



**ftn**informatika

# *Java Web Development*

Modul 2

Termin 4

# Sadržaj

1. Nedostaci dosadašnjeg pristupa u generisanju dinamičkog HTML sadržaja
2. *Template engine*
3. *Thymeleaf*
  - I. standardni dijalekat
  - II. izrazi
  - III. način popunjavanja šablona
  - IV. podešavanje
  - V. izvor podataka
  - VI. tok kontrole
4. GET
  - I. Bez parametara
  - II. Sa parametrima u URL-u
5. zadatak za vežbu
6. domaći

# Nedostaci dosadašnjeg pristupa u generisanju dinamičkog HTML sadržaja

- dizajn stranica (HTML) i programska obrada (*Java*) su pomešani u istim datotekama
- teško je razdvojiti funkcije dizajnera i programera
- jako je nepraktično rukovati HTML kodom unutar *String* literala u *Java* klasama
- svaka promena u izgledu stranice zahteva kompajliranje *controller*-a i ponovno pokretanje aplikacije (*Eclipse* doduše ovo obavlja automatski i brzo)

# Template engine

## Ideja

- zaminiti **mnogo HTML-a** sadržanog u **malo programskog koda** sa...

```
out.append(
    "<div class=\"row\">\r\n" +
    "    <div class=\"col\">\r\n" +
    "        <table class=\"table table-striped\">\r\n" +
    "            <thead>\r\n" +
    "                <tr>\r\n" +
    "                    <th>Redni broj</th>\r\n" +
    "                    <th>Naziv</th>\r\n" +
    "                    <th></th>\r\n" +
    "                </tr>\r\n" +
    "            </thead>\r\n" +
    "            <tbody class=\"table-group-divider\">\r\n" +
    );
for (int it = 0; it < zanrovi.size(); it++) {
    out.append(
        "<tr>\r\n" +
        "    <td class=\"text-end\"> " + (it + 1) + "</td>\r\n" +
        "    <td><a href=\"/zanrovi/prikaz?id=" + zanrovi.get(it).getId() + "\"> " + zanrovi.get(it).getNaziv() + "</a></td>\r\n" +
        "    <td><a href=\"/filmovi?zavrId=" + zanrovi.get(it).getId() + "\">filmovi</a></td>\r\n" +
        "</tr>\r\n" +
    );
}
out.append(
    "    </tbody>\r\n" +
    "    </table>\r\n" +
    "    </div>\r\n" +
    "</div>"
);
```

# Template engine

## Ideja

... malo programskog koda sadržanog u mnogo HTML-a!

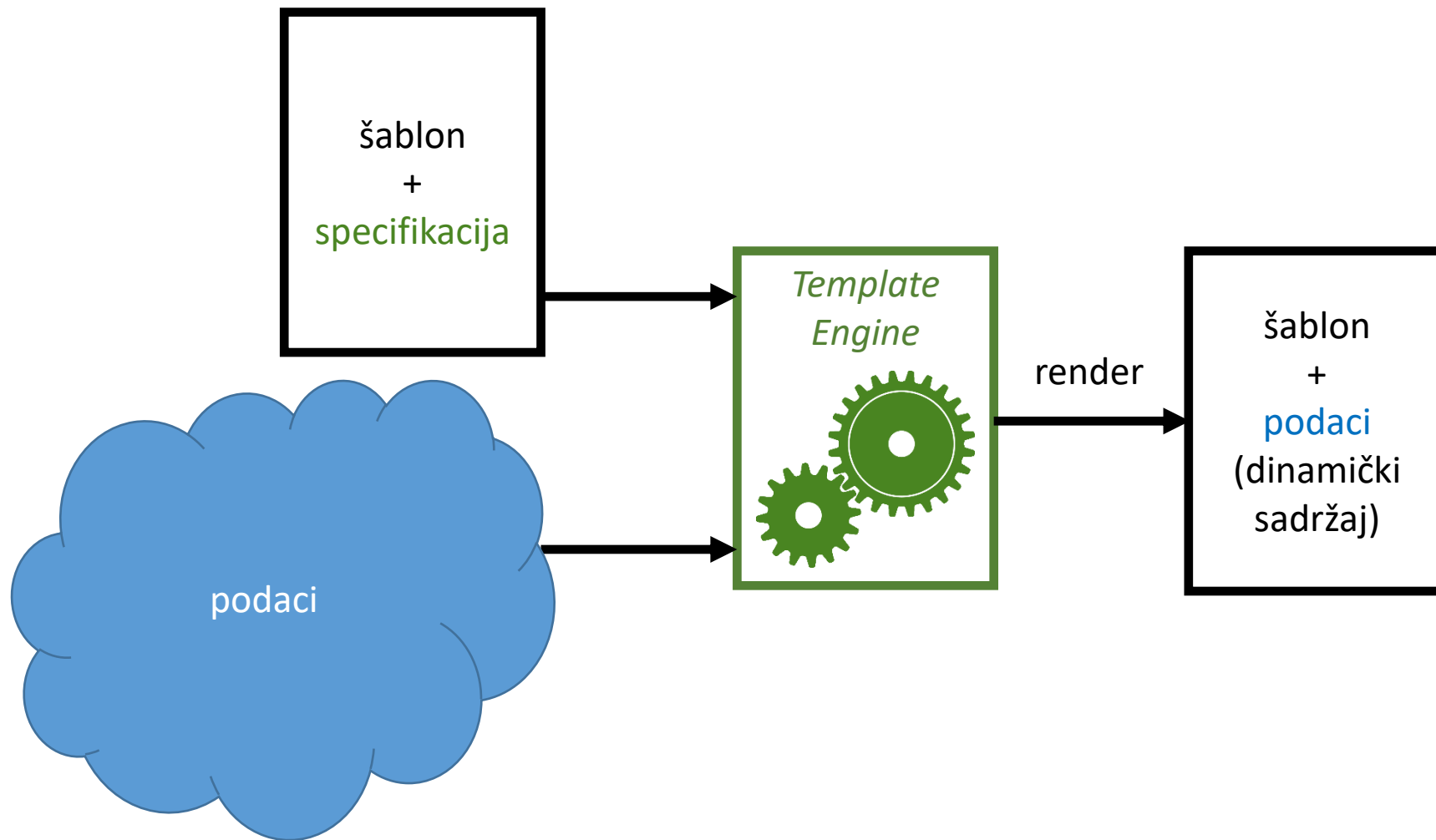
```
<div class="row">
  <div class="col">
    <table class="table table-striped">
      <thead>
        <tr>
          <th>Redni broj</th>
          <th>Naziv</th>
          <th></th>
        </tr>
      </thead>
      <tbody class="table-group-divider">
        <tr th:each="itZanr, status: ${zanrovi}">
          <td class="text-end" th:text="${status.count}">1</td>
          <td><a th:href="/zanrovi/prikaz?id=${itZanr.id}|" th:text="${itZanr.naziv}">žanr 1</a></td>
          <td><a th:href="/filmovi?zanrId=${itZanr.id}|">filmovi</a></td>
        </tr>
      </tbody>
    </table>
  </div>
</div>
```

# Template engine

- softverska komponenta koja ima za zadatak da **automatski** u proizvoljan tekstualni šablon, napisan određenom sintaksom, **ubaci** odgovarajuće **podatke** na posebno unapred obeležena mesta i na specificirani način
- *template engine* može biti deo *framework*-a ili može biti *3rd-party* biblioteka
- programer piše statičke **tekstualne šablone** u pogodnom *editor*-u za odabranu sintaksu, a zatim šablone **dopunjuje specifikacijom** na osnovu koje će *template engine* za vreme izvršavanja ubaciti odgovarajuće podatke
- način pisanja specifikacije zavisi od samog *template engine*-a
- *template engine*-u je pored tekstualnog šablona sa specifikacijom potreban i **izvor podataka** da bi proizveo konačan rezultat
- ako je šablon statički HTML dokument, tada iz njega nastaje **dinamički HTML** sadržaj

# Template engine

## Ideja



# Thymeleaf



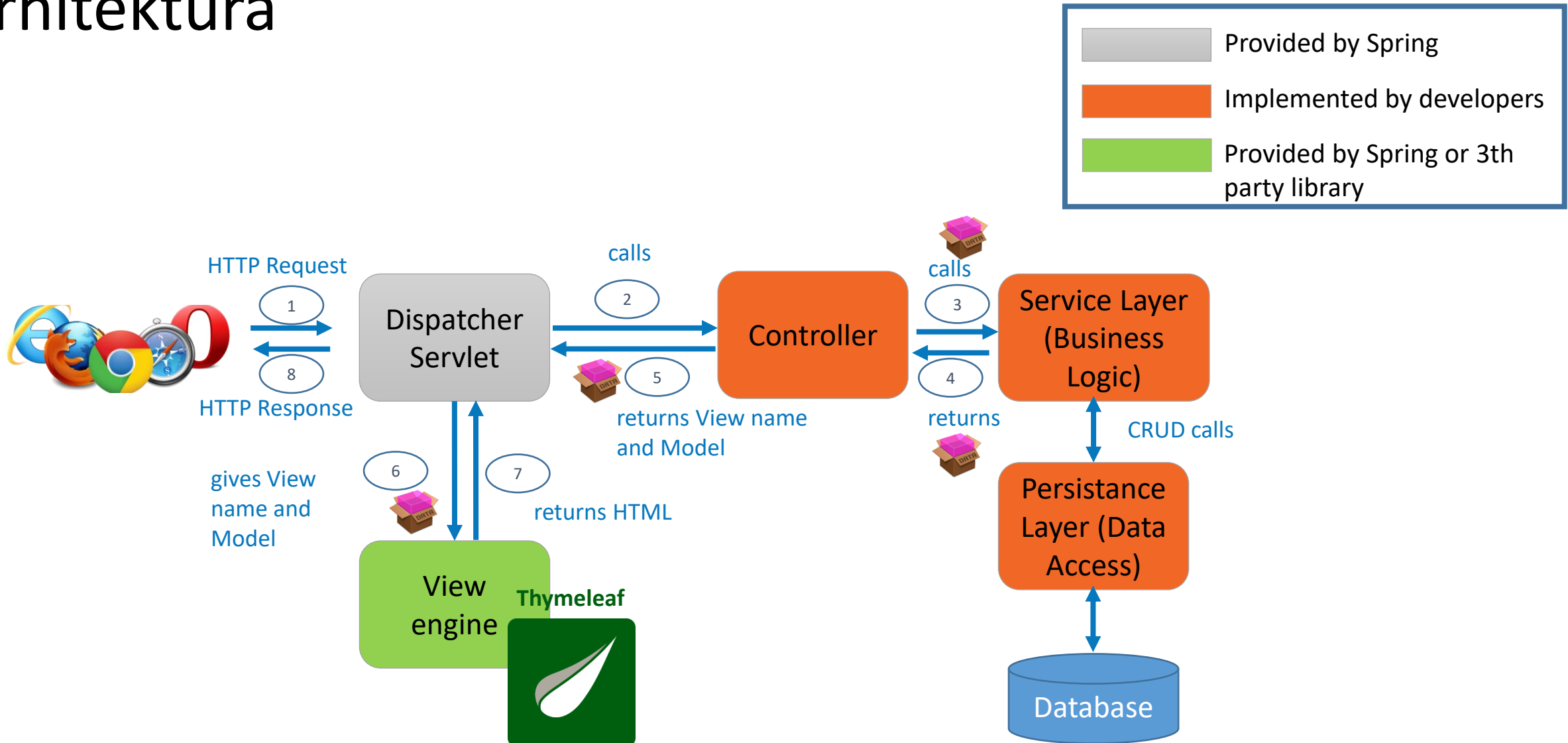
*thyme* = majčina dušica

- moderan *open source template engine* baziran na *Java* programskom jeziku
- može se koristiti u sklopu web aplikacija ili samostalno za druge namene
- podržava XML, XHTML i *HTML5* sintakse
- dobro integrisan u *Spring Framework*, pa se u sklopu njega vrlo jednostavno koristi za generisanje dinamičkog HTML sadržaja od strane servera



# Thymeleaf

## Arhitektura



# Thymeleaf

U kontekstu HTML dokumenata, **šablon je moguće popuniti na sledeće načine:**

- popunjavanjem tekstualnog sadržaja elemenata
- popunjavanjem vrednosti atributa
- uslovnim prikazom elemenata i atributa
- ponavljanjem elemenata ili grupe elemenata
- način pisanja specifikacije na osnovu koje će *Thymeleaf* popuniti HTML šablon se naziva **dijalekat**
- *Thymeleaf* podržava više dijalekata
- **Standardni dijalekat** predviđa **posebne HTML attribute** za specifikaciju načina popunjavanja šablona
- **atribut specificira šta** *Thymeleaf* treba da obavi za element u okviru kog je naveden, a **vrednost atributa specificira kako** će to da se obavi

## Standardni dijalekat

- oslanja se na direktive
- oblik: `<element th:atribut="izraz"></element>`

```
<div class="row">
  <div class="col">
    <table class="table table-striped">
      <thead>
        <tr>
          <th>Redni broj</th>
          <th>Naziv</th>
          <th></th>
        </tr>
      </thead>
      <tbody class="table-group-divider">
        <tr th:each="itZanr, status: ${zanrovi}">
          <td class="text-end" th:text="${status.count}">1</td>
          <td><a th:href="/zanrovi/prikaz?id=${itZanr.id}|" th:text="${itZanr.naziv}">žanr 1</a></td>
          <td><a th:href="/filmovi?zanrId=${itZanr.id}|">filmovi</a></td>
        </tr>
      </tbody>
    </table>
  </div>
</div>
```

## Izrazi

- vrednost atributa je tipično određena **izrazima**
- izračunavanje izraza rezultuje nekom **vrednošću** određenog tipa (tekstualnog, numeričkog, *boolean*, *null* i sl., ili može biti i referenca na objekat)

Postoji **više vrsta izraza** od kojih ćemo se ograničiti na:

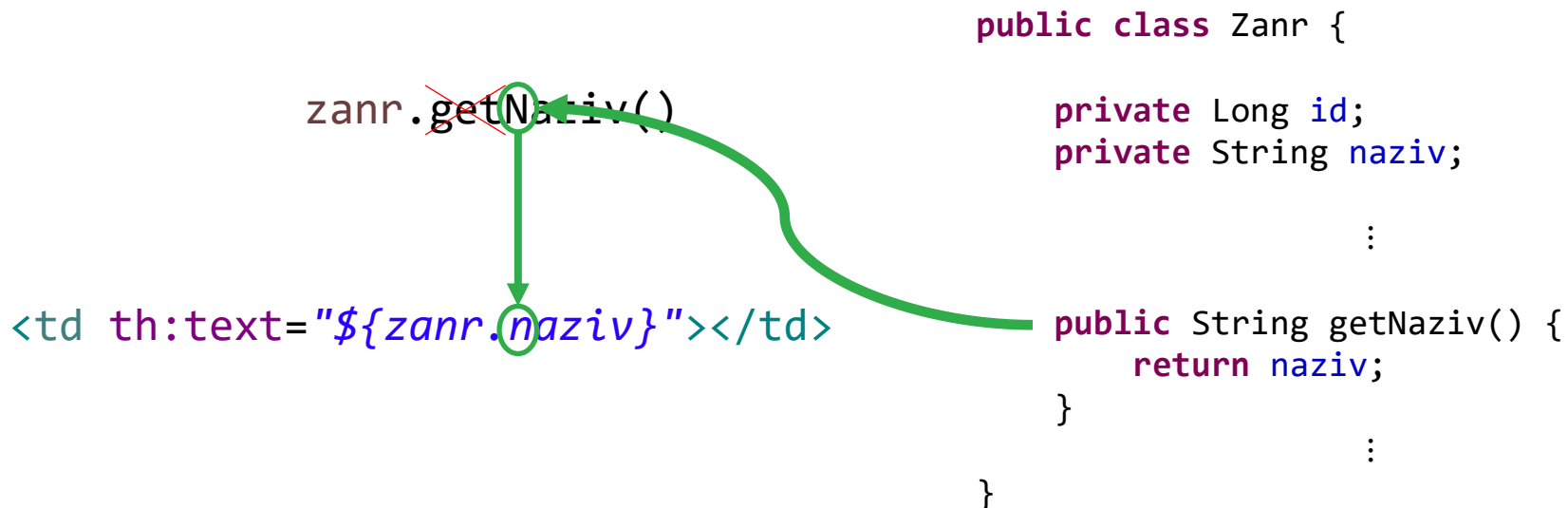
- ***`${...}`*** : *variable expressions* (izračunavaju vrednost na osnovu promenljivih)
- ostatak se može pronaći na sledećoj adresi:  
<https://www.thymeleaf.org/doc/articles/standarddialect5minutes.html>

# Thymeleaf

## Izrazi (*variable expressions*)

- izračunavaju vrednost na osnovu **promenljivih**
- oblik: *`${identifikator.property}`*
- da bi se atributi objekata mogli čitati, oni moraju imati implementirane **getter-e**

samo prvi  
karakter nakon  
*get* prefiksa se  
umanjuje



# Thymeleaf

## Izrazi (*variable expressions*)

- izračunavaju vrednost na osnovu **promenljivih**
- oblik: *`${identifikator.property}`*
- da bi se atributi objekata mogli čitati, oni moraju imati implementirane **getter-e**

samo prvi  
karakter nakon *is*  
prefiksa se  
umanjuje

~~korisnik.isAdministrator()~~

`<td th:text="${korisnik.administrator}"></td>`

```
public class Korisnik {  
  
    private String korisnickoIme;  
    private String lozinka;  
    private String eMail;  
    private String pol;  
    private boolean administrator;  
  
    :  
  
    public boolean isAdministrator() {  
        return administrator;  
    }  
  
    :  
}
```

## Izrazi (*variable expressions*)

- ako je *property* nekog objekta takođe objekat, može se čitati i *property* tog objekta, itd. do proizvoljne dubine

```
projekcija.getFilm().getId()
```

```
${projekcija.film.id}
```

## Upis tekstualnog sadržaja u HTML elemente

- u element se dodaje novi (nestandardni) HTML atribut **th:text**
- kao vrednost tog atributa se navodi **izraz**
- nakon popunjavanja šablona (*render*-ovanja), atribut će nestati, a vrednost izraza će se upisati u sadržaj elementa

```
"<tr><th>trajanje:</th><td>" + film.getTrajanje() + "</td></tr>\r\n"
```

```
<tr><th>trajanje:</th><td th:text="${film.trajanje}"></td>
```



```
<tr><th>trajanje:</th><td>182</td>
```



## Supstitucija

- važi za bilo koji tekstualni izraz
- izraz se umeće između znakova `/` zajedno sa statičkim tekstom
- nakon popunjavanja šablona (*render*-ovanja), vrednost izraza će se dodati na tekst između znakova `/`
- oblik: `/statički_tekstizraz/`

```
"<tr><th></th><td><a href=\"projekcije?filmId=\" + film.getId() + "\">projekcije</a></td></tr>\r\n"
```

```
<tr><th></th><td><a th:href="/projekcije?filmId=${film.id}"/>projekcije</a></td></tr>
```



statički tekst      izraz



```
<tr><th></th><td><a href="projekcije?filmId=1">projekcije</a></td></tr>
```

## Ponavljanje HTML elemenata

- ako se u elementu navede atribut **th:each**, **element** zajedno sa svojim podelementima se **ponavlja za svaki element kolekcije** koja je određena izrazom
- oblik: `<element th:each="element, status: ${kolekcija}">...</element>`
- **element** promenljiva poprima vrednost jednog po jednog elementa kolekcije
- **status** je pomoćna promenljiva koja sadrži dodatne informacije o iteraciji kroz elemente kolekcije:
  - **index**: indeks tekuće iteracije, počevši od 0
  - **count**: broj prođenih elemenata kolekcije
  - **size**: ukupan broj elemenata kolekcije
  - **even/odd**: vraća *true* ako je indeks tekuće iteracije neparan/paran
  - **first**: vraća *true* ako je tekući element prvi u kolekciji
  - **last**: vraća *true* ako je tekući element poslednji u kolekciji

## Ponavljanje HTML elemenata

```
<div class="row">
  <div class="col">
    <table class="table table-striped">
      <thead>
        <tr>
          <th>Redni broj</th>
          <th>Naziv</th>
          <th></th>
        </tr>
      </thead>
      <tbody class="table-group-divider">
        <tr th:each="itZanr, status: ${zanrovi}">
          <td class="text-end" th:text="${status.count}">1</td>
          <td><a th:href="/zanrovi/prikaz?id=${itZanr.id}|" th:text="${itZanr.naziv}">žanr 1</a></td>
          <td><a th:href="/filmovi?zanrId=${itZanr.id}|">filmovi</a></td>
        </tr>
      </tbody>
    </table>
  </div>
</div>
```

*itZanr* i *status* su  
lokalne promenljive koje  
su dostupne u svim  
podelementima

render



```
<div class="row">
  <div class="col">
    <table class="table table-striped">
      <thead>
        <tr>
          <th>Redni broj</th>
          <th>Naziv</th>
          <th></th>
        </tr>
      </thead>
      <tbody class="table-group-divider">
        <tr>
          <td class="text-end">1</td>
          <td><a th:href="/zanrovi/prikaz?id=1">naučna fantastika</a></td>
          <td><a th:href="/filmovi?zanrId=1">filmovi</a></td>
        </tr>
        <tr>
          <td class="text-end">2</td>
          <td><a th:href="/zanrovi/prikaz?id=2">akcija</a></td>
          <td><a th:href="/filmovi?zanrId=2">filmovi</a></td>
        </tr>
        <tr>
          <td class="text-end">3</td>
          <td><a th:href="/zanrovi/prikaz?id=3">komedija</a></td>
          <td><a th:href="/filmovi?zanrId=3">filmovi</a></td>
        </tr>
        <tr>
          <td class="text-end">4</td>
          <td><a th:href="/zanrovi/prikaz?id=4">horor</a></td>
          <td><a th:href="/filmovi?zanrId=4">filmovi</a></td>
        </tr>
        <tr>
          <td class="text-end">5</td>
          <td><a th:href="/zanrovi/prikaz?id=5">avantura</a></td>
          <td><a th:href="/filmovi?zanrId=5">filmovi</a></td>
        </tr>
      </tbody>
    </table>
  </div>
</div>
```

## Podešavanje

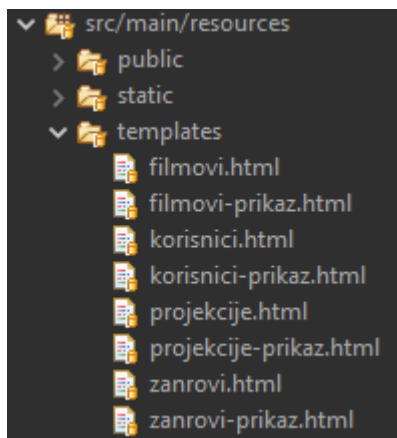
- da bi se *Thymeleaf* uključio u *Spring Boot* projekat, sledeća međuzavisnost se mora dodati u *pom.xml* datoteku:

```
<dependency>  
    <groupId>org.springframework.boot</groupId>  
    <artifactId>spring-boot-starter-thymeleaf</artifactId>  
</dependency>
```

# Thymeleaf

## Podešavanje

- *Thymeleaf* šabloni se smeštaju u *templates* poddirektorijum *resource* direktorijuma projekta:

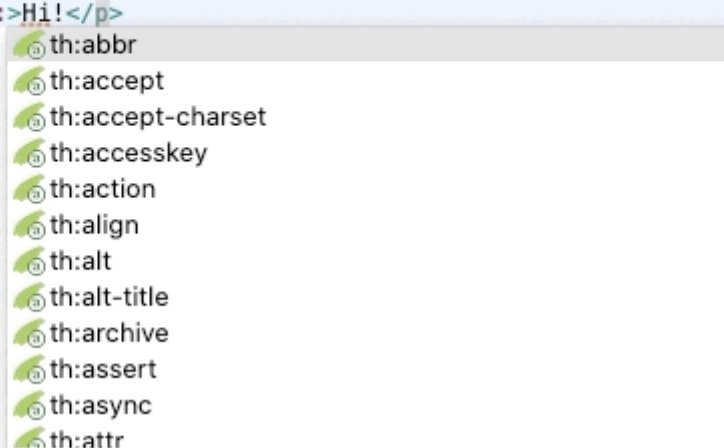


# Thymeleaf

## Podešavanje (opciono)

- da bi *Eclipse*-ov HTML editor imao **kontekstno-zavisnu podršku** za *Thymeleaf* direktive potrebno je instalirati **plugin** sa ove adrese:  
<https://marketplace.eclipse.org/content/thymeleaf-plugin-eclipse>
- nakon toga je još potrebno dodati **Thymeleaf namespace** u korenski element HTML dokumenta u svakom šablonu

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <title>Test</title>
</head>
<body>
  <p th:>Hi!</p>
</body>
</html>
```



# Thymeleaf

## Tok kontrole:

1. zahtev stiže do *controller*-a
  2. *controller* popunjava *ModelMap* objekat *podacima* zavedenim pod unapred odabranim *ključevima*
  3. *controller* prosleđuje *Thymeleaf*-u *naziv šablona*
- koristi se kada potrebno generisati dinamički HTML sadržaj na osnovu podataka koji su dobijeni iz servisnog sloja (čitanje iz baze, obrada i sl.)

# *HTTP GET*

## GET (bez parametara)

- Služi za traženje podataka od servera.

### **Primer HTTP zahteva:**

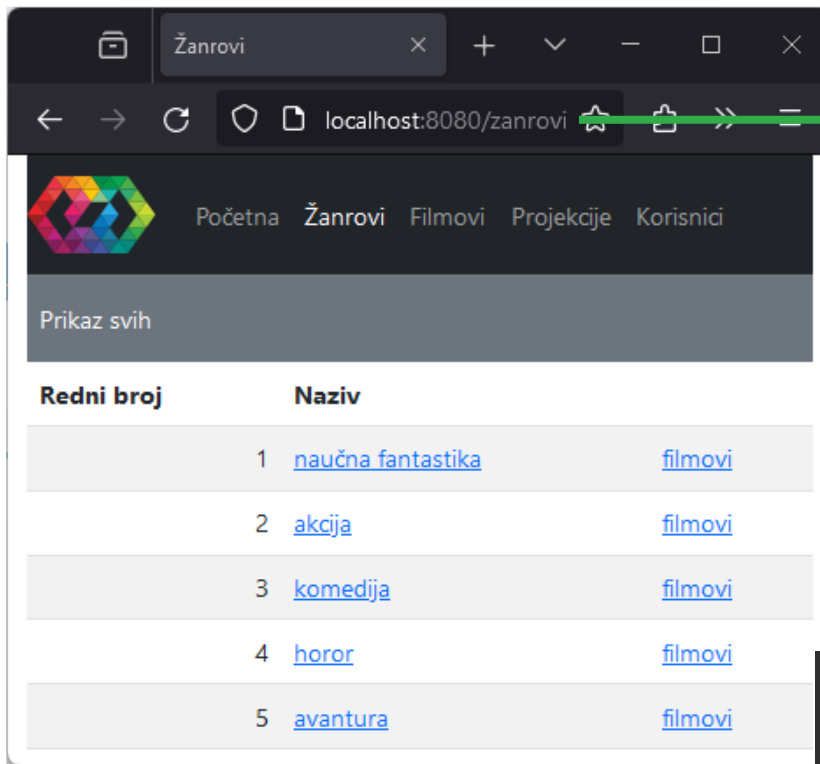
GET /zanrovi HTTP/1.1

Host: localhost:8080



# HTTP GET

## GET (bez parametara)



request

/zanrovi

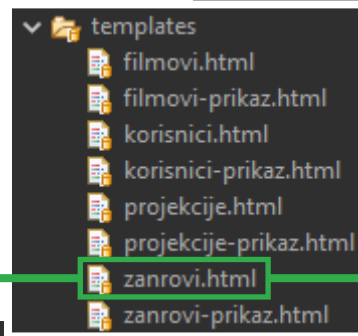
```
@Controller
@RequestMapping("/zanrovi")
public class ZandrController {
    private Bioskop bioskop;

    :

    @GetMapping("") // bez @ResponseBody
    public String getAll(ModelMap request) {
        request.addAttribute("zanrovi", bioskop.getZanrovi().values());
        return "zanrovi"; // forwarding na template
    }

    :
}
```

ne navodi se .html



request

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">

:

<tbody class="table-group-divider">
    <tr th:each="itZanr, status: ${zanrovi}">
        <td class="text-end" th:text="${status.count}">1</td>
        <td><a th:href="|/zanrovi/prikaz?id=${itZanr.id}|" th:text="${itZanr.naziv}">žanr 1</a></td>
        <td><a th:href="|/filmovi?zanrId=${itZanr.id}|">filmovi</a></td>
    </tr>
</tbody>

:

</html>
```

response

# HTTP GET

## GET (bez parametara)

Žanrovi

localhost:8080/zanrovi

Početna Žanrovi Filmovi Projekcije Korisnici

Prikaz svih

Redni broj	Naziv
1	<a href="#">naučna fantastika</a> <a href="#">filmovi</a>
2	<a href="#">akcija</a> <a href="#">filmovi</a>
3	<a href="#">komedija</a> <a href="#">filmovi</a>
4	<a href="#">horor</a> <a href="#">filmovi</a>
5	<a href="#">avantura</a> <a href="#">filmovi</a>

request

/zanrovi

```
@Controller
@RequestMapping("/zanrovi")
public class ZandrController {
    private Bioskop bioskop;

    :

    @GetMapping("") // bez @ResponseBody
    public String getAll(ModelMap request) {
        request.addAttribute("zanrovi", bioskop.getZanrovi().values());
        return "zanrovi"; // forwarding na template
    }

    :
}
```

templates

- filmovi.html
- filmovi-prikaz.html
- korisnici.html
- korisnici-prikaz.html
- projekcije.html
- projekcije-prikaz.html
- zanrovi.html
- zanrovi-prikaz.html

identifikator u *template-u*

request

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">

    :

    <tbody class="table-group-divider">
        <tr th:each="itZanr, status: ${zanrovi}">
            <td class="text-end" th:text="${status.count}">1</td>
            <td><a th:href="|/zanrovi/prikaz?id=${itZanr.id}|" th:text="${itZanr.naziv}">žanr 1</a></td>
            <td><a th:href="|/filmovi?zanrId=${itZanr.id}|">filmovi</a></td>
        </tr>
    </tbody>

    :

</html>
```

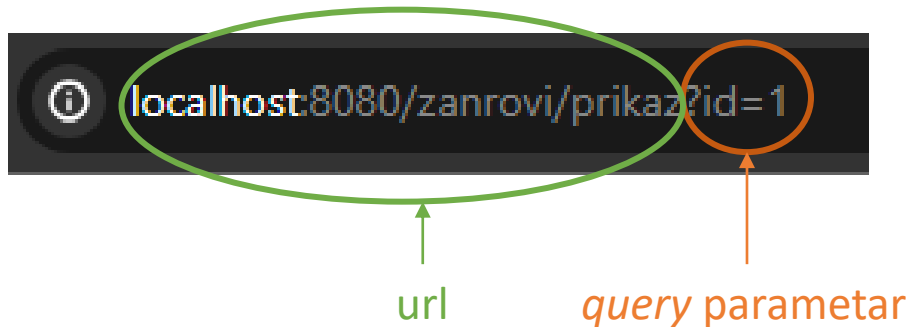
# Primer

- *com.ftninformatika.jwd.modul2.termin4.bioskop*

# HTTP GET

## GET (URL sa parametrima)

- Služi za traženje podataka od servera.
- Kod **GET** metode, posle znaka „?” zapisuju se vrednosti URL promenljivih u obliku ***ključ=vrednost&ključ=vrednost ...***
- Ovakvi parametri se nazivaju ***query*** parametri.



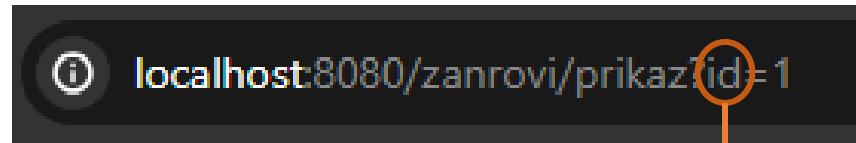
### Primer HTTP zahteva:

GET /zanrovi/prikaz?id=1 HTTP/1.1  
Host: localhost:8080

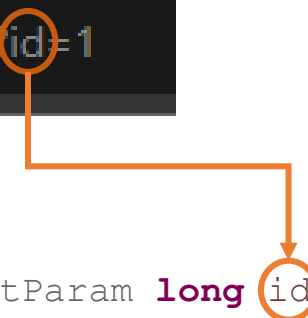
# Spring Boot

## Preuzimanje parametara

- Prihvatanje podataka koristeći `@RequestParam` anotaciju argumenata *handler* metode
- `@RequestParam` se koristi za preuzimanje **parametara upita**, **parametara iz formi** i **fajlova iz zahteva**.
- Ime argumenta metode mora da se poklapa sa imenom ulaznog parametara forme
- Moguća je automatska konverzija parametara u primitivni tip ili objekat *wrapper* klase



localhost:8080/zanrovi/prikaz?id=1



```
@GetMapping("/prikaz")
public String get(ModelMap request, @RequestParam long id) {
    request.addAttribute("zanr", bioskop.getZanrovi().get(id));
    return "zanrovi-prikaz";
}
```

# Spring Boot

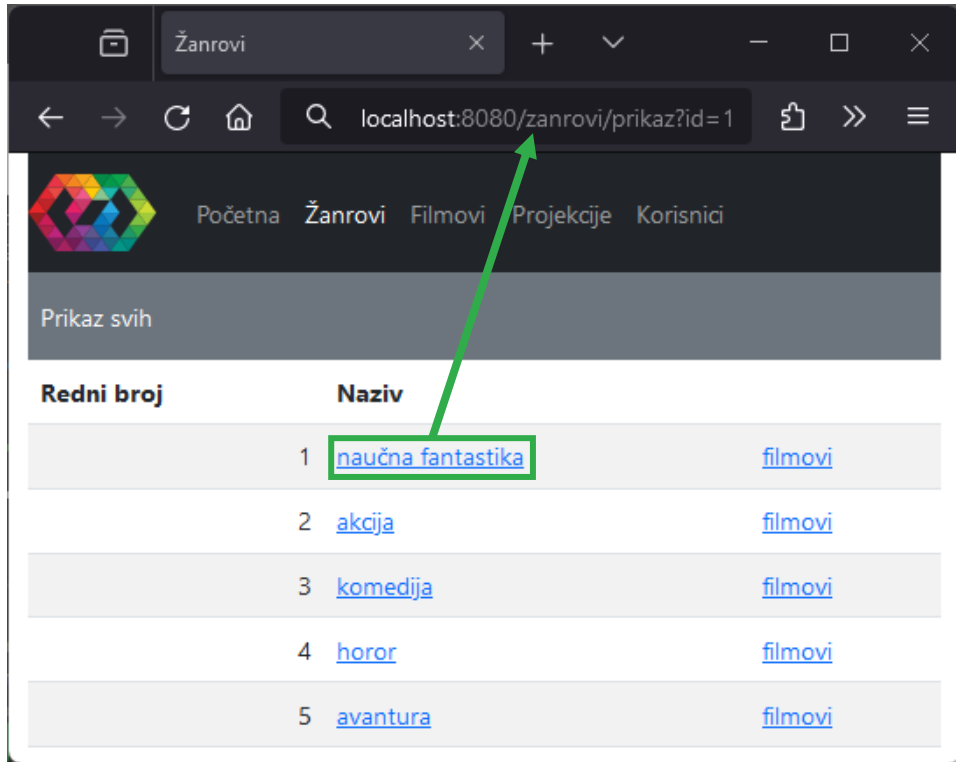
## Preuzimanje parametara

- `@RequestParam` anotacija može biti dodatno opisana atributima ***name***, ***value***, ***required*** i ***defaultValue***
- Atribut ***value*** se koristi kada je ime parametra različito od imena promenjive u *handler* metodi `@RequestParam(value="ime")` ili skraćena anotacija `@RequestParam("ime")`
- `@RequestParam` može biti i opcioni ukoliko se navede da je ***required=false***. U tom slučaju treba voditi računa da taj parametar može imati i ***null*** vrednost ukoliko nije prosleđen
- Varijanta opcione vrednosti `@RequestParam` za koji se može zadati predefinisana vrednost (ako se ne prosledi) moguće je sa atributom ***defaultValue***

```
@GetMapping("")
public String getAll(ModelMap request,
    @RequestParam(required = false, defaultValue = "0") long zanrId) {
    Collection<Film> rezultat = new ArrayList<>();
    for (Film itFilm: bioskop.getFilmovi().values()) {
        for (Zanr itZanr: itFilm.getZanrovi()) {
            if (zanrId == 0 || itZanr.getId() == zanrId) {
                rezultat.add(itFilm);
                break;
            }
        }
    }
    request.addAttribute("filmovi", rezultat);
    return "filmovi";
}
```

# Thymeleaf

## GET (URL sa parametrima)

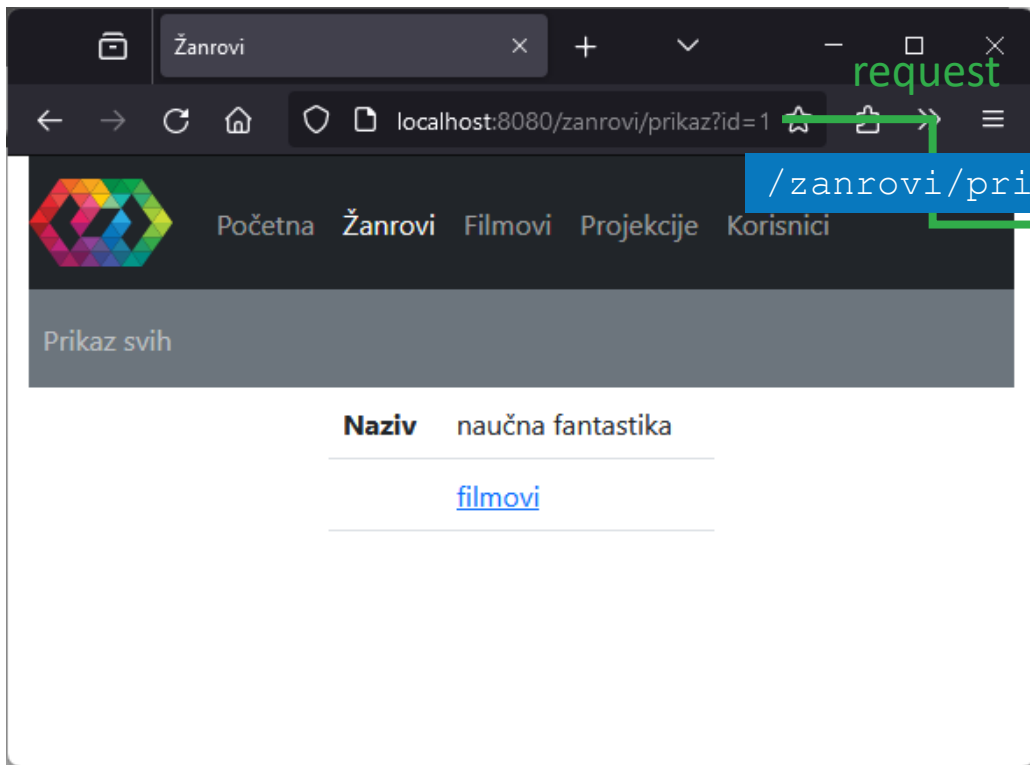


The screenshot shows a web browser window with the address bar displaying `localhost:8080/zanrovi/prikaz?id=1`. A green arrow points from the `id=1` parameter in the URL to the first row of the table, which contains the genre `naučna fantastika`. The table has two columns: `Redni broj` and `Naziv`. The first row is highlighted, and the text `naučna fantastika` is enclosed in a green box. The other rows in the table are `akcija`, `komedija`, `horor`, and `avantura`.

Redni broj	Naziv
1	naučna fantastika
2	akcija
3	komedija
4	horor
5	avantura

# Thymeleaf

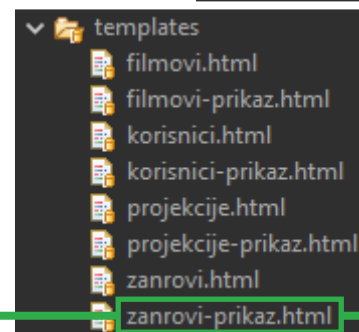
## GET (URL sa parametrima)



```
@Controller
@RequestMapping("/zanrovi")
public class ZandrController {
    private Bioskop bioskop;

    @GetMapping("/prikaz") // bez @ResponseBody
    public String get(ModelMap request, @RequestParam long id) {
        request.addAttribute("zanr", bioskop.getZanrovi().get(id));
        return "zanrovi-prikaz"; // forwarding na template
    }
}
```

ne navodi se .html

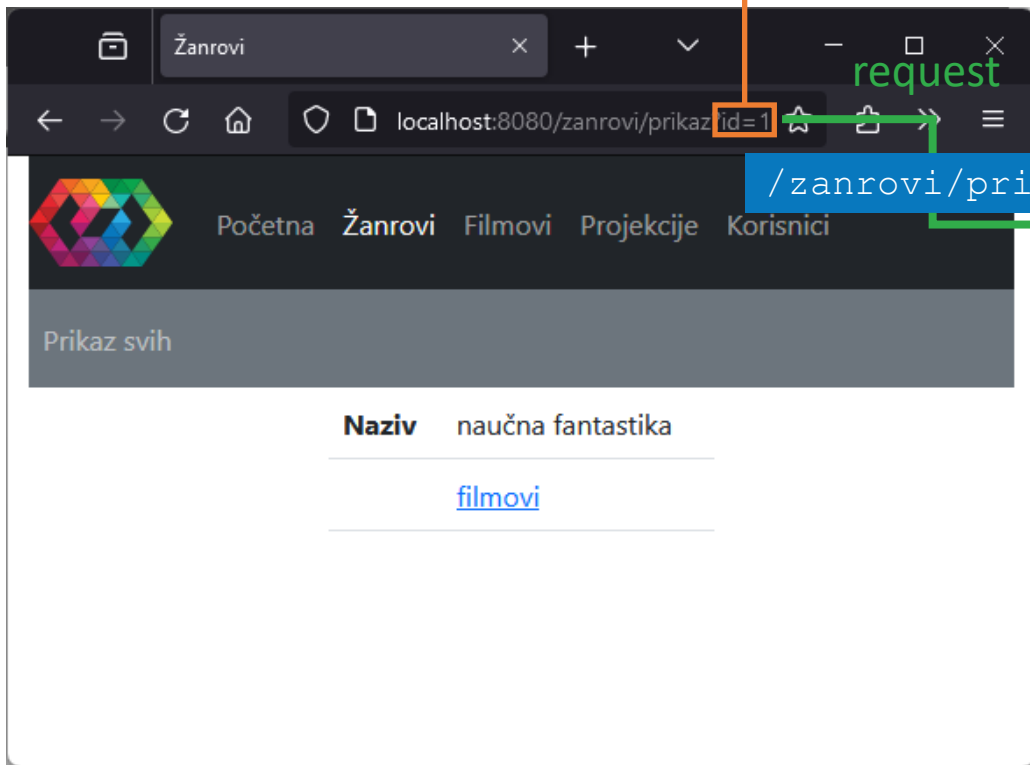


```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    ...
    <table class="table">
        <tr><th>Naziv</th><td th:text="${zanr.naziv}">žanr 1</td></tr>
        <tr><th></th><td><a th:href="|/filmovi?zanrId=${zanr.id}|">filmovi</a></td></tr>
    </table>
    ...
</html>
```



# Thymeleaf

## GET (URL sa parametrima)

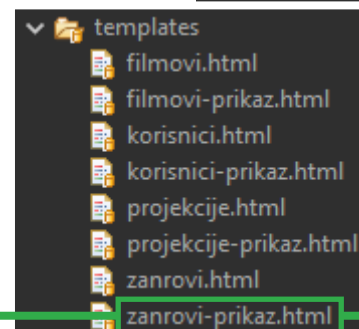


```
@Controller
@RequestMapping("/zanrovi")
public class ZandrController {
    private Bioskop bioskop;

    :

    @GetMapping("/prikaz") // bez @ResponseBody
    public String get(ModelMap request, @RequestParam long id) {
        request.addAttribute("zanr", bioskop.getZanrovi().get(id));
        return "zanrovi-prikaz"; // forwarding na template
    }

    :
}
```

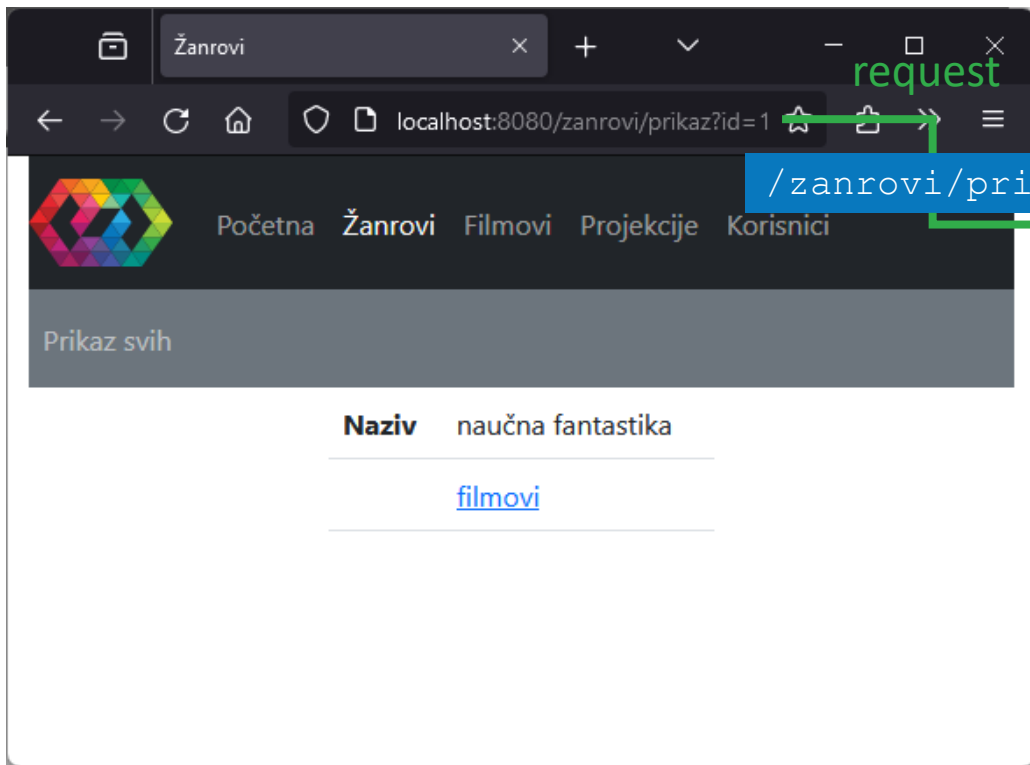


```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    :
    <table class="table">
        <tr><th>Naziv</th><td th:text="${zanr.naziv}">žanr 1</td></tr>
        <tr><th></th><td><a th:href="|/filmovi?zanrId=${zanr.id}|">filmovi</a></td></tr>
    </table>
    :
</html>
```

response

# Thymeleaf

## GET (URL sa parametrima)

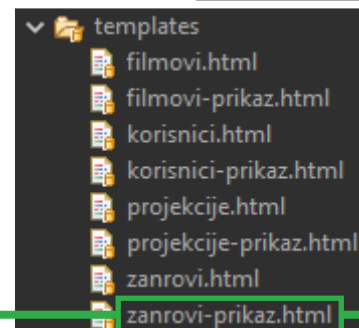


```
@Controller
@RequestMapping("/zanrovi")
public class ZandrController {
    private Bioskop bioskop;

    :

    @GetMapping("/prikaz") // bez @ResponseBody
    public String get(ModelMap request, @RequestParam long id) {
        request.addAttribute("zanr", bioskop.getZanrovi().get(id));
        return "zanrovi-prikaz"; // forwarding na template
    }

    :
}
```



identifikator u template-u

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    :
    <table class="table">
        <tr><th>Naziv</th><td th:text="${zanr.naziv}">žanr 1</td></tr>
        <tr><th></th><td><a th:href="|/filmovi?zanrId=${zanr.id}|">filmovi</a></td></tr>
    </table>
    :
</html>
```

# Primer

- *com.ftninformatika.jwd.modul2.termin4.bioskop*

## Izrazi (*variable expressions*)

- izrazi mogu biti i logički

```
korisnik == null  
${korisnik == null}
```

```
korisnik != null && korisnik.isAdministrator()  
${korisnik != null and korisnik.administrator}
```

*or*

*not*

## Izrazi (*variable expressions*)

- postoje i uslovni izrazi

```
<td th:text="${itKorisnik.administrator}? 'da': 'ne'}"></td>
```

↓ render

```
<td>da</td>
```

ili

```
<td>ne</td>
```

# Thymeleaf

## Izrazi (*variable expressions*)

- izrazi mogu sadržati i pozive ugrađenih funkcija
- oblik: `${#klasa.funkcija(argument1, argument2, ...)}`

```
film.getZanrovi().contains(itZanr)  
${#lists.contains(film.zanrovi, itZanr)}
```

```
korisnik.getPol().equals("muški")  
${#strings.equals(korisnik.pol, 'muški')}
```

```
projekcija.getDatumIVreme().format(DateTimeFormatter.ofPattern("dd.MM.yyyy. HH:mm"))  
${#temporals.format(projekcija.datumIVreme, 'dd.MM.yyy. HH:mm')}
```

samo prvi karakter nakon  
prefiksa se umanjuje!

## Zaključak

- Priprema/obrada podataka i prikaz podataka se sada nalaze u različitim datotekama!
- HTML kod je mnogo lakše *debug*-ovati u HTML *editor*-u nasuprot tome kad bi bio zapisan u *String* literalima u *controller*-ima!
- Programski kod je mnogo lakše *debug*-ovati u *controller*-ima!
- *Thymeleaf* koristiti kada god je potrebno dinamičko generisanje HTML sadržaja!
- HTML kod nikada više ne upisivati u *String* literale u *controller*-ima!

# Zadatak

- Implementirati generisanje dinamičkog HTML sadržaja uz pomoć *Thymeleaf*-a za kategorije.

U folderu *domaći* se nalazi projekat koji možete da iskoristite za izradu zadatka. U folderu *html* u okviru projekta *restoran* se nalaze html datoteke koje možete iskoristiti za kreiranje šablona. Potrebno je izmeniti kontroler po uzoru na projekat bioskop, tako da ima *handler* metode za dobavljanje svih kategorija i za prikaz informacija o konkretnoj kategoriji.



# Domaći zadatak

Implementirati generisanje dinamičkog HTML sadržaja uz pomoć *Thymeleaf*-a za ostatak aplikacije.

# Dodatni materijali

- <https://www.thymeleaf.org/doc/articles/standarddialect5minutes.html>
- <https://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.html>