Databanken en webtoepassingen in Java

K. Coolsaet

Universiteit Gent

16 maart 2018

WiFi — UGentGuest
Gebruikersnaam: `guestM3java`
Wachtwoord: `ag4oZnpQ`
Eerst surfen naar http://www.ugent.be

Cursuswebsite: `http://inigem.ugent.be/moevie.html`

Kennismaken

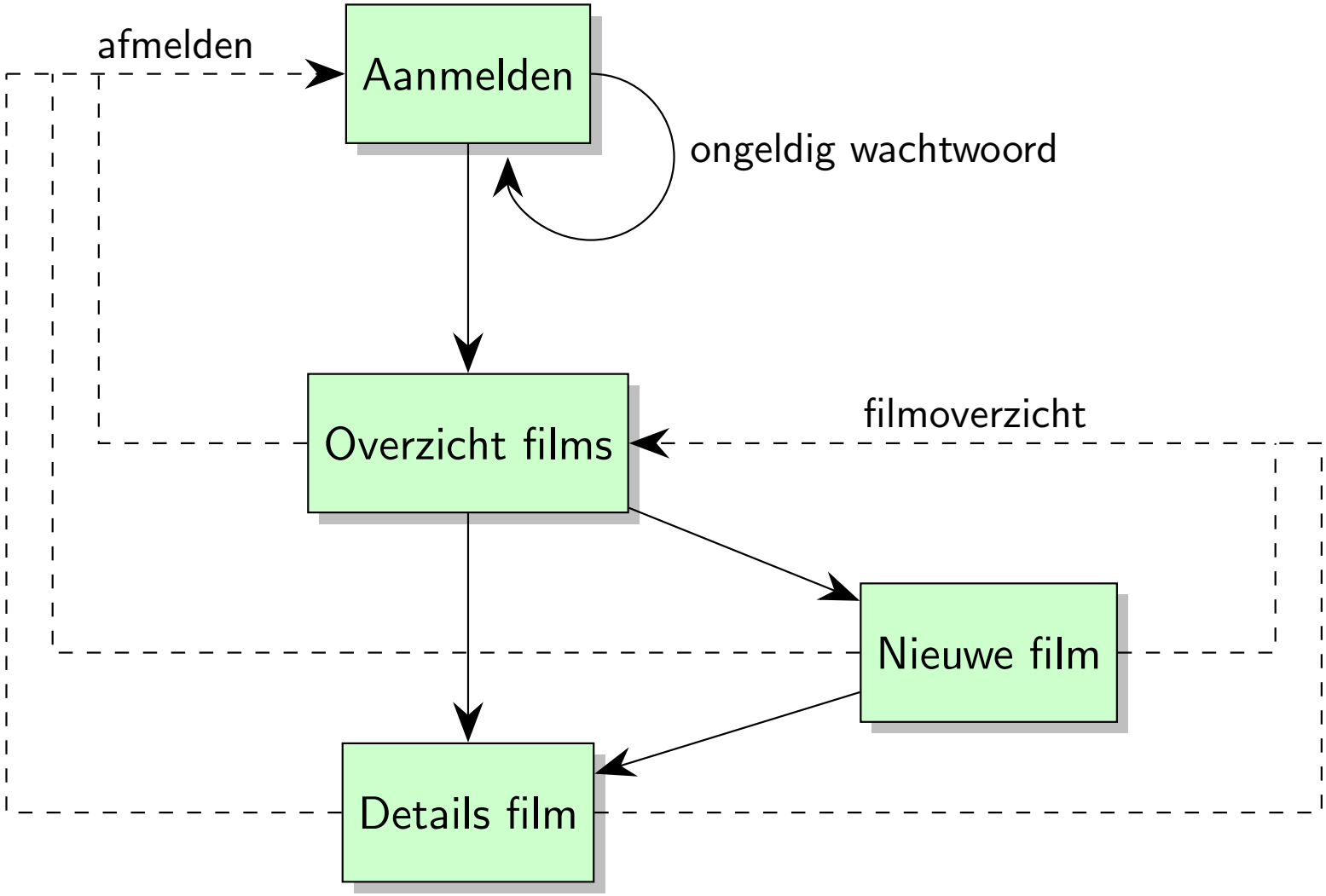
- **Wie ben ik?**
- **Wie zijn jullie?**
- (Actieve) kennis van
  - Java
  - SQL
  - HTML
  - CSS
  - Webtoepassingen (welke taal/framework?)
- Software goed geïnstalleerd?
- IntelliJ IDEA?
- Windows 7/8/10, macOS, Linux?

Voorwoord

- Waarom deze cursus?
- Keuze: databanken (JDBC vs. JPA)
- Keuze databankserver: Derby = Java DB
- Keuze: webtoepassingsraamwerk
  - Steile leercurve
  - (Eerste versie van Moevie geschreven in **Play**)
  - **Spark**, lichtgewicht framework
  - Template Language **FTL** (FreeMarker)
  - Kleine bijkomende bibliotheek **spark-xtra**

Moevie

- Mappen in het *Moevie*-project
- Moevie-webtoepassing bestaat uit 4 pagina's (Demo)



Moevie databank

Gebruikers
Films
Beoordelingen

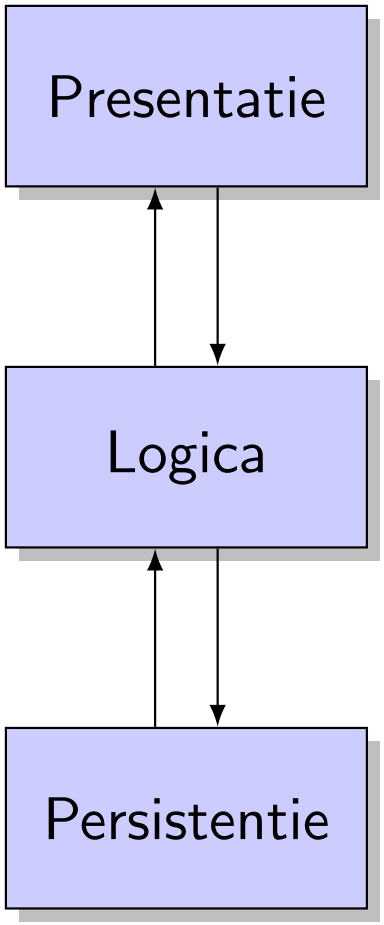
ID	Wachtwoord
albert	`e=mc^2`
aladdin	`sesamopenu`
david	`bowie`

ID	Titel	Hoofdrol	Eigenaar	Aangemaakt
1	Star Wars	Mark Hamil	albert	2017-02-18 14:32
2	Terminator	Arnold Schwarzenegger	albert	2017-02-18 14:32
3	Indiana Jones	Harrison Ford	aladdin	2017-02-18 14:32
4	Casablanca	Humphrey Bogart	albert	2017-02-19 13:18
5	The Lego Movie	Batman	aladdin	2017-02-19 13:27

Film	Gebruiker	Commentaar	Sterren	Aangemaakt
1	albert	Dat vind ik een leuke . . .	4	2017-02-18 14:32
1	aladdin	Nogal ouderwets maar OK	3	2017-02-18 14:32
2	albert	Enge film!	3	2017-02-18 14:32
3	albert	Geen	3	2017-02-19 13:17
4	albert	NULL	NULL	2017-02-19 13:18
3	aladdin	NULL	NULL	2017-02-19 13:18
4	aladdin	Trage film, in zwart en wit	3	2017-02-19 13:18
2	aladdin	NULL	NULL	2017-02-19 13:27

Het meerlagenmodel

Algemeen



Wat?

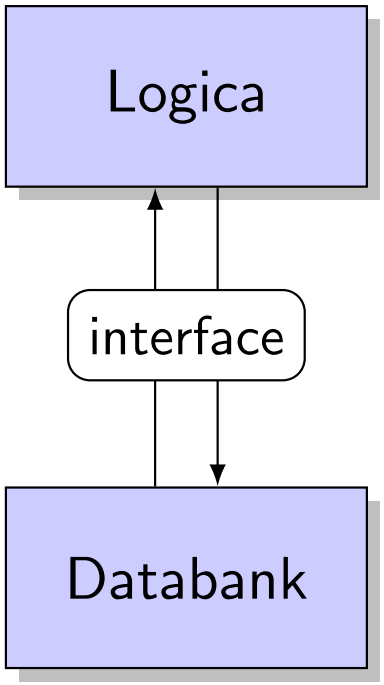
- Elke laag communiceert slechts met de 'naburige' lagen

Waarom?

- Onafhankelijk van elkaar ontwikkeld
- Implementatie gemakkelijk vervangen door een andere
- Verschillende versies in verschillende omstandigheden

Meerlagenmodel

Moevie

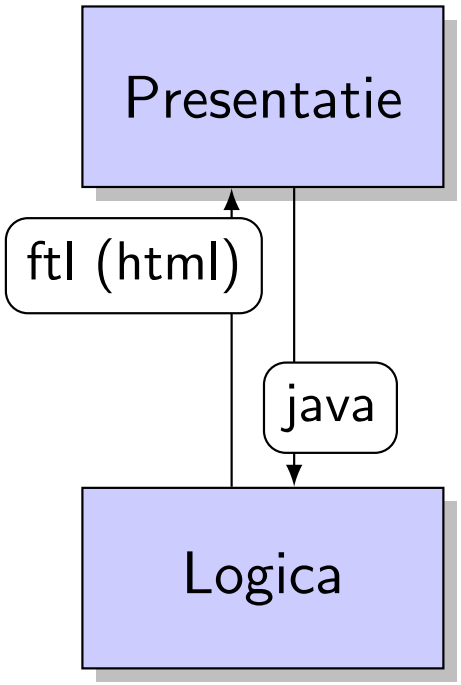


Persistentielaag

- Afzonderlijk IDEA-project
- Getest in ander IDEA-project
- Interface is Java-interface
- Implementatie is verborgen

Meerlagenmodel

Moevie



Presentatielaag

- Zelfde IDEA-project als logicalaag
- Java vs. HTML (FTL)
- Afzonderlijke mappen in project (*src* / *resources*)

Toch onafhankelijk van elkaar!

Opdracht

Voer Moevie uit met een andere presentatielaag (blz. 26)

(Tussendoor . . .)

Bierkiezer

Fase 1: Functionele analyse

- Welke schermen voorzie je?
- Maak alvast (eenvoudige!) HTML-pagina's
- Bedenk interessante uitbreidingen

Persistentielaag

Elementaire gegevens

```
public class Film {
    private int id;
    private String titel;
    private String eigenaar;
    private String hoofdrol;

    public Film(int id, String titel, String hoofdrol, String eigenaar){
        this.id = id;
        this.titel = titel;
        this.hoofdrol = hoofdrol;
        this.eigenaar = eigenaar;
    }

    // + getters (en setters)
}
```

Belangrijk! Elementaire structuur, *geen* verwijzingen naar andere objecten.

Persistentielaag

Data-access-objecten

- Beoordelingen kan je *niet* aan het filmobject zelf vragen.
- Wel aan een **data access object** (DAO)

List<Beoordeling> lijst = dao.lijstBeoordelingen(filmId);

- Moevie heeft 3 DAO's: gebruikers-, film- en beoordelingDAO.
- Elke DAO is een *interface*

```
public interface FilmDAO {
    Film getFilm(int film);
    List<Film> lijstFilms();
    int nieuweFilm(String titel, String eigenaar, String hoofdrol);
    void verwijderFilm(int film);
}
```

Persistentielaag

Data-access-provider

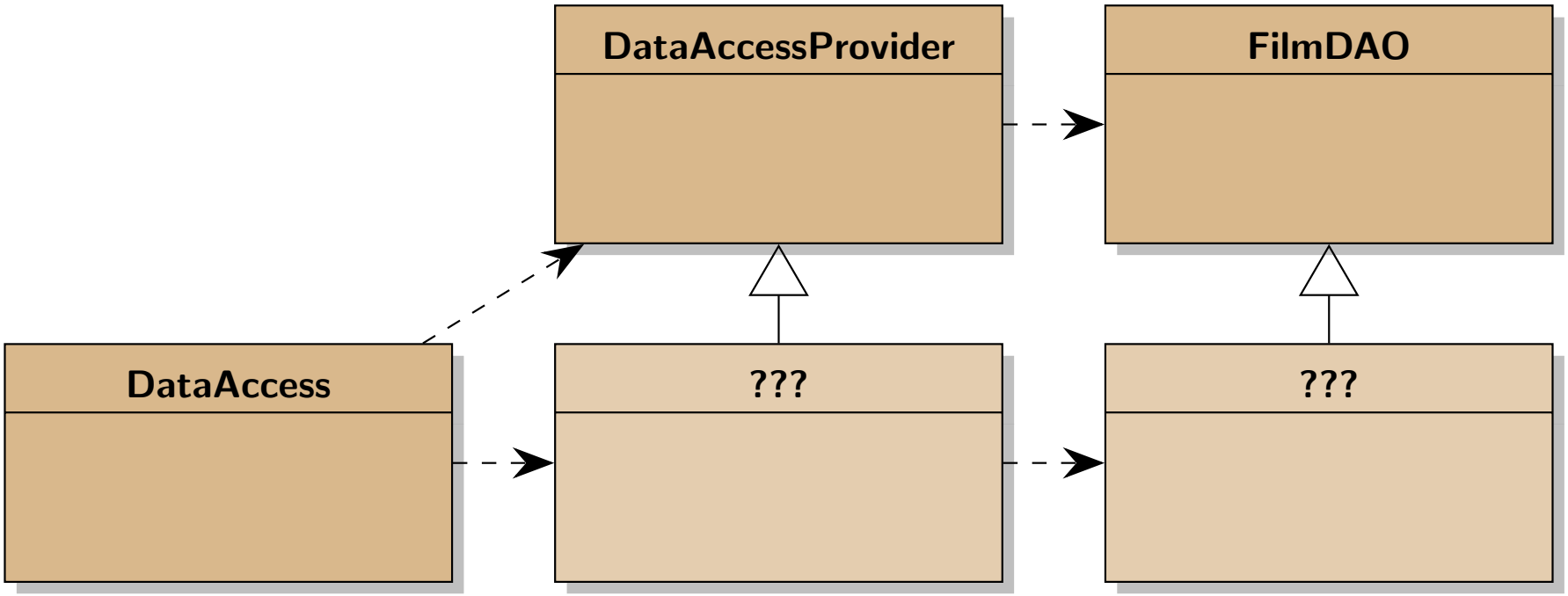
- Hoe kom je aan een DAO?
- '*dao* = **new** *FilmDAO*();' werkt niet (waarom?)
- Gebruik een data-access-**provider**

```
public interface DataAccessProvider {
    GebruikerDAO getGebruikerDAO();
    FilmDAO getFilmDAO();
    BeoordelingDAO getBeoordelingDAO();
}
```

- Opnieuw interface. . .

```
DataAccessProvider dap = DataAccess.getDAP ("moevie");
FilmDAO dao = dap.getFilmDAO();
...
```

Persistentielaag



Persistentielaag

Opdracht

Schrijf een programma dat alle films afdrukt die `albert` heeft gezien.

- Bekijk de broncode van *FilmDAO* in het *db*-project
- Maak een nieuw IDEA-project *oefening* aan naast *db*
- Voeg de bibliotheken *db.jar* en *derby.jar* toe aan *oefening*
- Schrijf je programma en voer het uit met als werkmap *etc\db*

(Tussendoor . . .)

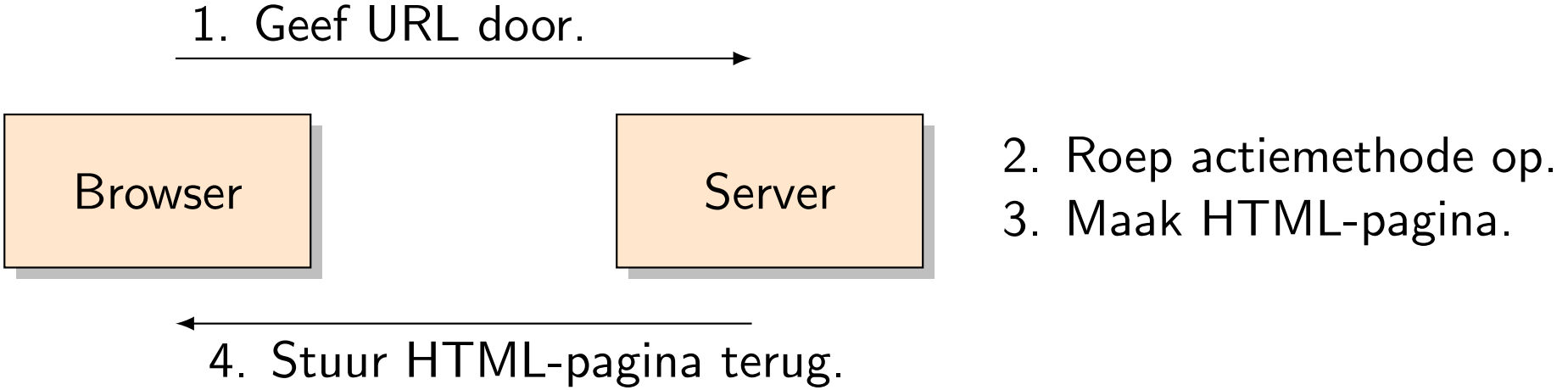
Bierkiezer

Fase 2: Ontwerp van de persistentielaag

- Hoe zien de DAO-interfaces eruit
- Schrijf zelf (eenvoudige!) DAO-implementaties met hardgecodeerde informatie over Belgische bieren
- Vertrek van de minimale webtoepassing (cf. cursussite)
- **Belangrijk** Alle (publieke) klassen moeten in een package geplaatst worden (bijv. *bierkiezer.db* of *bier.db*)

Spark

Wat gebeurt er wanneer we `http://localhost:4567/films` opvragen?



Actiemethode behoort tot een **controller-klasse**.

Actiemethode

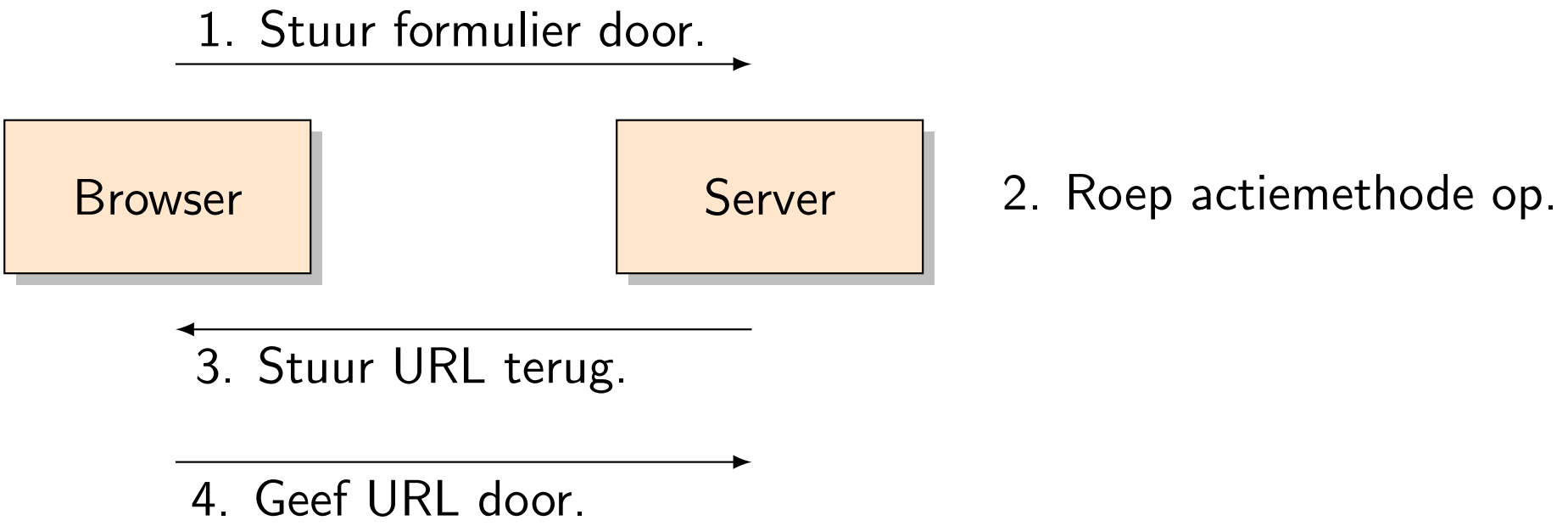
Welke methode?

- Wordt bepaald door **pad** en **route** (zie broncode).
- Onderscheid tussen GET en POST

Actiemethode

- Geen parameters
- Retourtype *Page* (zie code).

Formulier



Redirect naar pad.

Sjablonen — templates

Webpagina's worden gemaakt gebaseerd op **sjablonen**

```
public Page index() {
    Page p = page ("films.html");
    ...
    return p;
}
```

Sjablonen lijken op HTML ...

```
<h1>Overzicht films</h1>
<table>
    <thead>
    <tr>
        <th>Film</th>
        <th>Hoofdrol</th>
        <th>Rating</th>
        <th>Eigenaar</th>
    </tr>
    </thead>
```

Sjablonen

```
...
<tbody>
<#list films as f>
    <tr>
        <td><a href="/films/${f.id}">${f.titel}</a></td>
        <td>${f.hoofdrol}</td>
        <td>${f.sterren}</td>
        <td>${f.eigenaar}</td>
    </tr>
</#list>
</tbody>
</table>
```

. . . Maar zijn geschreven in FTL (*FreeMarker Template Language*).

- Controlestructuren
- Uitdrukkingen / variabelen
- (Macro's)

Sjablonen

with-constructie

Hoe gegevens doorgeven van actiemethode naar sjabloon?

```
public Page index() {
    List<Film> mijnFilms = DataAccess.getDAP()
                                     .getFilmDAO().lijstFilms();
    Page p = page("films.html").with ("films", mijnFilms);
    return p;
}
```

Withs kunnen in 'kettingen' voorkomen.

```
return page("login.html")
            .with("gebruikersnaam", gebruikersnaam)
            .with("wachtwoord", wachtwoord)
            .with("error", error);
```

URL-parameters

Route van een URL kan ook ‘parameters’ bevatten:

`http://localhost:4567/films/3`

(Toon details over film met ID=3)

Routedefinitie:

```
private void routes() {
    ...
    get("/films/:id", Filmdetail.class, "index");
}
```

In actiemethode:

```
public Page index() {
    String idString = request.params(":id");
    ...
    return page ("film.html").with(...);
}
```

Query-parameters

Alternatief: 'query parameter' in de *query string*

```
http://localhost:4567/films/find?id=3
```

Actiemethode:

```
public Page find() {
    String idString = request.queryParams("id");
    ...
    return page ("film.html").with(...);
}
```

Inhoud van **formulieren** wordt ook doorgestuurd in *query parameters*.

Route

```
private void routes() {
    ...
    post("/session", Start.class, "login");
    ...
}
```

Query-parameters

Formulier

```
<form action="/session" method="POST">
    <label>Login-ID</label>
    <input type="text" name="gebruikersnaam" value="">

    <label>Wachtwoord</label>
    <input type="password" name="wachtwoord" value="">

    <button type="submit">Aanmelden</button>
</form>
```

Actiemethode:

```
public Page login() {
    String naam = request.queryParams("gebruikersnaam");
    String wwoord = request.queryParams("wachtwoord");
    // doe iets met naam en wwoord
}
```

(Om de dag af te sluiten . . .)

Bierkiezer

Fase 3: Eerste werkende versie

Maak een webtoepassing `bierkiezer` die op zijn minst een eerste homepagina toont. Werk dan de basisfunctionaliteit uit

- **Vooraf** Maak een bibliotheek van de persistentielaag (demo)
- Vertrek van de minimale webtoepassing
- Voeg de persistentielaag toe als externe bibliotheek
- Plaats je actiemethode in de reeds bestaande controller *Start*
- Verander de naam van de package in *bierkiezer* of *bier*.
- Plaats sjablonen in de *views*-map

- Eerst het bierkiezen zelf
- Daarna het beheer (je hoeft nog niet 'vriendelijk' te reageren op fouten.)