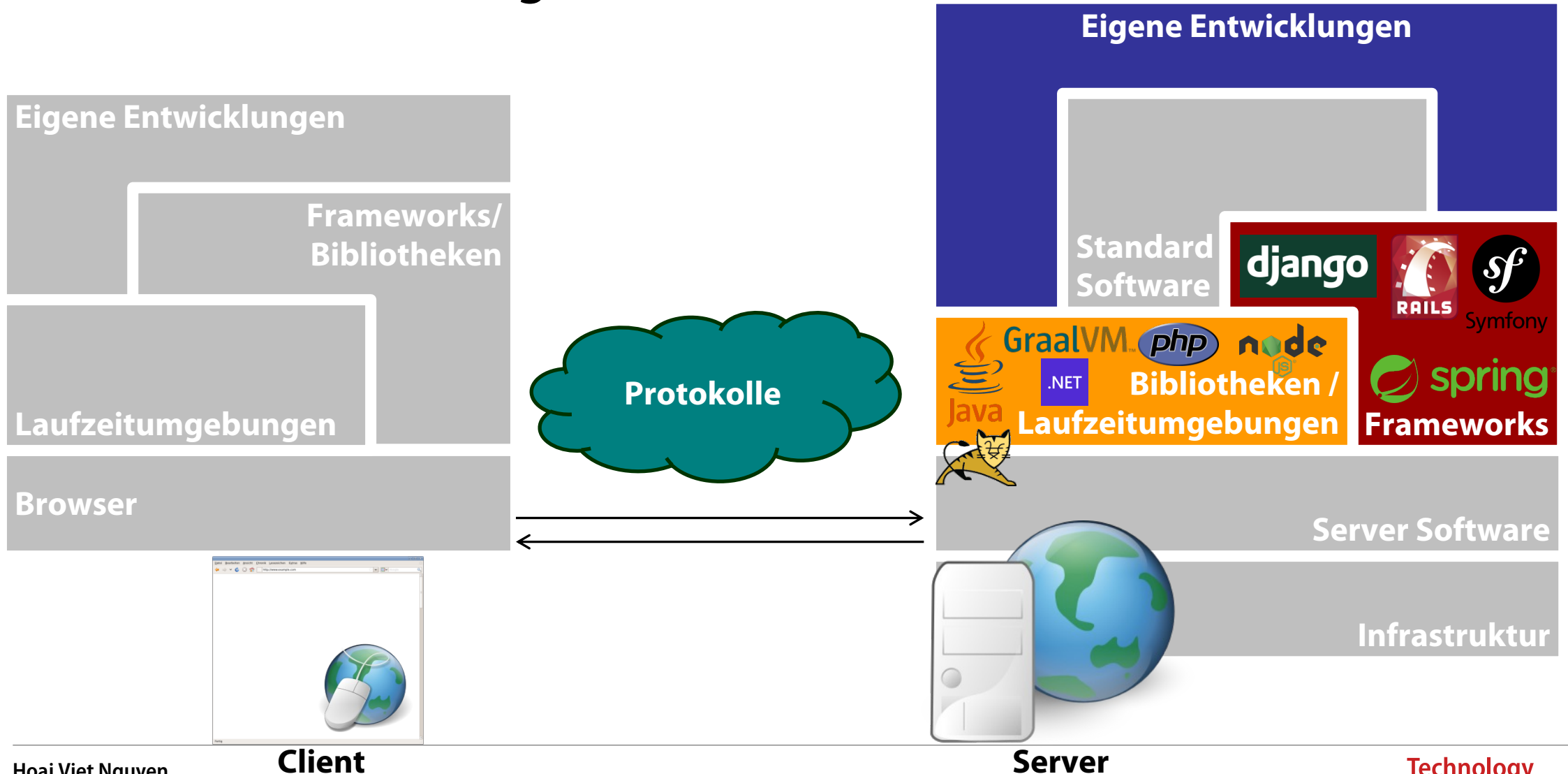
Client

Server

Einführung in die Server-seitige Webprogrammierung mit Spring

Technology
Arts Sciences
TH Köln

Wo befinden wir uns gerade?



Hoai Viet Nguyen

Client

Server

Einführung in die Server-seitige Webprogrammierung mit Spring

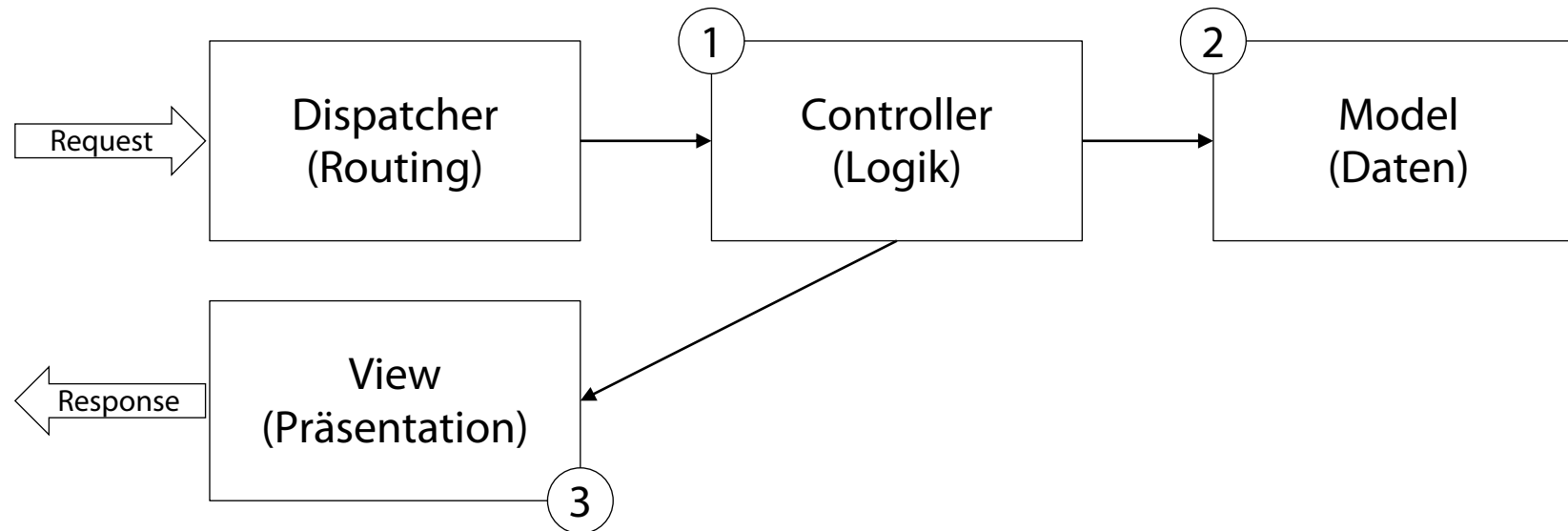
Technology
Arts Sciences
TH Köln

Vorteile für die Verwendung von Webframeworks

- **Rahmengerüst für Entwicklung von komplexen Anwendungen**
- **Vorgefertigte Module und Bibliotheken für effizientere Entwicklung**
- **Best Practices für Skalierbarkeit, Wartbarkeit und Sicherheit**
- **Große Community und Support in öffentlichen Foren**
- **Kostenlos verfügbar und open-source**
- **In Webframeworks steckt viel Expertenwissen**
- **Hilft dabei die Komplexität zu beherrschen und zu strukturieren**

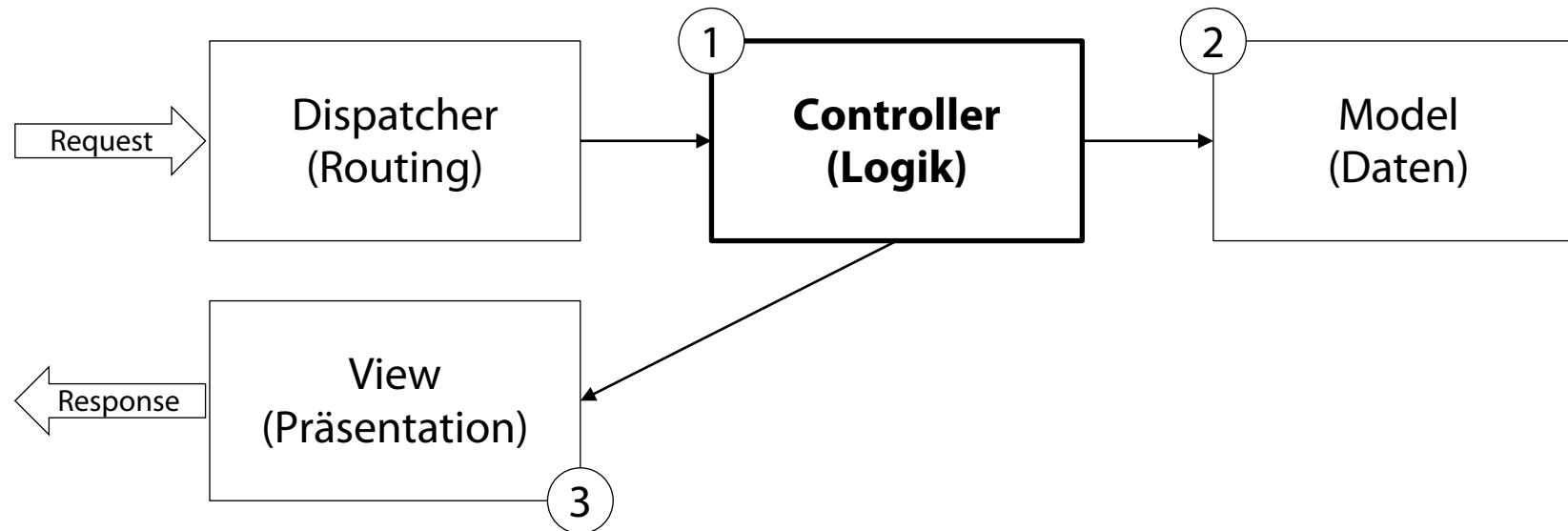
Model-View-Controller Architekturmuster

- Trennung von Logik (Programmierer), Daten (Datenmanager) und Präsentation (Designer)



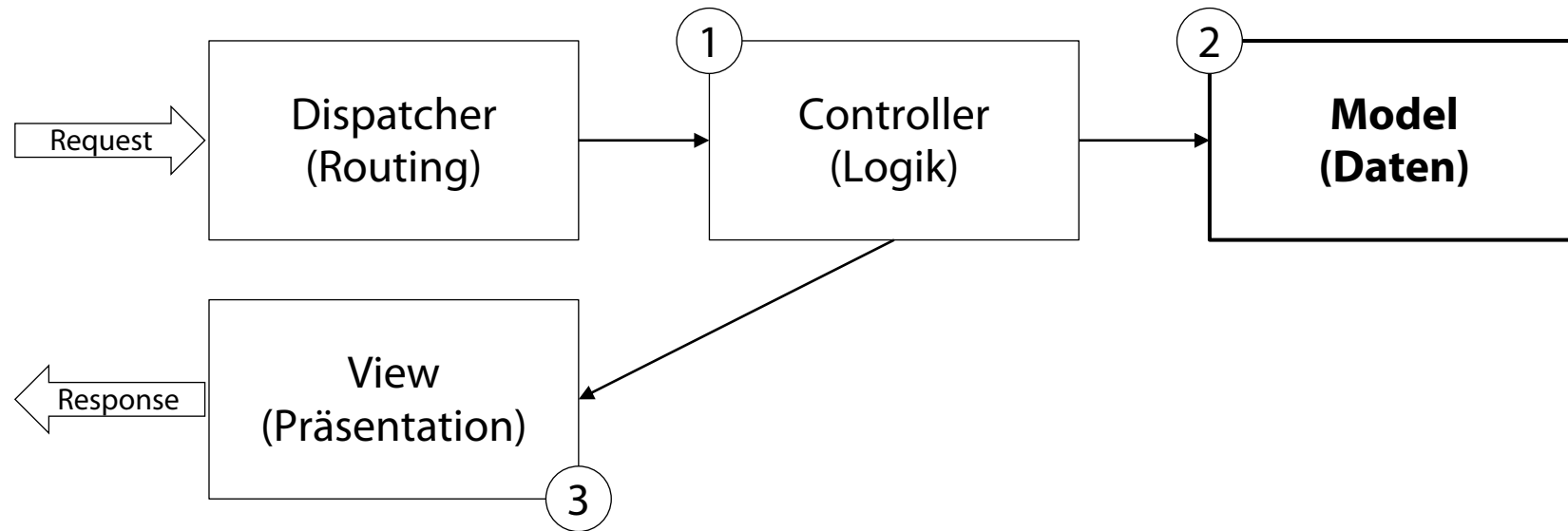
Controller

- Reagiert auf Input (z.B. Anfragen, Nutzereingaben oder Ereignisse) und verarbeitet diese.



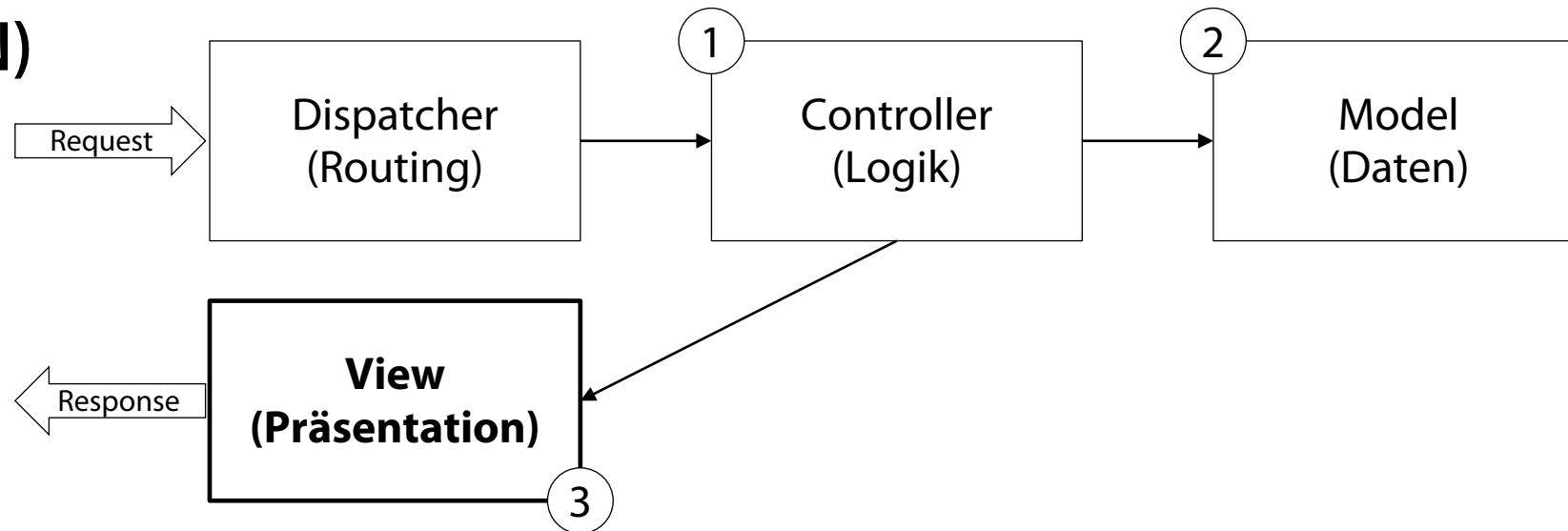
Model

- **Anwendungsspezifische Daten (i.d.R. Datenbank). MVC macht keine Vorgabe, dass es sich um eine Datenbank handeln muss.**



View

- Gibt die Daten in geeignete/angefragter Repräsentation aus. Daten können verschiedene Repräsentationen haben (z.B. HTML, XML oder JSON)





Live-Coding Controller in Spring

Zusammenfassung

- Webframeworks bieten ein Rahmengerüst, welcher dazu beiträgt die Komplexität bei Implementierung zu beherrschen
- MVC ist ein Architekturmuster für die Trennung von Aufgaben
 - Controller -> Logik (Programmierer)
 - Model -> Daten (Datenmanager)
 - View-> Präsentation (Designer)
- In Webframeworks wird ein Handler als Controller bezeichnet

Diese Fragen sollten Sie nach der heutigen Sitzung beantworten können

- Warum sollte man Webframeworks verwenden?
- Welche Server-seitigen Webframeworks gibt es?
- Was ist Model-View-Controller?
- Wie implementiere ich einen Controller in Spring?

Danksagung

Die verwendeten Icons stehen unter der Flaticon Basic Lizenz (flaticon.com) und stammen von

- Freepik
- Atif
- juicy fish
- BZZRINCANTATION