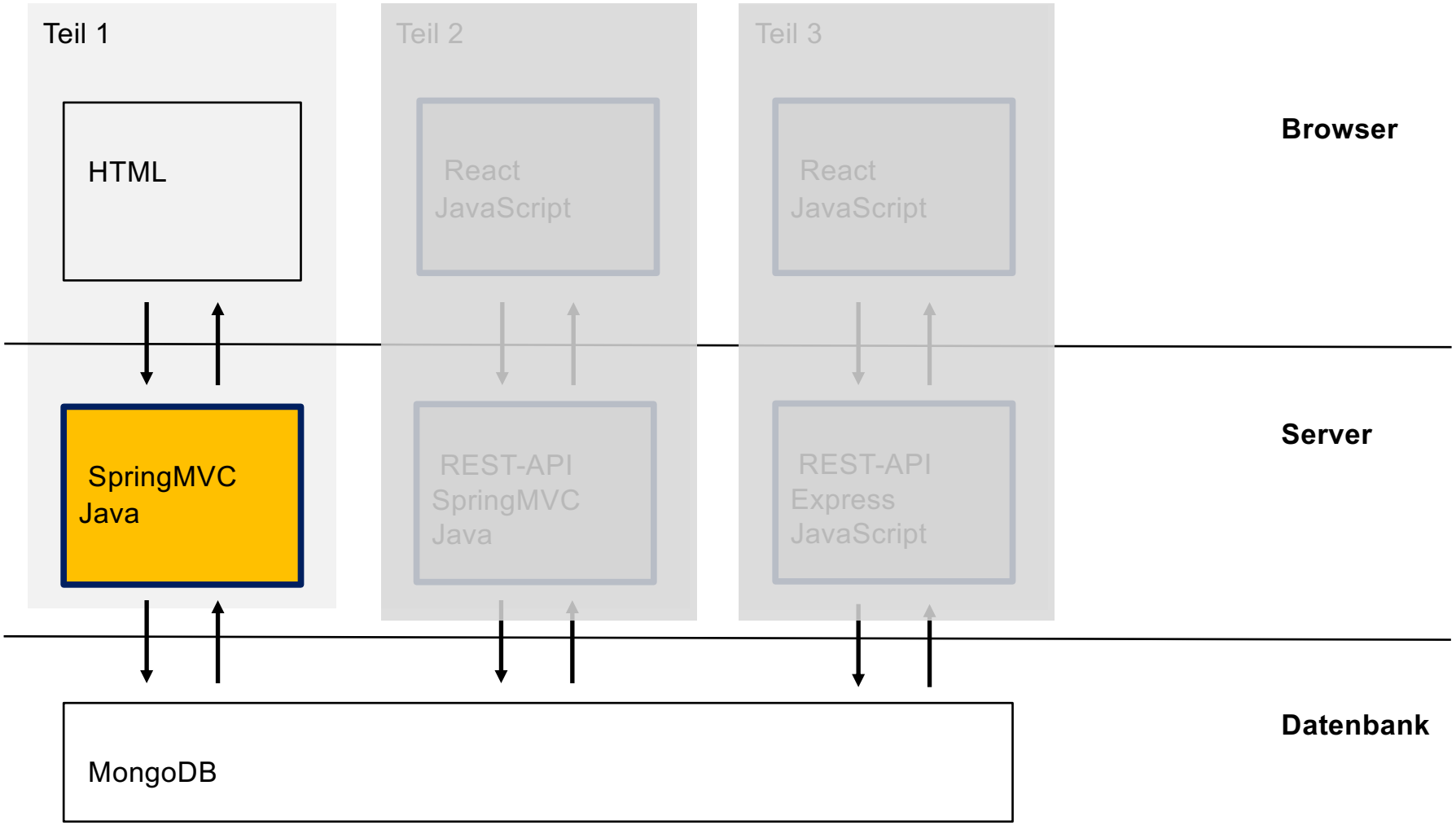
"Controller" Design Pattern

Themen heute

- SpringMVC

- Der Controller
  - FrontController
  - PageController

Lab "flashcard": Setup



Arbeitsblatt 4: Setup

- Setup des SpringMVC Projektes
- Besprechung der Applikation
  - Einsatz von Gradle
  - Funktion des Files "application.properties"
  - Embedded Tomcat als Webserver
    - Bootstrap
  - MongoDB als Datenbank
    - "mongod" als Datenbank-Daemon
    - "mongo" als interaktives Terminal zur Datenbank

Cleaner Code!

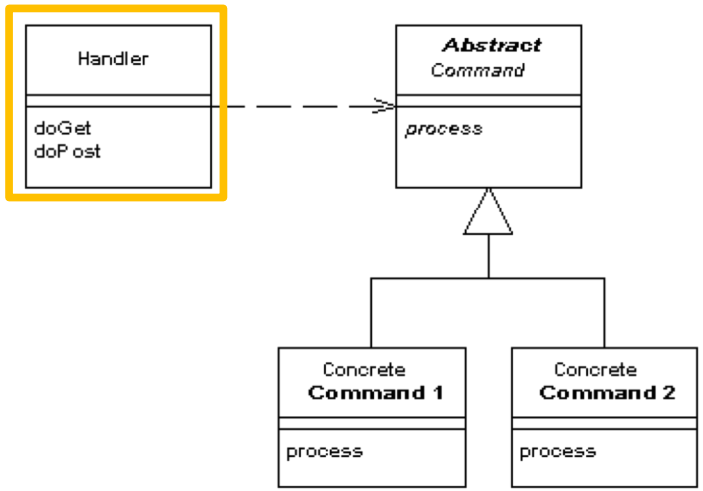
- Wichtige Design-Prinzipien:
  - **Separation of concerns (SoC)**
    "A design principle for separating a computer program into distinct sections, such that each section addresses a separate concern"
    - Wikipedia
  - **Single responsibility principle (SrP)**
    "the single responsibility principle states that every context (class, function, variable, etc.) should have a single responsibility, and that responsibility should be entirely encapsulated by the context."
    - Wikipedia

Was sind Design-Probleme von BasicServlet?

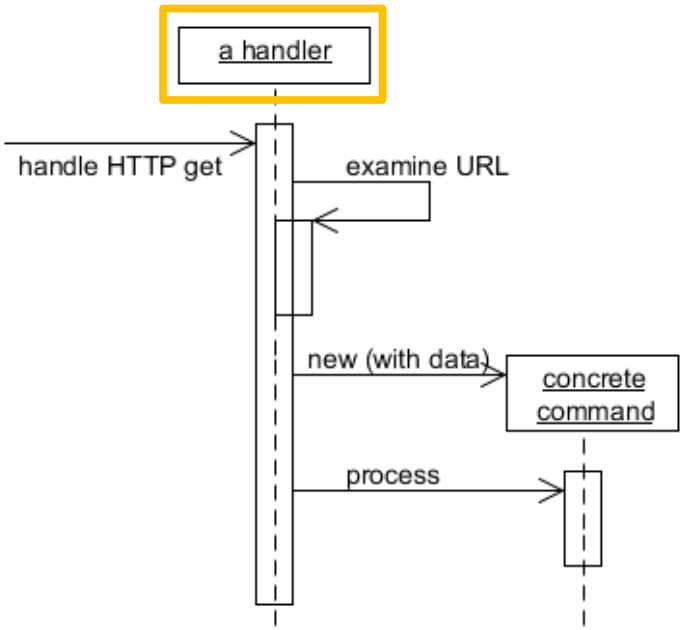
- Was ist gut?
  - Die Persistierung ist in die Klasse QuestionnaireRepository ausgelagert
  - Methode doGet() als klarer und einziger Einstiegspunkt
  - Aufteilung der fachlichen Logik auf Handler-Methoden durch ein Pseudo-Mapping mit "if"-Abfragen
- Was ist verbesserungswürdig?
  - **Dispatching**: Design-Prinzip SoC anwenden
    - URL Handling auslagern
    - Fachliche Logik noch klarer trennen (Klassen!)
  - **Use Case Handling**: Design-Prinzip SrP anwenden
    - Handler einführen
  - **View Handling**: Trennung HTML und Java
    - View-relevanter Code auslagern
    - Unabhängigkeit von der HTML Technologie herstellen

Dispatching → DP "Front Controller"

Klassendiagramm



Sequenzdiagramm



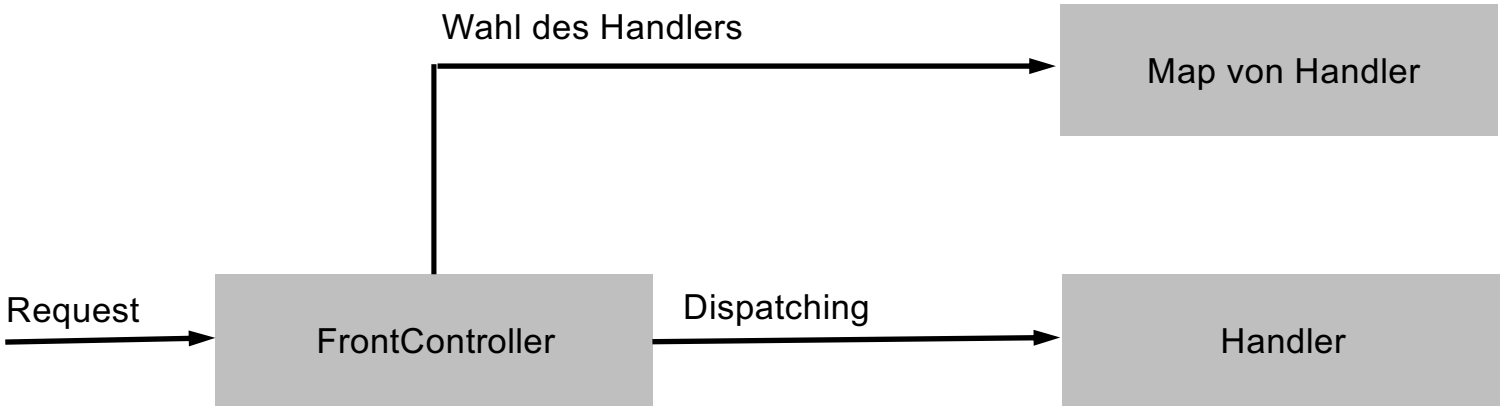
aus Buch "Patterns for Enterprise Application Architecture" von Martin Fowler

"Front Controller" in SpringMVC

- Der "Front Controller" wird über die Klasse "DispatcherServlet" realisiert.
  - Das "DispatcherServlet" wird bei jeder SpringMVC Applikation aufgrund der Annotationen @SpringBootApplication → @EnableAutoConfiguration automatisch initialisiert.
  - Das "DispatcherServlet" wird per Default auf "/" gemappt.
  - Siehe Log-Meldung:
    ```
    "... Mapping servlet: 'dispatcherServlet' to [/]"
    ```

→ Es sind keine weiteren Konfigurationen bzgl. "Front Controller" notwendig.

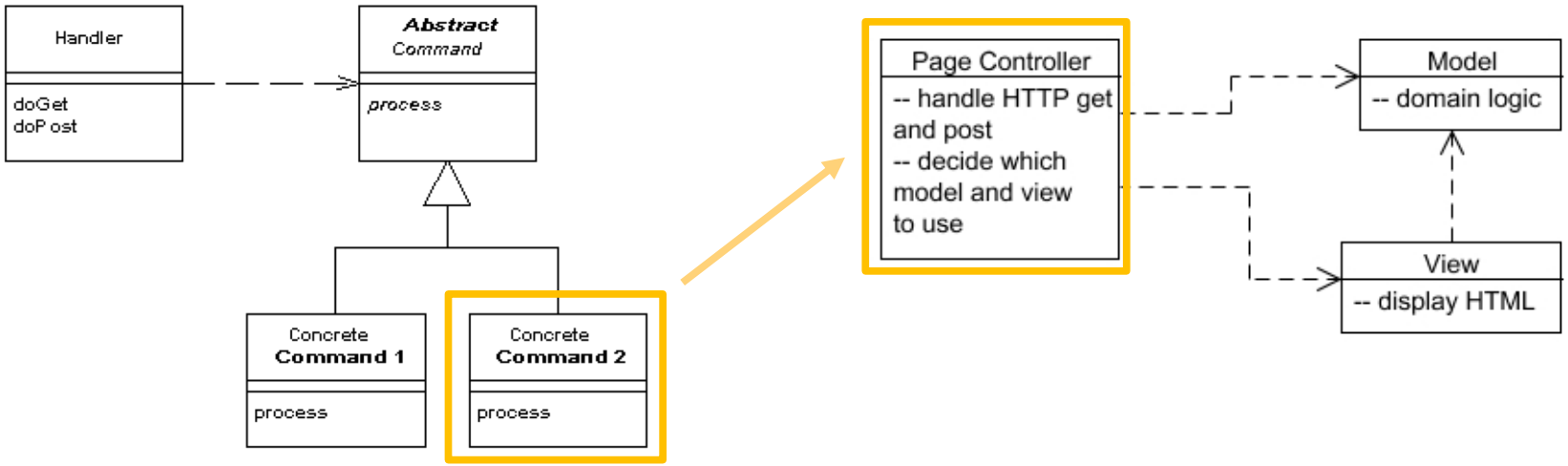
Dispatching im "Front Controller"



- Wie wird die Map der Handler erstellt?
  Annotation auf Klasse/Methode: **@RequestMapping**

Use Case Handler → DP "Page Controller"

Klassendiagramm



"Concrete Command" wird zu "Page Controller"

aus Buch "Patterns for Enterprise Application Architecture" von Martin Fowler

Page Controller in SpringMVC (1/2)

- Wie wird ein POJO zu einem Page Controller?
  - **Annotation `@Controller`**
- Wie kann der Front Controller den korrekten Page Controller auswählen?
  - **Annotation `@RequestMapping` auf der Klasse**
- Wie kann der Front Controller die korrekte Handler-Methode im Page Controller auswählen?
  - **Annotation `@RequestMapping` auf der Methode (oder @GetMapping, @PostMapping, ...)**

Page Controller in SpringMVC (2/2)

```
@Controller
@RequestMapping("/questionnaires")
public class QuestionnaireController {
    @Autowired
    private QuestionnaireRepository questionnaireRepository;

    @RequestMapping(method = RequestMethod.GET)
    public void findAll(HttpServletResponse response,
                        HttpServletRequest request) throws IOException {
        List<Questionnaire> questionnaires = questionnaireRepository.findAll();
        ...
    }
    @GetMapping(value="/{id}")
    public void findAll(@PathVariable Long id, HttpServletResponse response,
                        HttpServletRequest request) throws IOException {
        Questionnaire> questionnaire = questionnaireRepository.findById(id);
        ...
    }
}
```

Arbeitsblatt 5: "BasicServlet" in SpringMVC

- Erster Schritt zu einem SpringMVC Controller

- Basis ist das "BasicServlet" aus Arbeitsblatt 3
  - Klasse "Questionnaire" kopieren
  - Klasse "QuesionnaireRepository" kopieren und ergänzen
  - Klasse "QuestionnaireController" erstellen aus "BasicServlet"

Integration von MongoDB

- Klasse "Questionnaire"
  - Entität, die in der Datenbank persistent gespeichert wird
- Klasse "QuestionnaireRepository"
  - Verbindung zur Datenbank
  - CRUD Methoden auf die Entität "Questionnaire"

Arbeitsblatt 6: Integration MongoDB

- Klasse "Questionnaire" zu Entität umschreiben
  - Entität, die in der Datenbank persistent gespeichert wird
- Klasse "QuestionnaireRepository" als MongoRepository
  - Verbindung zur Datenbank
  - CRUD Methoden auf der Entität "Questionnaire"

Übung 3 : HelloWorldController

- Hausaufgabe
- Neuer "HelloWorldController" als PageController
  - Request
    `http://localhost:8080/flashcard-mvc/hello?name=Hugo`
  - Response als Text
    ```
    Hello Hugo
    You have 3 Questionnaires in your repo.
    ```