



Osnove web programiranja

Thymeleaf template engine

Termin 8

Sadržaj

1. Nedostaci dosadašnjeg pristupa u generisanju dinamičkog HTML sadržaja
2. *Template engine*
3. *Thymeleaf*
 - I. Podešavanje
 - II. izvori podataka
 - III. tok kontrole
 - IV. standardni dijalekat
 - V. Izrazi (variable, selection, link, message, fragments)
 - VI. Literali
 - VII. Operatori
 - VIII. Korišćenje ugrađenih objekata
 - IX. Flexible layouts
 - X. načini popunjavanja šablona
 - XI. Uslovni prikaz HTML elementa i atributa
 - XII. Ponavljanje HTML elemenata
 - XIII. Blokovi
4. USE CASE korišćenje Thymeleaf za Bloskop web aplikaciju
5. Zaključak
6. Ideja za vežbu

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

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

```
out.append(
    "<table class=\"tabela\">\r\n" +
    "    <caption>Žanrovi</caption>\r\n" +
    "    <tr>\r\n" +
    "        <th>r. br.</th>\r\n" +
    "        <th>naziv</th>\r\n" +
    "        <th></th>\r\n" +
    "    </tr>\r\n");
// meni: žanrovi
for (int it = 0; it < zanrovi.size(); it++) {
    out.append(
        "    <tr>\r\n" +
        "        <td>" + (it + 1) + "</td>\r\n" +
        "        <td><a href=\"Zanrovi/Details?id=\"" + zanrovi.get(it).getId() + "\">" + zanrovi.get(it).getNaziv() +
        "</a></td>\r\n" +
        "        <td><a href=\"Filmovi?zanrId=\"" + zanrovi.get(it).getId() + "\">filmovi</a></td>\r\n" +
        "    </tr>");
}
out.append(
    "</table>\r\n");
```

Template engine

Ideja

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

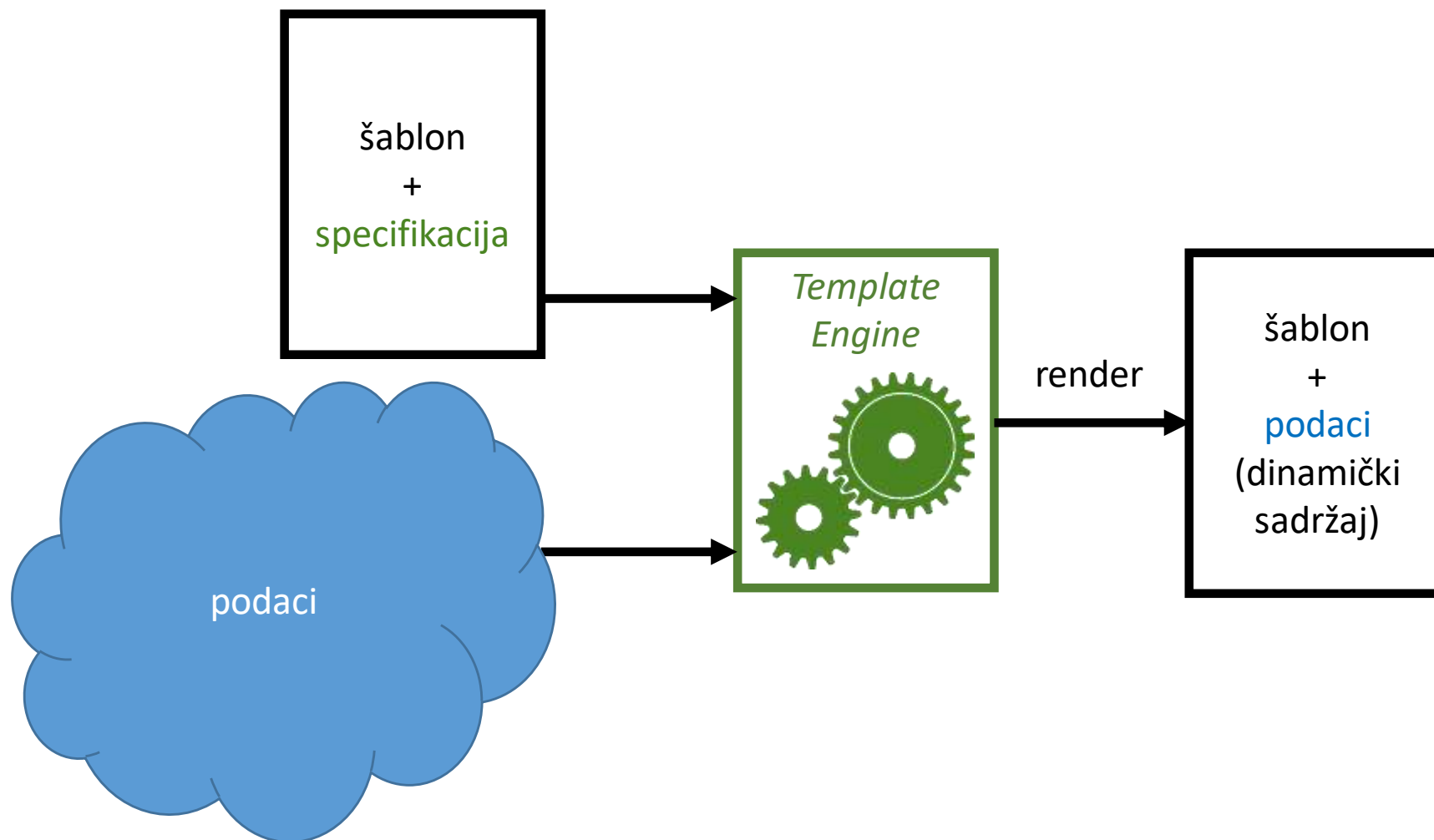
```
<table class="tabela">
  <caption>Žanrovi</caption>
  <tr>
    <th>r. br.</th>
    <th>naziv</th>
    <th></th>
  </tr>
  <tr th:each="itZanr, status: ${zanrovi}">
    <td th:text="${status.index + 1}"></td>
    <td><a th:href="/Zanrovi/Details?id=${itZanr.id}|" th:text=${itZanr.naziv}></a></td>
    <td><a th:href="/Filmovi?zanrId=${itZanr.id}|">filmovi</a></td>
  </tr>
</table>
```

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

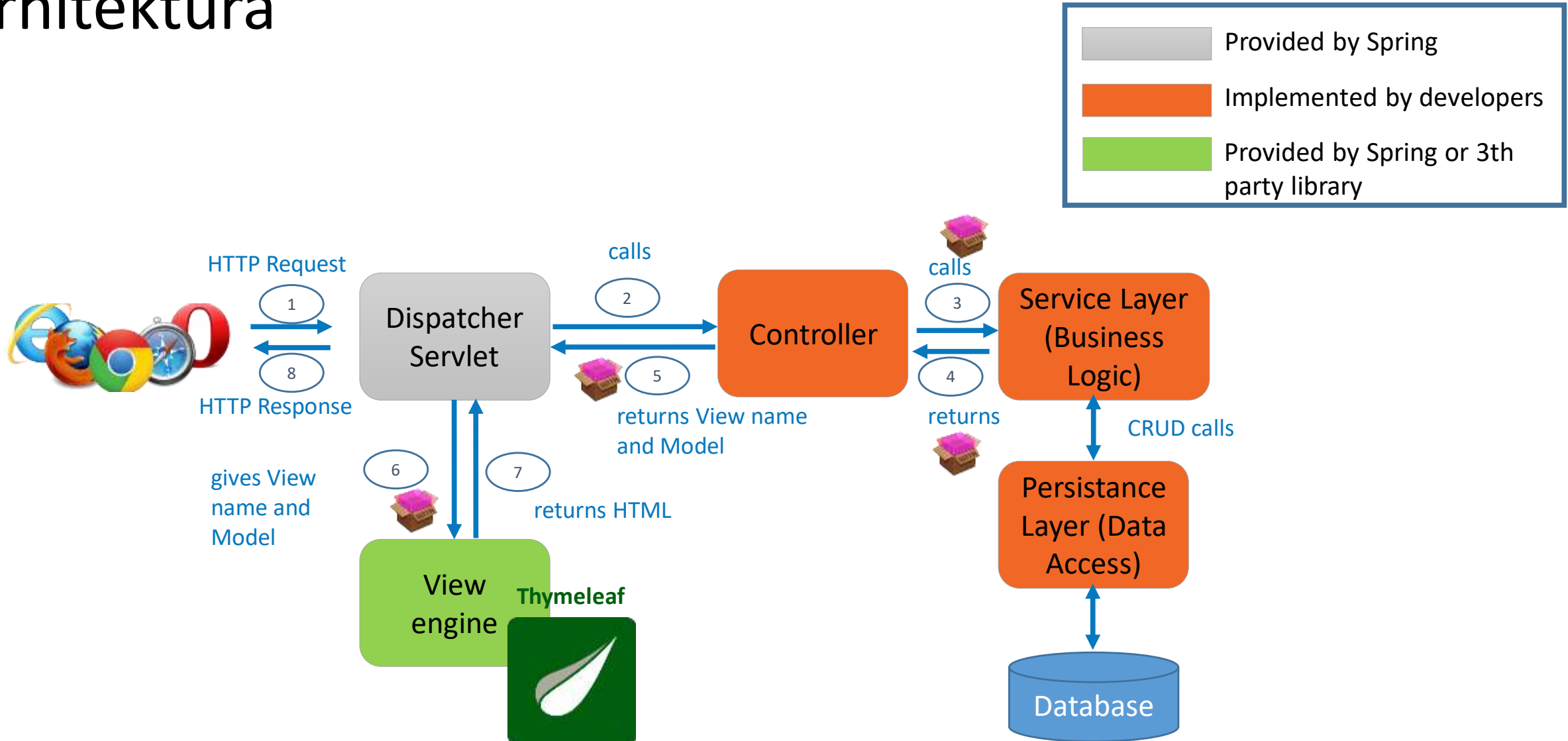


thyme = majčina dušica

- 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

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>
```

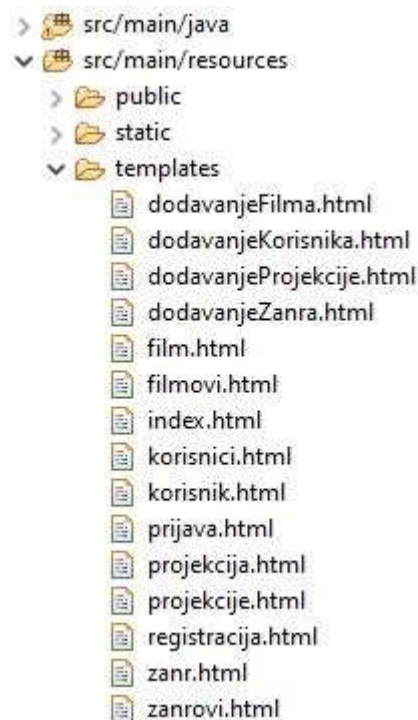
- da bi se *Thymeleaf* uključio u *Spring Boot* projekat, sledeći unos se mora dodati u *application.properties* datoteku:

```
spring.thymeleaf.enabled=true
```

Thymeleaf

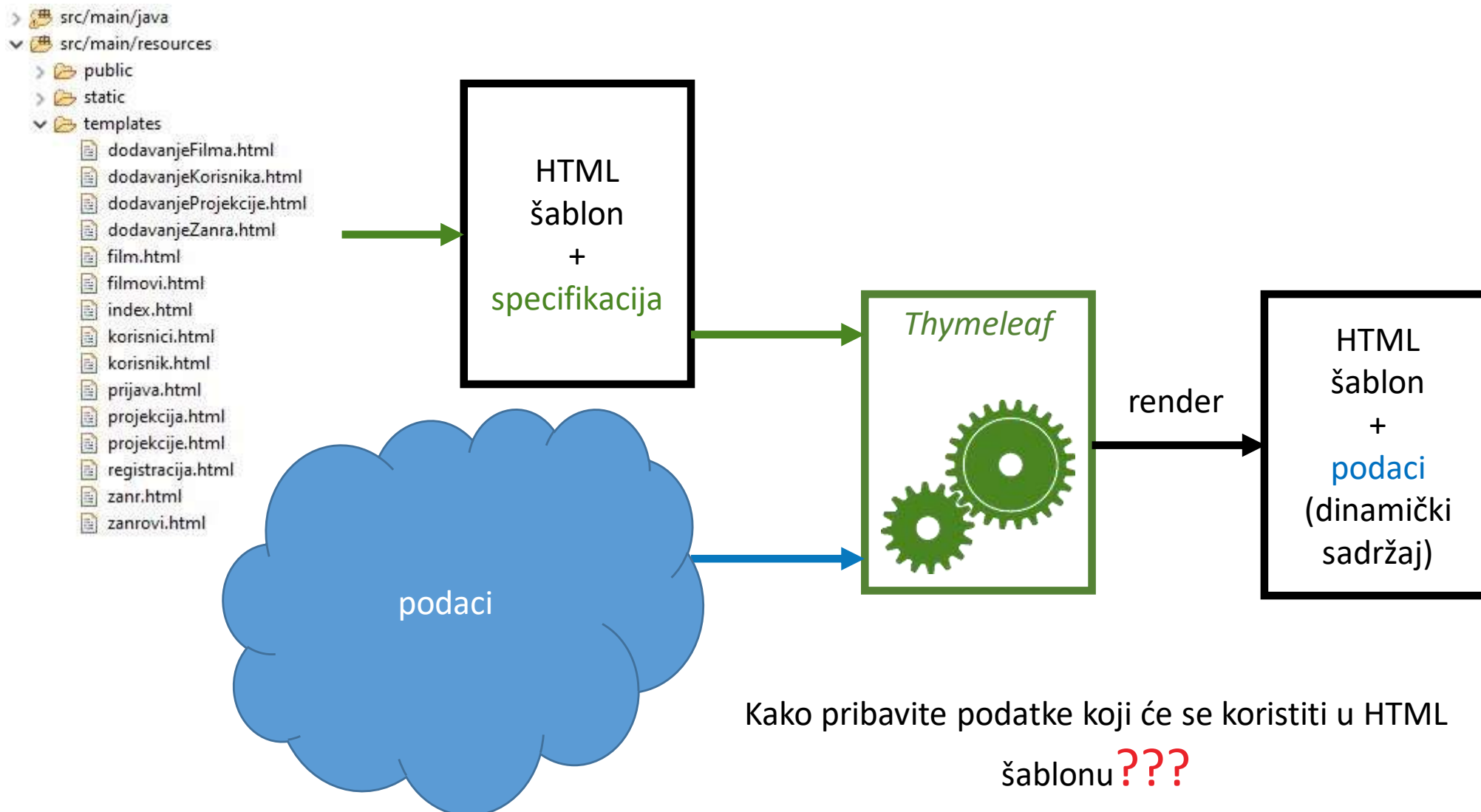
Podešavanje

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



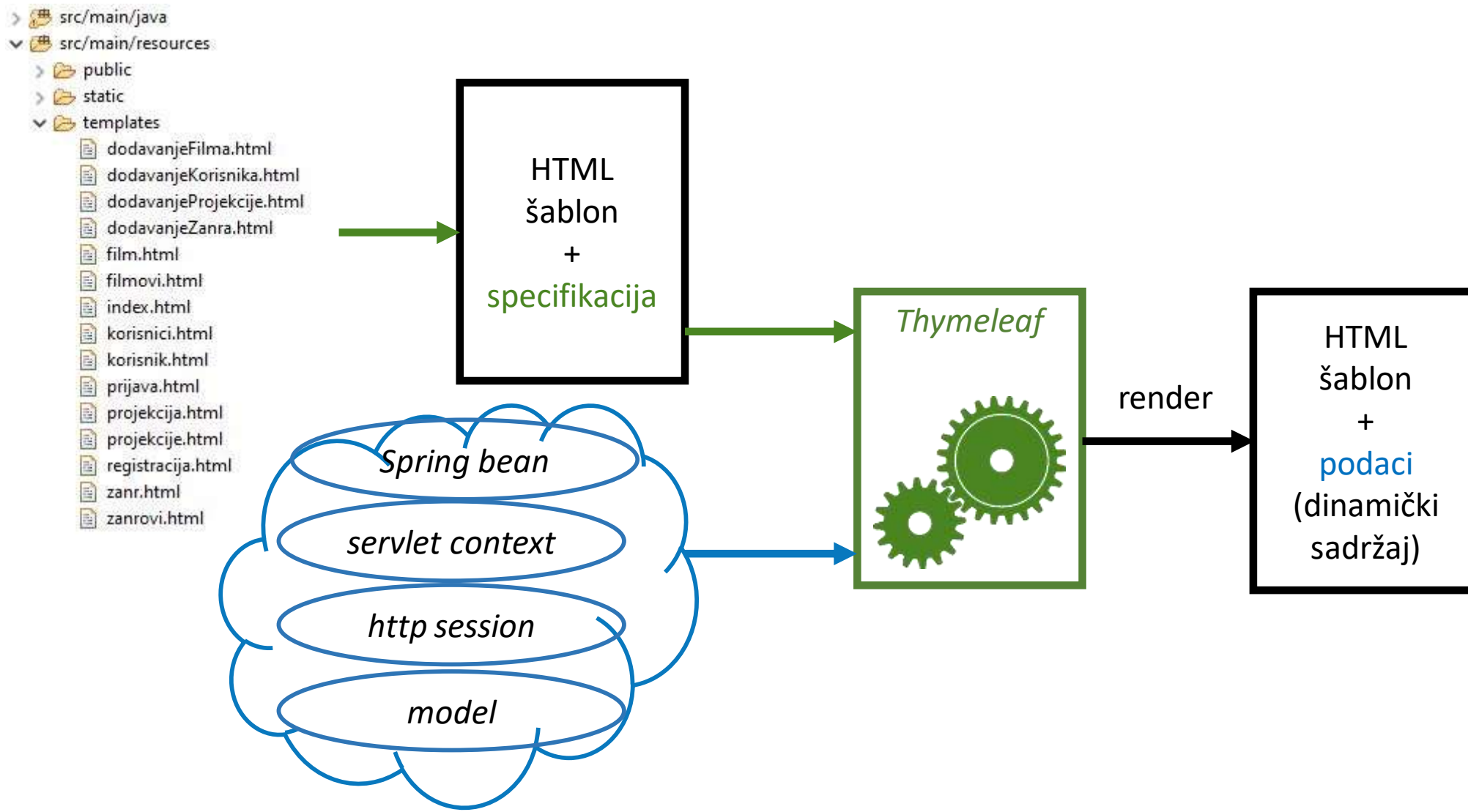
Thymeleaf

Izvori podataka



Thymeleaf

Izvori podataka



Thymeleaf

Čitanje iz *Spring Bean*-a

```
@Configuration
public class SecondConfiguration implements WebMvcConfigurer{

@Bean(name= {"memorijaAplikacije"}, initMethod="init", destroyMethod="destroy")
public MemorijaAplikacije getMemorijaAplikacije() {
    return new MemorijaAplikacije();
}

public class MemorijaAplikacije extends HashMap {

    public void init() {
        Korisnik korisnik1 = new Korisnik("usernameKor", "passKor", "korisnik1@korisnik1", "muški", true)
        this.put("korisnik1", korisnik1);
    }
    ...
}
```



```
<p th:text="${@memorijaAplikacije.get('korisnik1').eMail}"> </p>
```



Thymeleaf

Čitanje iz *servlet context*-a

```
@Controller
@RequestMapping(value="/Filmovi")
public class FilmoviController implements ServletContextAware {

    public static final String STATISTIKA_FILMOVA_KEY = "statistikaFilmova";
```

```
@Component
public final class InitServletContextInitializer implements ServletContextInitializer {

    @Override
    public void onStartup(ServletContext servletContext) throws ServletException {
        servletContext.setAttribute(FilmoviController.STATISTIKA_FILMOVA_KEY, new FilmStatistika());
    }

}
```

```
<li th:each="brojac: ${#servletContext.getAttribute('statistikaFilmova').filmovi}">
...
</li>
```

Thymeleaf

Čitanje iz sesije

```
@Controller
@RequestMapping(value="/Filmovi")
public class FilmoviController implements ServletContextAware {

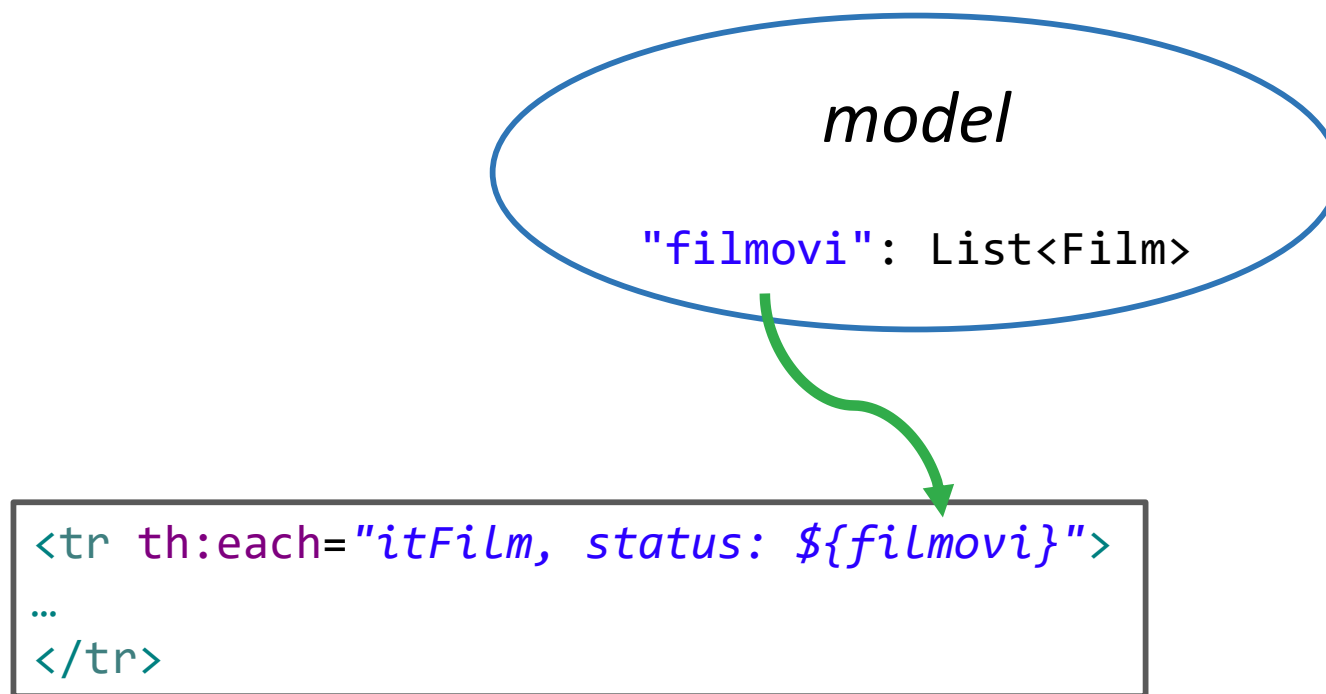
    public static final String POSECENI_FILMOVI_ZA_KORISNIKA_KEY = "poseceniFilmoviZaKorisnika";
```

```
@Component
public class InitHttpSessionListener implements HttpSessionListener {

    public void sessionCreated(HttpSessionEvent event) {
        HttpSession session = event.getSession();
        session.setAttribute(FilmoviController.POSECENI_FILMOVI_ZA_KORISNIKA_KEY, new ArrayList<Film>());
    }
```

```
<li th:each="itFilm: ${session.poseceniFilmoviZaKorisnika}"> ... </li>
```


Čitanje iz modela



Thymeleaf

Tok kontrole: slučaj #1

```
@Controller
@RequestMapping(value="/")
public class IndexController {

    @GetMapping ← bez @ResponseBody
    public String index() {
        return "index"; ← bez ".html"
    }
}
```

src/main/java
src/main/resources
public
static
templates
dodavanjeFilma.html
dodavanjeKorisnika.html
dodavanjeProjekcije.html
dodavanjeZanra.html
film.html
filmovi.html
index.html
korisnici.html
korisnik.html
prijava.html

zahtev

Spring bean

servlet context

http session

Thymeleaf

render

index.html
+
podaci
(dinamički
sadržaj)

odgovor

IndexController

Thymeleaf

Tok kontrole: slučaj #1

1. zahtev stiže do *controller*-a
 2. *controller* prosleđuje *Thymeleaf*-u **naziv šablona**
- koristi se kada potrebno generisati dinamički HTML sadržaj na osnovu podataka koji se nalaze isključivo u *Spring bean*-u, *servlet context*-u i/ili u sesiji, npr. da li je korisnik prijavljen

Thymeleaf

Tok kontrole: slučaj #2

```
@Controller
@RequestMapping(value="/Filmovi")
public class FilmoviController {

    @GetMapping(value="/Details")
    public ModelAndView details(@RequestParam Long id) {
        Film film = filmService.findOne(id);
        List<Zanr> zanrovi = zanrService.findAll();

        ModelAndView rezultat = new ModelAndView("film");
        rezultat.addObject("film", film);
        rezultat.addObject("zanrovi", zanrovi);

        return rezultat;
    }
}
```

bez @ResponseBody

bez ".html"

zahtev

src/main/java
src/main/resources
public
static
templates
dodavanjeFilma.html
dodavanjeKorisnika.html
dodavanjeProjekcije.html
dodavanjeZanra.html
film.html
filmovi.html
index.html
korisnici.html
korisnik.html
prijava.html

Spring bean

servlet context

http session

Thymeleaf

film.html
+
podaci
(dinamički
sadržaj)

render

odgovor

Tok kontrole: slučaj #2

1. zahtev stiže do *controller*-a
 2. *controller* kreira *ModelAndView* objekat koji sadrži naziv šablona zajedno sa podacima i prosleđuje ga *Thymeleaf*-u
- koristi se kada potrebno generisati dinamički HTML sadržaj na osnovu podataka koji su dobijeni iz servisnog sloja (čitanje iz baze, obrada i sl.)
 - ako je potrebno (a često jeste), dodatno je moguće pročitati podatke i iz *Spring Bean*-a, *servlet context*-a i/ili sesije

Thymeleaf

Tok kontrole: slučaj #2 – varijacija 2

```
@Controller
@RequestMapping(value="/Filmovi")
public class FilmoviController {

    @GetMapping(value="/Details")
    public String details(@RequestParam Long id,
        ModelMap map) {
        Film film = filmService.findOne(id);
        List<Zanr> zanrovi = zanrService.findAll();

        map.put("film", film);
        map.put("zanrovi", zanrovi);

        return "film";
    }
}
```

bez @ResponseBody

zahtev

src/main/java
src/main/resources
public
static
templates
dodavanjeFilma.html
dodavanjeKorisnika.html
dodavanjeProjekcije.html
dodavanjeZanra.html
film.html
filmovi.html
index.html
korisnici.html
korisnik.html
prijava.html

bez ".html"

"film"

model

"film": Film
"zanrovi": List<Zanr>

Spring bean

servlet context

http session

Thymeleaf

film.html
+
podaci
(dinamički
sadržaj)

render

odgovor

Tok kontrole: slučaj #2 – varijacija 2

1. zahtev stiže do *controller*-a
 2. *controller* popunjava *ModelMap* objekat koji sadrži podatke i vraća naziv šablona koji prosleđuje *Thymeleaf*-u
- koristi se kada potrebno generisati dinamički HTML sadržaj na osnovu podataka koji su dobijeni iz servisnog sloja (čitanje iz baze, obrada i sl.)
 - ako je potrebno (a često jeste), dodatno je moguće pročitati podatke i iz *Spring Bean*-a, *servlet context*-a i/ili sesije

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

Thymeleaf

Standardni dijalekat

- oblik: `<element th:atribut="izraz"></element>`

```
<table class="tabela">
  <caption>Žanrovi</caption>
  <tr>
    <th>r. br.</th>
    <th>naziv</th>
    <th></th>
  </tr>
  <tr th:each="itZanr, status: ${zanrovi}">
    <td th:text="${status.index + 1}"></td>
    <td><a th:href="/Zanrovi/Details?id=${itZanr.id}/" th:text=${itZanr.naziv}></a></td>
    <td><a th:href="/Filmovi?zanrId=${itZanr.id}/">filmovi</a></td>
  </tr>
</table>
```

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* (dodela vrednosti na osnovu izraza koji sadrži promenljive)
- **$*\{...\}$** : *selection expressions* (dodela vrednosti na osnovu izraza koji pristupa delovima prethodno odabranog objekta)
- **$\#\{...\}$** : *message (i18n) expressions* (pribavljanje locale specifičnih poruka iz spoljnog izvora)
- **$@\{...\}$** : *link (URL) expressions* (formiraju URL-ove u odnosu na bazični URL)
- **$\sim\{...\}$** : *fragment expressions* (pribavljanje fragmenata drugih HTML stranica)
- ostatak se može pronaći na sledećoj adresi:
<https://www.thymeleaf.org/doc/articles/standarddialect5minutes.html>

Izrazi - *variable expressions*

- predstavljaju Object-Graph Navigation Language (**OGNL**) expressions.
 - **OGNL** je open source **Expression** Language (EL) za Java jezik
- *Variable expressions* se izvršavaju nad **context promenljivama** koje se u Spring-u još nazivaju i **model attributes** promenljive (u Handler metodama se u Model dodaju atributi po principu ključ i vrednost. U parametar metode ModelAndView **map** ili povratavrednost metode ModelAndView, a zatim se model prosleđuje pogledu)
- Koriste se za **dodelu vrednosti** na definisanoj poziciji u HTML na osnovu izraza koji sadrži **promenljive**
- oblik: ***`${identifikator.property}`***
- *identifikator* predstavlja **ključ** pod kojim se objekat dodat u model, dok *property* predstavlja **atribut** objekta.
- da bi se atributi objekata mogli čitati, oni moraju imati implementirane **getter-e**

Thymeleaf

Izrazi - *variable expressions*

- **dodela vrednosti** na osnovu izraza koji sadrži **promenljive**
- oblik: `${identifikator.property}`
- da bi se atributi objekata mogli čitati, oni moraju imati implementirane **getter-e**

samo prvi
karakter nakon
get prefiksa se
umanjuje

```
<input type="hidden" name="id" th:value="${zanr.id}">
```

zanr.getId()

```
public class Zanr {  
  
    private Long id;  
    private String naziv;  
  
    public Zanr(Long id, String naziv) {  
        this.id = id;  
        this.naziv = naziv;  
    }  
  
    public Long getId() {  
        return id;  
    }  
}
```

Objekat klase **Zanr** pod ključem **zanr** je bio prethodno dodat u **Model** i prosleđen pogledu

Thymeleaf

Izrazi - *variable expressions*

- **dodela vrednosti** na osnovu izraza koji sadrži **promenljive**
- oblik: *`${identifikator.property}`*
- da bi se atributi objekata mogli čitati, oni moraju imati implementirane **getter-e**

samo **prvi**
karakter nakon *is*
prefiksa se
umanjuje

statistikaFilмова.isEmpty()

```
public class FilmStatistika {  
  
    private Map<Long, FilmBrojac> popularniFilmovi;  
  
    public FilmStatistika() {  
        popularniFilmovi = new HashMap<>();  
    }  
  
    public boolean isEmpty() {  
        return popularniFilmovi.isEmpty();  
    }  
}
```

`<table class="horizontalni-meni" th:unless="${statistikaFilмова.empty}">`

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}">neki tekst</td>
```



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

Upis vrednosti HTML atributa (bilo kog)

- bilo koji standardni HTML atribut se proširuje prefiksom **th:**
- kao vrednost tog atributa se navodi **izraz**
- nakon popunjavanja šablona (*render*-ovanja), atribut će ostati, a vrednost izraza će se upisati u vrednost atributa

```
"<base href=\"\" + baseUrl + "\">\r\n"
```

```
<base th:href="${baseUrl}">
```



```
<base href="/Bioskop/">
```

```
"<input type=\"hidden\" name=\"id\" value=\"\" + film.getId() + "\"/>\r\n"
```

```
<input type="hidden" name="id" th:value="${film.id}"/>
```



```
<input type="hidden" name="id" value="1"/>
```


Literali

- U okviru Thymeleaf izraza moguće je korićenje literala:
 - *Text literals*: 'one text', 'Another one!',...
 - *Number literals*: 0, 34, 3.0, 12.3,...
 - *Boolean literals*: true, false
 - *Null literal*: null
 - *Literal tokens*: jedan, nekiTekst, moj.Tekst, moj.Tekst2, ...

Literali - Text literals

- *Text literals* – predstavljaju karatere u stringu koji se navode između jednostrukih apostrofa **'tekst'**. Ako je potrebno navesti karakter jednostruki apostrof tada se on eskejpuje **'rekao \'važi\' kako da ne'**.

```
<h3>Text Literals</h3>
```

```
<p>
```

```
    Pera je <span th:text="'rekao \'važi\' kako da ne'"> Jovane</span>.
```

```
</p>
```



Text Literals

Pera je rekao 'važi' kako da ne.

Literali - Number literals

- *Number literals* – predstavljaju brojeve nad kojima je moguće raditi aritmetičke operacije

```
<h3>Number Literals</h3>
```

```
<p>Trenutna godina je <span th:text="2020">neki tekst</span>.</p>
```

```
<p>Naredna godina je <span th:text="2020 + 1">neki tekst</span>.</p>
```



Number Literals

Trenutna godina je 2020.

Naredna godina je 2021.

Literali – Boolean literals

- Boolean *literals* – predstavljaju vrednost *true* ili *false* nad kojima je moguće raditi logičke operacije

```
<h3>Boolean Literals</h3>
```

```
<p th:if="5>3 == true">Tekst se ispisuje ako je izraz "5>3 == true" tačan</span>.</p>
```

```
<p th:if="5<3 == false">Tekst se ispisuje ako je izraz "5<3 == false" tačan</span>.</p>
```



Boolean Literals

Tekst se ispisuje ako je izraz "5>3 == true" tačan.

Tekst se ispisuje ako je izraz "5<3 == false" tačan.

Literali – Literal tokens

- *Literal tokens* – predstavljaju simplifikaciju text literala u kome se tekst sastoji samo od jedne reči (nema razmaka, nema simola “,”).
- *Dozvoljeno je korišćenje*
 - slova (A-Z i a-z)
 - brojeva (0-9)
 - zagrada ([i])
 - tačke (.)

```
<h3>Literal tokens</h3>
<p th:text="jedan">neki tekst</p>
<p th:text="'dva'">neki tekst</p>
<p th:text="moj.Tekst">neki tekst</p>
<p th:text="moj.Tekst2">neki tekst</p>
```

render



Literal tokens

jedan

dva

moj.Tekst

moj.Tekst2

LiteralsController i literals.html

Operatori – aritmetički

- U okviru teksta koji procesira Thymeleaf kao izraz dozvoljeno je korišćenje **aritmetičkih operatora** ako su vrednosti promenljivih brojevi, tada je rezultat izraza **numerička vrednost**
 - Binarni operatori : +, -, *, /, %
 - Minus znak (unarni operator): -

Sabiranje 2 + 1 je: ` neki tekst
`

Oduzimanje 5 - 1 - 1 je: ` neki tekst
`

Množenje 10*2 je: ` neki tekst
`

Deljenje 10/2 je: ` neki tekst
`

Ostatak pri deljenju 5%2 je: ` neki tekst
`



Sabiranje 2 + 1 je: 3

Oduzimanje 5 - 1 - 1 je: 3

Množenje 10*2 je: 20

Deljenje 10/2 je: 5

Ostatak pri deljenju 5%2 je: 1

OperatorsController i operators.html

Operatori – konkatencija teksta

- U okviru samog *variable expressions* dozvoljeno je korišćenje **operatora +** ako barem jedna od **vrednosti String**
 - Binarni operator + radi konatenciju teksta

```
<p th:text="'ovo je' + ' neki ' + 'tekst'">neki tekst</p>  
<p th:text="'Rezultat zbira 2 i 3 je:' + (2+3)">neki tekst</p>  
<p th:text="'Rezultat zbira 2 i 3 je:' + 2 + 3">neki tekst</p>
```

↓ render

ovo je neki tekst

Rezultat zbira 2 i 3 je:5

Rezultat zbira 2 i 3 je:23

Operatori – relacioni

- Izrazi mogu biti i **logički**, gde je rezultujuća vrednost True ili False
- Dozvoljeno je korišćenje relacionih operatora
 - Operatori poređenja: >, <, >=, <= (gt, lt, ge, le)
 - Operatori jednakosti: ==, != (eq, ne)

Rezultat `5 > 3 == true` je: ` 3 == true">
`

Rezultat `5 lt 3 == true` je: `
`

Rezultat `5 > 3 == true` je: ` 3} == true">
`

Rezultat `5 lt 3 == true` je: `
`



Rezultat `5 > 3 == true` je: true

Rezultat `5 lt 3 == true` je: false

Rezultat `5 > 3 == true` je: true

Rezultat `5 lt 3 == true` je: false

Operatori – logički

- Izrazi mogu biti i **logički**, gde je rezultujuća vrednost True ili False
- Dozvoljeno je korišćenje logičkih operatora
 - Binarni operator: and, or
 - Logička negacija(unarni operator): !, not

Rezultat `5%2==1 ∧ 5>3` je: `3">
`

Rezultat `5%2==1 ∧ 5<3` je: `<span th:text="5%2==1 and 5<3">
`

Rezultat `5%2==1 ∨ 5<3` je: `<span th:text="5%2==1 or 5<3">
`

Rezultat `¬(5>3)` je: `3)">
`

Rezultat `¬(5>3)` je: `3)">
`



Rezultat `5%2==1 ∧ 5>3` je: true

Rezultat `5%2==1 ∧ 5<3` je: false

Rezultat `5%2==1 ∨ 5<3` je: true

Rezultat `¬(5>3)` je: false

Rezultat `¬(5>3)` je: false

Korišćenje ugrađenih objekata

Nastavak naredni cas

- Sa ciljem da se postigne veća fleksibilnost izraza OGNL jezika omogućeno je korišćenje bazičnih objekata (**Basic Objects**) i pomoćnih objekata (**Utility Objects**)
- Objekti se pozivaju u oblik: *#naziv_objekta*

<https://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.html#appendix-a-expression-basic-objects>

<https://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.html#appendix-b-expression-utility-objects>

ObjektiController i objekti.html

Izrazi – Basic Objects

- Bazični objekti (**Basic Objects**) su:
 - **#ctx**: the context object `org.thymeleaf.context.WebContext` implements `IWebContext`. Sadrži ostale navedene ispod `${#ctx.request}`
 - **#vars**: the context variables.
 - **#locale**: the context locale. Direkna veza sa `java.util.Locale` koji je asociran sa trenutnim zahtevom `#Locale.Language`
 - **#request**: (only in Web Contexts) the `javax.servlet.http.HttpServletRequest` object. Pribavljanje parametra ime `${#request.getParameter('ime')}`, atributa `mojAtribut` `${#request.getAttribute('mojAtribut')}` i atributa zaglavlja `User-Agent` `${#request.getHeader('User-Agent')}`
 - **#response**: (only in Web Contexts) the `javax.servlet.http.HttpServletResponse` object. Pribavljanje njegovih vrednosti iz templejta nema baš puno smisla jer još nije objekat formiran osim ako u kontroleru prethodno ne definišemo neku vrednost `${#response.getHeader('zaglavlje1')}`.
 - **#servletContext**: (only in Web Contexts) the `ServletContext` object. Pribavljanje objekta `statistikaFilmova` `${#servletContext.getAttribute('statistikaFilmova').filmovi}`.
 - **param**: mapping for retrieving request parameters (query ili data parametri). Pribavljanje parametra ime `${param.ime[0]}`. Metode `${param.size()}`, `${param.isEmpty()}`, `${param.containsKey('ime')}`
 - **session**: (only in Web Contexts) mapping for retrieving `HttpSession` object. Pribavljanje objekta `prijavljeniKorisnik` `${session.prijavljeniKorisnik}`.
 - **application**: retrieving servlet context attributes `${application.statistikaFilmova}`

Izrazi – Basic Objects

```
Lokalizacija jezik:<span th:text="${#locale.language}"></span><br/>
Parametar zahteva parametar1:<span th:text="${#request.getParameter('parametar1')}"></span><br/>
Atribut zahteva tekst1:<span th:text="${#request.getAttribute('tekst1')}"></span><br/>
Atribut zahteva korisnik1 (deo korisnickoIme):<span
th:text="${#request.getAttribute('korisnik1').korisnickoIme}"></span><br/>
Atribut zahteva korisnik2 (deo korisnickoIme):<span
th:text="${#request.getAttribute('korisnik2').korisnickoIme}"></span><br/>
Atribut zaglavlja zahteva User-Agent<span th:text="${#request.getHeader('User-Agent')}"></span><br/>
Atribut zaglavlja odgovora zaglavlje1:<span th:text="${#response.getHeader('zaglavlje1')}"></span><br/>
Parametar zahteva parametar1:<span th:if="${param.parametar1} != null"
th:text="${param.parametar1[0]}"></span><br/>
Atribut sesije prijavljeniKorisnik(deo korisnickoIme):<span th:if="${session.prijavljeniKorisnik} != null"
th:text="${session.prijavljeniKorisnik.korisnickoIme}"></span><br/>
```



<http://localhost:8080/Bioskop/Objekti?parametar1=moj+parametar>

Lokalizacija jezik:sr
Parametar zahteva parametar1:moj parametar
Atribut zahteva tekst1:Atribut zahteva tekst1
Atribut zahteva korisnik1 (deo korisnickoIme):pera
Atribut zahteva korisnik2 (deo korisnickoIme):steva
Atribut zaglavlja zahteva User-AgentMozilla/5.0 (Windows NT 6.1; V
Atribut zaglavlja odgovora zaglavlje1:Atribut zaglavlja odgovora
Parametar zahteva parametar1:moj parametar
Atribut sesije prijavljeniKorisnik(deo korisnickoIme):a

Izrazi - Utility Objects

- Pomoćni objekti (**Utility Objects**) su:
 - #execInfo: information about the template being processed.
 - **#messages**: methods for obtaining externalized messages inside variables expressions, in the same way as they would be obtained using #{...} syntax.
 - #uris: methods for escaping parts of URLs/URIs
 - **#conversions**: methods for executing the configured conversion service (if any).
 - **#dates**: methods for java.util.Date objects: formatting, component extraction, etc.
 - **#temporals**: methods for java.time.LocalDate, java.time.LocalDateTime objects.
 - **#calendars**: analogous to #dates, but for java.util.Calendar objects.
 - **#numbers**: methods for formatting numeric objects.
 - **#strings**: methods for String objects: contains, startsWith, prepending/appending, etc.
 - **#objects**: methods for objects in general.
 - **#bools**: methods for boolean evaluation.
 - **#arrays**: methods for arrays.
 - **#lists**: methods for lists.
 - **#sets**: methods for sets.
 - **#maps**: methods for maps.
 - **#aggregates**: methods for creating aggregates on arrays or collections.
 - #ids: methods for dealing with id attributes that might be repeated (for example, as a result of an iteration).

Izrazi- Utility Objects

- izrazi pomoćnih objekata se koriste za **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!

Izrazi- Utility Objects

Poruke #messages za test1: `
`

Poruke #messages za test2 i parametre A, B, C: `
`



Poruke #messages za test1: Ovo je test poruka

Poruke #messages za test2 i parametre A, B, C: Ovo je test poruka koja sadrži vrednosti: A, B i C!

Izrazi- Utility Objects

Dan od trenutnog Date: `
`
Formatiranje trenutnog Date: `
`
Formatiranje trenutnog LocalDate: `
`
Formatiranje trenutnog LocalDateTime: `
`
Kreiranje datuma 2020-12-03: `
`
Formatiranje proizvoljnog Date 2020-12-03: `
`



Dan od trenutnog Date: 3
Formatiranje trenutnog Date: 03-12-2020 01:35
Formatiranje trenutnog LocalDate: 03-12-2020
Formatiranje trenutnog LocalDateTime: 03-12-2020 01:35
Kreiranje datuma 2020-12-03: Thu Dec 03 00:00:00 CET 2020
Formatiranje proizvoljnog Date 2020-12-03: 03-12-2020 00:00

Izrazi- Utility Objects

Formatiranje broja 123.123456 na 2 decimale: `
`

Formatiranje broja 123.123456 na 2 decimale sa .: `<span`

`th:text="${#numbers.formatDecimal(123.123456,0,2,'POINT')}">
`

Formatiranje broja 123.123456 na 2 decimale sa ,: `<span`

`th:text="${#numbers.formatDecimal(123.123456,0,2,'COMMA')}">

`



Formatiranje broja 123.123456 na 2 decimale: 00123,12

Formatiranje broja 123.123456 na 2 decimale sa .: 123.12

Formatiranje broja 123.123456 na 2 decimale sa ,: 123,12

Izrazi- Utility Objects

Provera praznog teksta: `
`

Provera praznog teksta parametar1: `<span`

`th:text="${#strings.isEmpty(#request.getParameter('parametar1'))}">
`

Provera sadržanja reči 'dva' u tekstu 'jedan dva tri': `
`

Zamena sadržanja reči 'dva' sa 'pet' u tekstu 'jedan dva tri': `
`



Provera praznog teksta: false

Provera praznog teksta parametar1: true

Provera sadržanja reči 'dva' u tekstu 'jedan dva tri': true

Zamena sadržanja reči 'dva' sa 'pet' u tekstu 'jedan dva tri': jedan pet tri

Izrazi - *selection expressions*

- slični variable expressions
- koriste se za izračunavanje vrednosti za **prethodno odabrani objekat**
- oblik: **{property}*
- Obekat se mora prethodno odabrati sa **th:object** atributom

```
"<table>\r\n"  
" <tr><th>Id</th><td>" + zanr.getId() + "</td></tr>\r\n"  
" <tr><th>Naziv</th><td>" + zanr.getNaziv() + "</td></tr>\r\n"  
"</table>\r\n"
```

```
<table th:object="${zanr}">  
  <tr><th>Id</th><td th:text="*{id}"></td><tr>  
  <tr><th>Naziv</th><td th:text="*{naziv}"></td><tr>  
</table>
```

Izrazi - *selection expressions*

- slični variable expressions
- koriste se za izračunavanje vrednosti za **prethodno odabrani objekat**
- oblik: **{property}*
- Obekat se mora prethodno odabrati sa **th:object** atributom

```
<table th:object="${zanr}">
  <tr><th>Id</th><td th:text="*{id}"></td><tr>
  <tr><th>Naziv</th><td th:text="*{naziv}"></td><tr>
</table>
```



```
<table>
  <tr><th>Id</th><td>2</td><tr>
  <tr><th>Naziv</th><td>akcija</td><tr>
</table>
```

Izrazi - *link (URL) expressions*

- koriste se za formiranje URL-ova u odnosu na bazični URL
- tipično se koriste za popunjavanje **href** atributa
- oblik: *@{/url_nastavak}*
- Postoje različiti tipovi URL

- Apsolutni URL

http://localhost:8080/Bioskop/Filmovi/Details?id=1

- Relativni URL se dele na *Bio na /Filmovi/index.html*

- Relativni u odnosu na stanicu: *Login.html* → */Filmovi/Login.html*
- Relativni u odnosu na context, gde će se iz contex objekta automatski dodati ime aplikacije: */Filmovi/Details?id=1* → */Bioskop/Filmovi/Details?id=1*
- Relativni u odnosu na server, dozvoljava pozivanje druge aplikacije na istom serveru: *~/DrugaAplikacija/OstatakURL*

Izrazi - *link (URL) expressions*

- Apsolutni URL <http://localhost:8080/Bioskop/Filmovi/Details?id=1>

```
<a th:href="@{http://localhost:8080/Bioskop/Filmovi/Details(id=${itFilm.id})}">Avengers:  
Endgame</a>
```



render

```
<a href="http://localhost:8080/Bioskop/Filmovi/Details?id=1">Avengers: Endgame </a>
```

Thymeleaf

Izrazi - *link (URL) expressions*

- Relativni u odnosu na context, gde će se iz contex objekta automatski dodati ime aplikacije */Filmovi/Details?id=1*
- bazični URL se čita iz *context path*-a aplikacije

`<base th:href="@{/}">`  `<base href="/Bioskop/">`

`<a th:href="@{/Korisnici}">korisnici`  `korisnici`

`<a th:href="@{/Filmovi/Details(id=${itFilm.id})}">Avengers: Endgame`
ili

`<a th:href="@{/Filmovi/Details?id=1}">Avengers: Endgame`

 render

`Avengers: Endgame`

Thymeleaf

Izrazi - *message (i18)* expressions

- nazivaju se još i *text externalization, internationalization* ili *i18n*
- koriste se za pribavljanje **locale specifičnih poruka** iz spoljnog izvora
- izvor može biti .property fajl
- oblik: ***`#{ključ}`***

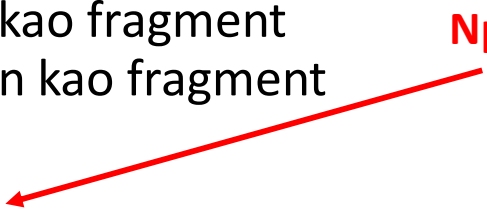
filmovi.film.trajanje=trajanje

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



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


Izrazi - *fragment expressions*

- koriste se za pribavljanje **fragmenata drugih HTML stranica**
- cilj je ponovno iskorišćenje često ponavljanih delova HTML stanice (npr. sekcije *header* i *fotter*)
- **U posmatranu html stranicu ubacujem delove drugih stranica**
- Fragmenti se mogu kreirati u posebnim stranicama ili u jednoj stanici
oblik: `<ElementZaKojiSeFragmentDefinise th:fragment="nazivFragmenta">`
- Postoje 3 osnovna načina za uključivanje sadržaj fragemnata
 - insert – u elementu se insertuje element koji je definisan kao fragment
 - replace – element se menja sa elementom koji je definisan kao fragment
 - include – uključivanje sadržaja - deprecated legacy code
- oblik: `<ElementZaKojiSeFragmentUkljucuje th:insert="~{nazivHTMLstranice :: nazivFragmenta}">`
 **Npr. div**
- <https://www.baeldung.com/spring-thymeleaf-fragments>
- <https://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.html#template-layout>

Thymeleaf

Izrazi - *fragment expressions*

fragmenti.html

```
<header th:insert=~{fragments/general.html :: header}> </header>
<h2>Rad sa fragmentima</h2>
<div th:replace=~{fragments/general.html :: footer}> </div>
```

+

general.html

```
<div th:fragment="header">
    <p>Neki tekst u zaglavlju</p>
</div>
<h2>Opšta stranica</h2>
<footer th:fragment="footer">
    <p>Neki tekst u podnožju</p>
</footer>
```

render

fragmenti.html

```
<header>
    <div>
        <p>Neki tekst u zaglavlju</p>
    </div>
</header>
<h2>Rad sa fragmentima</h2>
<footer>
    <p>Neki tekst u podnožju</p>
</footer>
```

FragmentiController i fragmenti.html

Flexible layouts

- Osnovna ideja je da se napravi fleksibilni prostorni raspored stranica koji nije više samo jednostavno ubacivnje fragmenata.
- Kreirati fragmente tako da oni dobijaju podatke iz templejta koji ih pozivaju kao npr. naslov stanice
- **Delove posmatrane html stranice ubacujem u drugu stranicu u kojoj je definisan prostorni raspored svih stranica**

Thymeleaf

Izrazi - *fragment expressions*

```
raspored.html
<!DOCTYPE html>
<html th:replace="~{fragments/layout.html :: zajednickaStranica(~{::title},~{::centralniDiv})}">

    <title th:text="'Fleksibilni layout'"></title>
    <div th:fragment="centralniDiv">
        <h2>Naslov fleksibilnog layouta</h2>
        <p>Tekst fleksibilnog layouta</p>
    </div>
</html>
```

+

layout.html

RasporedController i raspored.html

Izrazi - *fragment expressions*

```
layout.html
<!DOCTYPE html>
<html th:fragment="zajednickaStranica(title, centralniDiv)">
<head>
    <meta charset="UTF-8">
    <title th:replace="${title}">neki naslov koji se menja</title>
</head>
<body>
<header>
    <p>Neki tekst u zaglavlju</p>
</header>

<div th:replace="${centralniDiv}">centralni deo koji se menja</div>

<footer>
    <p>Neki tekst u podnožju</p>
</footer>
</body>
</html>
```

Izrazi - *fragment expressions*

```
raspored.html
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Fleksibilni layout</title>
</head>
<body>
    <header>
        <p>Neki tekst u zaglavlju</p>
    </header>

    <div>
        <h2>Naslov fleksibilnog layouta</h2>
        <p>Tekst fleksibilnog layouta</p>
    </div>
    <footer>
        <p>Neki tekst u podnožju</p>
    </footer>
</body>
</html>
```

Supstitucija

- važi za bilo koji tekstualni izraz, koristi se za formiranje tekstualnog izraza
- umesto standardnog formiranja teksta `"'neki tekst' + ${varijabla} + 'neki tekst'"`
- umeće izraz se između znakova `|` zajedno sa statičkim tekstom
- nakon popunjavanja šablona (*render*-ovanja), vrednost izraza će se dodati na tekst između znakova `|`
- oblik: `|statički_tekst izraz|`

Umesto standardnog formiranja teksta

```
<p th:text="'Rezultat zbira promenljive a i 3 je:' + ${a+3} + '!'"></p>
```

Koristi se supstitucija

```
<p th:text="|Rezultat zbira promenljive a i 3 je:${a+3}!|"></p>
```

Supstitucija

- važi za bilo koji tekstualni izraz, koristi se za formiranje tekstualnog izraza
- umesto standardnog formiranja teksta `"'neki tekst' + ${varijabla} + 'neki tekst'"`
- umeće izraz se između znakova `|` zajedno sa statičkim tekstom
- nakon popunjavanja šablona (*render*-ovanja), vrednost izraza će se dodati na tekst između znakova `|`
- oblik: `|statički_tekst izraz|`

```
"<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>
```


Definisanje lokalnih promenljivih

- Dozvoljeno je definisanje lokalnih promenljivih koje će biti vidljive samo u elementu u kojem su definisane
- Navedi promenljiva i njena vrednost u atributu `th:with`
- oblik:

```
<element th:with="nazivLokProm=${promenljiva}">  
    <element th:text="${nazivLokProm.atribut1}"> </element>  
</element>
```

Uslovni izraz

- Uslovni izraz je namenjen za izvršavanje jednog od dva ponuđena izraza u zavisnosti od rezultata evaluacije uslova
- Sva tri dela izraza (**condition**, **then** i **else**) su izrazi za sebe

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



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

ili

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

Čitanje parametara iz URL-a

```
<input type="search" name="naziv" th:value="${param.naziv}"/>
```



```
<input type="search" name="naziv" value="er"/>
```

Elvis operator – Predefinisani uslovni izraz

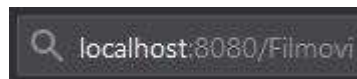
- Predstavlja specijalizaciju uslovnog izraza u kome se ne navodi **then** deo.
- Vrednost **then** dela je predefinisana i predstavlja zapravo vrednost varijable navedene u **condition** delu tj. u uslovu izraza.

```
<input type="search" name="naziv" th:value="${param.naziv}?: ''"/>
```



```
<input type="search" name="naziv" value="er"/>
```

ili



```
<input type="search" name="naziv" value=""/>
```

Uslovni prikaz HTML elementa

- ako se u elementu navede atribut **th:if**, **element** zajedno sa svojim podelementima se **prikazuje** ako je logička vrednost izraza tačna
- oblik: `<element th:if="{uslov}">...</element>`
- ako se u elementu navede atribut **th:unless**, **element** zajedno sa svojim podelementima se **prikazuje** ako je logička vrednost izraza netačna
- oblik: `<element th:unless="{uslov}">...</element>`

Uslovni prikaz HTML elementa

```
if (prijavljeniKorisnik != null) {  
    out.append(  
        "<table class=\"korisnik\">\r\n" +  
        ...  
        "</table>\r\n"  
    );  
} else {  
    out.append(  
        "<table class=\"korisnik\">\r\n" +  
        ...  
        "</table>\r\n"  
    );  
}
```

```
<table class="korisnik" th:if="${prijavljeniKorisnik != null}">  
    ...  
</table>  
  
<table class="korisnik" th:unless="${prijavljeniKorisnik != null}">  
    ...  
</table>
```

uslov

uslov

Uslovni prikaz HTML atributa

- atribut će se prikazati ako je logička vrednost izraza tačna

uslov

```
if (film.getZanrovi().contains(itZanr)) {  
    out.append("<input type='checkbox' checked/><span>" + itZanr.getNaziv() + "</span><br/>\r\n");  
} else {  
    out.append("<input type='checkbox' /><span>" + itZanr.getNaziv() + "</span><br/>\r\n");  
}
```

↑

atribut

```
<input type="checkbox" th:checked="${#lists.contains(film.zanrovi, itZanr)}"/><span th:text="${itZanr.naziv}"></span><br>
```

uslov

Uslovni prikaz HTML elementa

- ako se u nadelementu navede atribut **th:switch** i u elementu se navede atribut **th:case**, tada **element** zajedno sa svojim podelementima se prikazuje ako vrednost elementa odgovara vrednosti promenljive definisane u nadelementu
- Odgovara sintaksi **switch** strukture u Javi
- oblik:

```
<element th:switch="${promenljiva}">  
    <element th:case="vrednost1"> </element>  
    <element th:case="${promenljiva2}"> </element>  
</element>
```


Uslovni prikaz HTML elementa

```
<div th:switch="${prijavljeniKorisnik.uloga}">  
  <p th:case="'admin'">Korisnik je administrator</p>  
  <p th:case="#{roles.manager}">Korisnik je menadžer</p>  
</div>
```

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 pređenih elemenata kolekcije, počevši od 1
 - **size:** ukupan broj elemenata kolekcije
 - **current:** predstavlja vrednost element promenljive
 - **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

```
<table class="tabela">
  <caption>Žanrovi</caption>
  <tr><th>r. br.</th><th>naziv</th><th></th></tr>
  <tr th:each="itZanr, status: ${zanrovi}">
    <td th:text="${status.index + 1}"></td>
    <td><a th:href="/Zanrovi/Details?id=${itZanr.id}" th:text=${itZanr.naziv}></a></td>
    <td><a th:href="/Filmovi?zanrId=${itZanr.id}">filmovi</a></td>
  </tr>
</table>
```



```
<table class="tabela">
  <caption>Žanrovi</caption>
  <tr><th>r. br.</th><th>naziv</th><th></th></tr>
  <tr>
    <td>1</td>
    <td><a href="/Zanrovi/Details?id=1">naučna fantastika</a></td>
    <td><a href="/Filmovi?zanrId=1">filmovi</a></td>
  </tr>
  <tr>
    <td>2</td>
    <td><a href="/Zanrovi/Details?id=2">akcija</a></td>
    <td><a href="/Filmovi?zanrId=2">filmovi</a></td>
  </tr>
  <tr>
    <td>3</td>
    <td><a href="/Zanrovi/Details?id=3">komedija</a></td>
    <td><a href="/Filmovi?zanrId=3">filmovi</a></td>
  </tr>
</table>
```

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

Blokovi

- ako više elemenata treba da se uslovno prikažu ili ponavljaju tada `th:if`, `th:unless` ili `th:each` atributi se mogu dodeliti njihovom zajedničkom nadelementu (`table`, `tr`, `ul`, i sl.) ako struktura HTML dokumenta to dozvoljava
- ako elementi nemaju pogodan zajednički nadelement, oni se tada teoretski mogu grupisati unutar `div` ili `span` elemenata
- ako bi grupisanje unutar `div` ili `span` elemenata dovelo do nepravilno formiranog HTML dokumenta, elementi koji se grupišu se mogu navesti unutar posebnog (nestandardnog) `th:block` elementa

```
<th:block th:if="${...}">
```

```
...
```

```
</th:block>
```

```
<th:block th:unless="${...}">
```

```
...
```

```
</th:block>
```

```
<th:block th:each="${...}">
```

```
...
```

```
</th:block>
```

Blokovi

- samo **obeleženi** elementi treba da se ponavljaju ali tako da ostanu unutar jednog **td** elementa
- ne smeju da se gupišu u **div** element jer bi se on tada nalazio unutar **form** elementa, a to onda ne bi bilo ispravno po HTML standardu

```
<form method="post" action="Filmovi/Edit" th:if="${session.prijavljeniKorisnik != null and session.prijavljeniKorisnik.administrator}">
  <input type="hidden" name="id" th:value="${film.id}"/>
  <table class="forma">
    <caption>Film</caption>
    <tr><th>naziv:</th><td><input type="text" th:value="${film.naziv}" name="naziv"/></td></tr>
    <tr>
      <th>žanr:</th>
      <td>
        <th:block th:each="itZanr: ${zanrovi}">
          <input type="checkbox" name="zanrId" th:value="${itZanr.id}" th:checked="${#lists.contains(film.zanrovi, itZanr)}"/>
          <span th:text="${itZanr.naziv}"></span><br>
        </th:block>
      </td>
    </tr>
    <tr><th>trajanje:</th><td><input type="number" min="5" th:value="${film.trajanje}" name="trajanje"/></td></tr>
    <tr><th></th><td><a th:href="/Projekcije?filmId=${film.id}|">projekcije</a></td></tr>
    <tr><th></th><td><input type="submit" value="Izmeni"/></td></tr>
  </table>
</form>
```

Case study – CRUD bioskop veb aplikacija

- USE CASE korišćenje Thymeleaf za Bioskop web aplikaciju
- *com.ftn.PrviMavenVebProjekat:*
 - *IndexController.java, index.html*
 - *FilmoviController.java, filmovi.html, dodavanjeFilma.html, film.html*

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!

Case study – CRUD bioskop veb aplikacija

Vežbanje posle predavanja

- Pokušajte da kopletnu aplikaciju prebacite tako da koristi Thymeleaf. Implementirati generisanje dinamičkog HTML sadržaja uz pomoć *Thymeleaf*-a.
- Iz svih kontrolera ukoniti kod kojim oni vraćaju html.
- Definirati neophodne HTML templejte. HTML templejti moraju da podrže internacionalizaciju sadržaja, potrebo je koristiti fleksibilni prostori raspored stanica.

Dodatni materijali

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

- Eclipse plugin Thymeleaf Plugin for Eclipse
<https://marketplace.eclipse.org/content/thymeleaf-plugin-eclipse>
- Dodati u html `<html xmlns:th="http://www.thymeleaf.org">`
- *Primer u objekti.html*

```
Dan od trenutnog Date: <span th:text="${#dates.date(standardDate)}"></span><br/>
Formatiranje trenutnog Date: <span th:text="${#dates.format(standardDate, 'dd/MM/yyyy')}"></span>
Formatiranje trenutnog LocalDate: <span th:text="${#dates.format(localDate, 'dd/MM/yyyy')}"></span>
Formatiranje trenutnog LocalDateTime: <span th:text="${#dates.format(localDateTime, 'dd/MM/yyyy HH:mm:ss')}"></span>
Kreiranje datuma 2020-12-03: <span th:text="${#dates.parse('2020-12-03')}"></span>
Formatiranje proizvoljnog Date 2020-12-03: <span th:text="${#dates.format(date, 'dd/MM/yyyy')}"></span>
```

- # dates.day
- # dates.dayOfWeek
- # dates.dayOfWeekName
- # dates.dayOfWeekNameShort