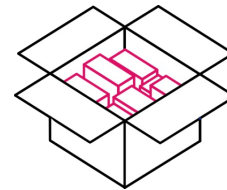


Domain Driven Design in Spring

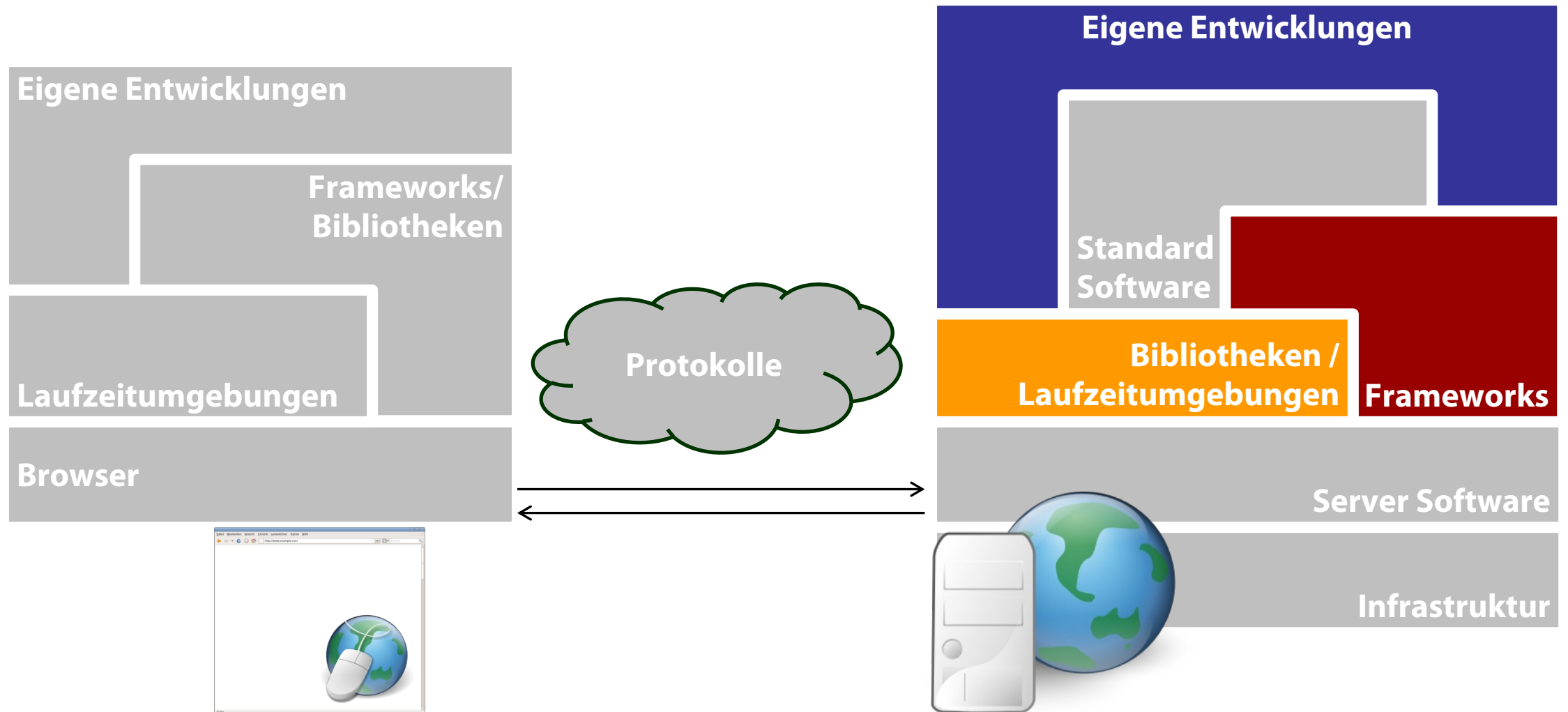
Ein kleiner Einstieg

Hoai Viet Nguyen – TH Köln

Technology
Arts Sciences
TH Köln



Wo befinden wir uns gerade?



Hoai Viet Nguyen

Client

Server

Domain Driven Design in Spring - Ein kleiner Einstieg

Technology
Arts Sciences
TH Köln

Recap letztes Video

- **Model-View-Controller**
 - **Controller:** Verantwortlich für die Logik (Programmierer)
 - **Model:** Verwaltet die Daten (Datenbankmanager)
 - **View:** Stellt die Daten dar (Designer)
- **Implementierung Controller in Spring**
- **Speicherung der Models in einer globalen Variable**

Strukturierung von Daten und Prozessen [Brell2021]

- **Komplexität von Software erfordert die Strukturierung von Daten**
- **Modellierung: Abbildung Wirklichkeit bzw. Abstraktion der Realität**
- **Modellierung von Daten und Prozessen ist elementar für die Entwicklung, Implementierung und Wartbarkeit von komplexen Softwaresystemen**

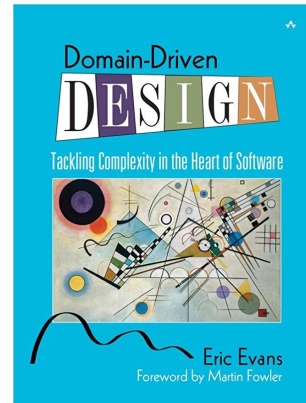
[Brell2021] C. Brell, „Modell und Modellierung – was ist das“, <https://cbrell.de/blog/modell-und-modellierung-was-ist-das/>

Welche Modellierungsansätze kennen Sie bereits?



Domain Driven Design (DDD) [Evans2003]

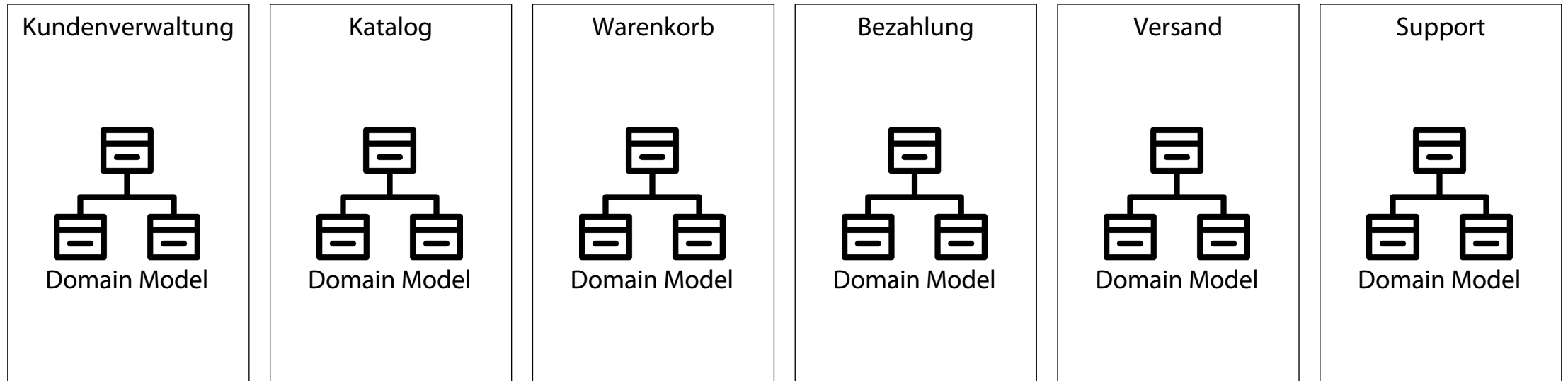
- Modellierung komplexer Softwaresysteme
- Unterteilung von Software in Domänen bzw. Fachlichkeit
- Definition eines Bounded Context für eine Domäne
- Bounded Context definiert gemeinsame Sprache innerhalb Domäne



[Evans2003] E. Evans, „Domain Driven Design“, Pearson International, 2023

DDD anhand eines Beispiels „großer“ Onlineshop

Bounded Contexts



Aufteilung Bounded Context in Building Blocks

- **Entities** (Objekte mit ID, identifizieren sich über ihre ID)
- **Value Objects** (Objekte ohne ID, identifizieren sich über ihren Wert)
- **Aggregates** (Strukturen aus Entities und Value Objects)
- **Services** (Geschäftslogik)
- **Repositories** (Ermöglicht Zugriff auf Entities und Value Objects)
- **Factories** (Ermöglicht das Erstellen eines neuen Aggregates)

Entities und Value Objects in Spring

`@Entity`

`class Task {`

Entspricht dem Model im MVC

`@Id`

`var id: UUID = UUID.randomUUID()`

`var name: String = ""`

`var open: Boolean = true`

`var createdAt: Date = Date()`

Value Objects -> Können einfache
Instanzvariablen aber auch
Objekte/Klassen sein

`override fun toString(): String {`
 `return "Task(id=$id, name='$name')"`
`}`

`}`

Repositories

Repositories in Spring
sind Interfaces

Definition der
Datenbankabfrage
durch Query-Annotation

Alternative Definition der
Datenbankabfrage
durch Camelcase

Beide Varianten liefern äquivalente Rückgabewerte

`@Repository`
`interface` TasksRepository :

`CrudRepository<Task, UUID> {`

`@Query("select t from Task t where t.open = true")`

`fun getAllOpen(): List<Task>`

`@Query("select t from Task t where t.open = true order by t.createdAt asc")`

`fun getAllOpenOrderByCreatedAtAsc(): List<Task>`

`fun findByOpenTrue(): List<Task>`

`fun findByOpenTrueOrderByCreatedAtAsc(): List<Task>`

`}`

Damit der Datenbankabruf funktioniert, müssen Repositories von
einem von Spring vordefinierten Repository Interface erben

Das CrudRepository-Interface beinhaltet
u.a. die Funktionen save, findAll, findById, deleteById

Diese Fragen sollten Sie nach der heutigen Sitzung beantworten können

- Warum sollten Softwaresysteme modelliert werden?
- Was ist Domain Driven Design (DDD)?
- Was sind die Building Blocks Elemente in DDD?
- Wie können Datenbankabrufe in Spring implementiert werden?

Zusammenfassung

- **Modellierung ist eine Methodik, um die Komplexität zu beherrschen**
- **DDD ist ein Ansatz, um komplexe Softwaresysteme zu modellieren**
- **DDD definiert u.a. Building Block Elemente, die in Spring nativ unterstützt werden: Entities, Value Objects, Aggregates, Services, Repositories und Factories**
- **Spring unterstützt zwei Varianten für den Datenbankabrufen**
 - **Query-Annotation: Definition von SQL-Queries**
 - **SQL-Queries in Form von Camelcase**

Danksagung

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

- Freepik
- Atif
- juicy fish
- BZZRINCANTATION