



# Spring fundamentals Taken

Deze cursus is eigendom van de VDAB©

## Inhoudsopgave

<b>1</b>	<b>TAKEN .....</b>	<b>7</b>
1.1	Seizoenen .....	7
1.2	GET of POST .....	7
1.3	Project .....	7
1.4	Welkom pagina.....	7
1.5	Controller .....	7
1.6	Thymeleaf.....	7
1.7	Adres .....	7
1.8	Sauzen .....	8
1.9	URL .....	8
1.10	Fragments.....	8
1.11	URL's en methods.....	8
1.12	Path variabele.....	9
1.13	@RequestMapping.....	9
1.14	Taal .....	9
1.15	Bezocht.....	9
1.16	Alfabet .....	9
1.17	Sauzen.csv .....	9
1.18	Sauzen.properties .....	10
1.19	Application.properties.....	10
1.20	Database toegang.....	10
1.21	DataSource .....	10
1.22	Repository .....	10
1.23	Repository test .....	10
1.24	Services.....	10
1.25	Dagverkopen .....	11

1.26	Form .....	11
1.27	Bean validation .....	11
1.28	Client sided validatie .....	11
1.29	Snack wijzigen .....	11
1.30	Zoek de friet .....	12
1.30.1	Tip .....	12
1.31	Raud de saus.....	13
1.32	Gastenboek .....	13
1.33	Gastenboekbeheer .....	14
1.34	Star Trek .....	15
1.34.1	Database .....	15
1.34.2	Welkom pagina .....	15
1.34.3	Werknemer pagina .....	15
1.34.4	Bestellingen pagina .....	16
1.34.5	Nieuwe bestelling pagina .....	16
<b>2</b>	<b>VOORBEELDOPLOSSINGEN.....</b>	<b>17</b>
2.1	GET of POST .....	17
2.2	Project .....	17
2.3	Welkom pagina.....	17
2.3.1	index.html .....	17
2.3.2	frida.css .....	17
2.4	Controller .....	17
2.4.1	IndexController .....	17
2.5	Thymeleaf.....	18
2.5.1	IndexController .....	18
2.5.2	index.html .....	18
2.6	Adres .....	18
2.6.1	Gemeente .....	18
2.6.2	Adres .....	18
2.6.3	IndexController .....	18
2.6.4	index.html .....	18
2.7	Sauzen .....	19
2.7.1	Saus .....	19
2.7.2	SausController.....	19
2.7.3	sauzen.html.....	19
2.8	URL .....	19
2.8.1	index.html en sauzen.html.....	19

2.9	Fragments.....	20
2.9.1	fragments.html .....	20
2.9.2	index.html .....	20
2.9.3	sauzen.html.....	20
2.9.4	frida.css .....	20
2.10	URL's en methods.....	20
2.11	Path variabele.....	21
2.11.1	sauzen.html.....	21
2.11.2	SausController.....	21
2.11.3	saus.html.....	21
2.11.4	frida.css .....	21
2.12	@RequestMapping.....	22
2.13	Taal .....	22
2.13.1	TaalController .....	22
2.13.2	talen.html.....	22
2.13.3	fragments.html .....	22
2.14	Bezocht.....	22
2.14.1	IndexController .....	22
2.14.2	index.html .....	22
2.15	Alfabet .....	23
2.15.1	SausController.....	23
2.15.2	sausAlfabet.html .....	23
2.15.3	frida.css .....	23
2.15.4	fragments.html .....	23
2.16	Sauzen.csv .....	24
2.16.1	SausRepository.....	24
2.16.2	SausRepositoryException .....	24
2.16.3	CSVSausRepository .....	24
2.16.4	CSVSausRepositoryTest.....	25
2.16.5	SausService .....	25
2.16.6	SausController.....	25
2.17	Sauzen.properties .....	26
2.17.1	SausRepository.....	26
2.17.2	CSVSausRepository .....	26
2.17.3	PropertiesSausRepository .....	26
2.17.4	PropertiesSausRepositoryTest .....	27
2.17.5	SausService .....	27
2.18	Application.properties.....	27
2.18.1	application.properties.....	27
2.18.2	CSVSausRepository .....	27
2.18.3	CSVSausRepositoryTest.....	27
2.18.4	PropertiesSausRepository .....	28

2.18.5	PropertiesSausRepositoryTest .....	28
2.19	Database toegang.....	29
2.20	DataSource .....	29
2.20.1	pom.xml .....	29
2.20.2	application.properties.....	29
2.20.3	DataSourceTest .....	29
2.21	Repository .....	29
2.21.1	Snack .....	29
2.21.2	SnackNietGevondenException .....	29
2.21.3	SnackRepository .....	29
2.22	Repository test .....	30
2.22.1	application.properties.....	30
2.22.2	spring.properties.....	30
2.22.3	insertSnacks.sql.....	30
2.22.4	SnackRepositoryTest .....	30
2.22.5	application.properties.....	31
2.23	Services.....	31
2.23.1	application.properties.....	31
2.23.2	SnackService.....	31
2.23.3	SnackController .....	32
2.23.4	snackAlfabet.html .....	32
2.23.5	fragments.html .....	32
2.24	Dagverkopen .....	32
2.24.1	VerkochtAantalPerSnack.....	32
2.24.2	SnackRepository.....	33
2.24.3	insertDagverkopen.sql .....	33
2.24.4	SnackRepositoryTest .....	33
2.24.5	SnackService.....	33
2.24.6	SnackController .....	33
2.24.7	verkochteaantallenpersnack.html .....	34
2.24.8	fragments.html .....	34
2.25	Form .....	34
2.25.1	BeginNaamForm .....	34
2.25.2	SnackController .....	34
2.25.3	beginNaam.html .....	34
2.25.4	fragments.html .....	35
2.26	Bean Validation .....	35
2.26.1	BeginNaamForm .....	35
2.26.2	SnackController .....	35
2.26.3	messages.properties .....	35
2.26.4	beginNaam.html .....	35
2.26.5	frida.css .....	35

2.27	Client sided validatie .....	35
2.27.1	beginNaam.html .....	35
2.28	Snack wijzigen .....	36
2.28.1	snackAlfabet.html en beginNaam.html .....	36
2.28.2	Snack .....	36
2.28.3	SnackController .....	36
2.28.4	wijzigSnack.html .....	36
2.28.5	preventDoubleSubmit.js .....	37
2.28.6	messages.properties .....	37
2.28.7	Index.html .....	37
2.29	Zoek de friet .....	37
2.29.1	Deur .....	37
2.29.2	ZoekDeFriet .....	37
2.29.3	FrietController .....	38
2.29.4	zoekDeFriet.html .....	38
2.29.5	fragments.html .....	38
2.29.6	application.properties .....	38
2.30	Raad de saus .....	39
2.30.1	SausRaden .....	39
2.30.2	SausRadenTest .....	39
2.30.3	SausRadenForm .....	40
2.30.4	SausController .....	40
2.30.5	sausRaden.html .....	41
2.30.6	fragments.html .....	41
2.31	Gastenboek .....	42
2.31.1	Nieuwe table in de database .....	42
2.31.2	Rechten in de database .....	42
2.31.3	GastenBoekEntry .....	42
2.31.4	GastenBoekRepository .....	42
2.31.5	insertGastenBoek.sql .....	43
2.31.6	GastenBoekRepositoryTest .....	43
2.31.7	GastenBoekService .....	43
2.31.8	GastenBoekEntryForm .....	43
2.31.9	GastenBoekController .....	44
2.31.10	gastenboek.html .....	44
2.31.11	fragments.html .....	44
2.32	Gastenboekbeheer .....	45
2.32.1	Rechten in database .....	45
2.32.2	GastenBoekRepository .....	45
2.32.3	GastenBoekRepositoryTest .....	45
2.32.4	GastenBoekService .....	45
2.32.5	GastenBoekController .....	45
2.32.6	gastenboek.html .....	46
2.32.7	frida.css .....	46

2.33	Star Trek .....	46
2.33.1	application.properties.....	46
2.33.2	spring.properties.....	46
2.33.3	DataSourceTest.....	46
2.33.4	Werknemer .....	46
2.33.5	Bestelling.....	47
2.33.6	OnvoldoendeBudgetException .....	47
2.33.7	WerknemerNietGevondenException .....	47
2.33.8	WerknemerRepository.....	47
2.33.9	insertWerknemers.sql.....	48
2.33.10	WerknemerRepositoryTest .....	48
2.33.11	BestellingRepository .....	49
2.33.12	insertBestellingen.sql.....	50
2.33.13	BestellingRepositoryTest .....	50
2.33.14	tWerknemerService .....	50
2.33.15	tBestellingService.....	51
2.33.16	BestellingServiceTest .....	51
2.33.17	IndexController .....	52
2.33.18	WerknemerController .....	52
2.33.19	fragments.html .....	53
2.33.20	index.html .....	53
2.33.21	werknemer.html .....	53
2.33.22	vorigeBestellingen.html .....	54
2.33.23	nieuweBestelling.html .....	54
2.33.24	messages.properties .....	55
2.33.25	startrek.css .....	55
<b>3</b>	<b>COLOFON.....</b>	<b>57</b>

## 1 TAKEN

### 1.1 Seizoenen

Je vindt seizoenen.war in het takenmateriaal. Installeer de website op je Tomcat webserver. Surf daarna naar de website.

### 1.2 GET of POST

Programmeer je volgende website onderdelen als een GET request of als een POST request ?

1. een request geeft een werknemer opslag.
2. een request toont de lonen van de werknemers.
3. Een request geeft een bonus aan alle werknemers.

### 1.3 Project

Maak een project voor de website voor Frituur Frida.

Gebruik de dependencies DevTools, Web, Thymeleaf en Validation.

### 1.4 Welkom pagina

Maak een welkom pagina.

- De pagina bevat een `<h1>` element met de tekst Frituur Frida.
- Geef de tekst een kleur met een CSS bestand.
- Gebruik frituur.ico uit het takenmateriaal.

### 1.5 Controller

Frituur Frida is gesloten op maandag.

Voeg een controller toe aan de website die requests verwerkt naar de welkom pagina.

Als je naar de website surft op maandag toon je de tekst gesloten.

Toon op andere dagen de tekst open.

### 1.6 Thymeleaf

Frituur Frida is gesloten op maandag.

Verwerk een request naar de welkom pagina in een controller.

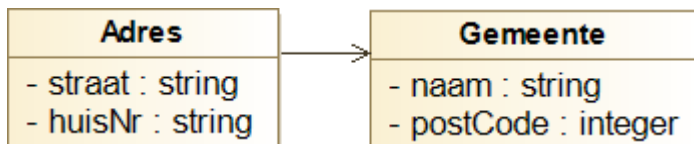
Die maakt een String met gesloten (op maandag) of open (op andere dagen).

De controller geeft de String door aan een Thymeleaf pagina.

Die toont met die data de tekst `We zijn vandaag gesloten.` of `We zijn vandaag open.`

### 1.7 Adres

Maak volgende immutable classes:



In de controller die hoort bij de welkom pagina

- maak je een Adres object en een bijbehorend Gemeente object
- vul je de attributen van die objecten met zelf verzonden adresgegevens van frituur Frida
- geef je dit adres object door aan de Thymeleaf pagina.

Toon in de Thymeleaf pagina het volledige adres van frituur Frida aan de hand van die data.



## 1.8 Sauzen

Maak een immutable class Saus:

Saus
<ul style="list-style-type: none"> <li>- nummer : long</li> <li>- naam : string</li> <li>- ingrediënten : string [*]</li> </ul>

Stel de ingrediënten voor als een String array.

Voeg een controller toe. Hij verwerkt requests naar de URL /sauzen

Doe volgende stappen bij een GET request naar /sauzen:

- Maak een array van Saus objecten. Vul de array met sauzen met volgende namen: cocktail, mayonaise, mosterd, tartare, vinaigrette. Verzin zelf id's en ingrediënten van de sauzen.
- Geef de array door aan een Thymeleaf pagina.
- Toon in die pagina de naam en de ingrediënten van de sauzen.

## 1.9 URL

Gebruik waar nodig URL expressions.

## 1.10 Fragments

Maak in elke pagina een menu (zoals in de theorie).

Vervang in elke pagina <head>...</head> door de oproep van een Thymeleaf fragment.

## 1.11 URL's en methods

Gegeven volgende fictieve controller:

```
@Controller
class LandController {
    @GetMapping
    public ModelAndView landen() {
        ...
    }
    @GetMapping("/landen/rijkste10")
    public ModelAndView rijkste10() {
        ...
    }
    @GetMapping("/landen/{code}")
    public ModelAndView land(@PathVariable String code) {
        ...
    }
    @GetMapping("/landen/werelddelen/{werelddeel}")
    public ModelAndView vanWerelddeel(@PathVariable String werelddeel) {
        ...
    }
}
```

Welke methods worden opgeroepen bij GET requests naar volgende URL's ?

- /landen/BE
- /landen
- /landen/NL
- /landen/rijkste10
- /landen/werelddelen/europa

### 1.12 Path variabele

Maak, in de pagina met de sauzen, van elke saus een hyperlink.

Als de gebruiker die aanklikt toon je een nieuwe pagina met de detail van die saus.

Toon in die pagina ook de afbeelding van die saus.

Je vindt de afbeeldingen bij het takenmateriaal.

Tip 1: het `src` attribuut van het `img` element bevat een URL.

Je kan het met `th:src` invullen met een URL expression.

Tip 2: je kan het `alt` attribuut van het `img` element met `th:alt` invullen met een variable expression.

### 1.13 @RequestMapping

Gebruik `@RequestMapping` in `SausController`.




### 1.14 Taal

De browser stuurt bij elke request een header `accept-language` mee.

Die bevat de taal en eventueel het land van de gebruiker.

Als de header begint met `nl` betekent dit dat de gebruiker Nederlands spreekt.

De gebruiker bepaalt de inhoud van de header in de browser instellingen. Bij Chrome:

1. Kies rechts boven .
2. Kies `Settings`.
3. Kies links `Advanced` en daarbinnen `Languages`.
4. Kies in het midden  naast `Language`.
5. Je kan een taal of een taal-land combinatie toevoegen met `Add Languages`.
6. Je kan met  naast een taal de volgorde van de talen instellen.

Toon een pagina met de tekst `Je spreekt Nederlands` of `Je spreekt geen Nederlands`.

Baseer je daarbij op de header `accept-language`.

### 1.15 Bezoekt

Je toont in de welkom pagina de hoeveelste keer de gebruiker de website bezoekt.

### 1.16 Alfabet

Maak een nieuwe pagina. Toon de letters van het alfabet(in kleine letters) als hyperlinks.

Als de gebruiker op zo'n letter klikt, toon je ook de sauzen waarvan de naam begint met die letter.

### 1.17 Sauzen.csv

Het takenmateriaal bevat een bestand `sauzen.csv`.

CSV is een afkorting van `comma separated values`.

Het duidt een tekstbestand aan waarbij de data op een regel gescheiden worden door een komma.

Maak een directory data in de root van de hard disk. Kopieer het bestand in die directory.

Maak een package `be.vdab.frida.repositories`.

Maak daarin een class `CSVSausRepository` met een method `List<Saus> findAll()`.

Maak van de class een Spring bean.

Lees in de method `findAll` `sauzen.csv`. Maak op basis daarvan een `List<Saus>`.

Maak een package `be.vdab.frida.services`.

Maak daarin een class `SausService` met volgende methods:

- `List<Saus> findAll()`
- `List<Saus> findByBeginNaam(char letter)`
- `Optional<Saus> findById(long id)`

Maak van de class een Spring bean. Injecteer de `CSVSausRepository` bean.

Roep in alle methods de repository op. Injecteer de `SausService` bean in de `SausController`.

Roep in de `@GetMapping` methods de methods van deze bean op.

### 1.18 Sauzen.properties

Het takenmateriaal bevat een bestand `sauzen.properties`:

```
1:cocktail,mayonaise, ketchup,cognac
2:mayonaise,ei,mosterd
3:mosterd,mosterd,azijn,witte wijn
4:tartare,mayonaise, augurk
5:vinaigrette,olijfolie,mosterd,azijn
```

Kopieer dit bestand in de directory `data` in de root van de harde schijf.

Maak in de package `be.vdab.frida.repositories` een class `PropertiesSausRepository`.

De class implementeert ook de interface `SausRepository`.

Maak van de class een Spring bean.

Gebruik `PropertiesSausRepository` als dependency in `SausService`.

### 1.19 Application.properties

Neem het pad van `sauzen.csv` op in `application.properties`:

```
CSVSausenPad=/data/sauzen.csv
```

Neem het pad van `sauzen.properties` op in `application.properties`:

```
propertiesSausenPad=/data/sauzen.properties
```

Gebruik deze instellingen in `CSVSausRepository` en `PropertiesSausRepository`.

### 1.20 Database toegang

Bij de website hoort een database `frida`.

Voer het script `frida.sql` uit (bij het takenmateriaal).

Dit script maakt een database `frida` met een table `snacks`.

Het script geeft aan de gebruiker cursist de nodige rechten.

Maak in IntelliJ een verbinding met de database `Frida`.

snacks	
id	INT
naam	VARCHAR(50)
prijs	DECIMAL(10,2)

### 1.21 DataSource

Maak een test voor de `DataSource`.

### 1.22 Repository

Maak in de package `be.vdab.frida.domain` een class `Snack`.

Maak in de package `be.vdab.frida.repositories`

een class `SnackRepository` met 3 methods:

- `Optional<Snack> findById(long id)`
- `void update(Snack snack)`
- `List<Snack> findByBeginNaam(String beginNaam)`

### 1.23 Repository test

Maak een test voor de class `SnackRepository`.

### 1.24 Services

Maak in de package `be.vdab.frida.services` een class `SnackService` met 3 methods:

- `Optional<Snack> read(long id)`
- `void update(Snack snack)`
- `List<Snack> findByBeginNaam(String beginNaam)`

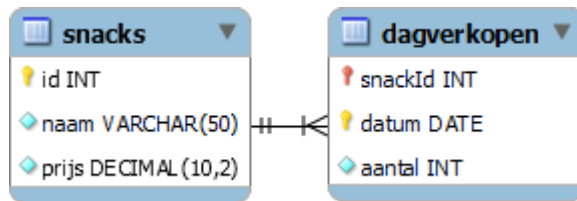
Maak een `SnackController` en een bijbehorende Thymeleaf pagina.

Toon in die pagina de letters van het alfabet. Iedere letter is een hyperlink. Als de gebruiker zo'n hyperlink aanklikt, toon je terug de pagina. Die bevat opnieuw het alfabet.

Toon daaronder de namen van de snacks waarvan de naam begint met de aangeklikte letter.

### 1.25 Dagverkopen

Voer het script dagverkopen.sql uit. Dit voegt een table dagVerkopen toe.



Het script vult de table met enkele records.

Je maakt een pagina. De gebruiker ziet een tabel met per snack één rij met drie kolommen:

1. Kolom 1: id van de snack.  
Je sorteerd de snacks op id.
2. Kolom 2: naam van de snack.
3. Kolom 3: totaal aantal verkocht van die snack.  
Je toont ook de snacks waarvoor nog niet verkocht werd. Je toont dan 0 in deze kolom.

### 1.26 Form

Toon een form aan de gebruiker. De form bevat één invoervak.

De gebruiker typt in dit invoervak de beginletters van een snack. Hij klikt daarna op een knop Zoeken. Toon dan de namen van de snacks waarvan de naam begint met de getypte beginletters.

### 1.27 Bean validation

Pas validatie toe in de form van de vorige oefening. Het invoervak is verplicht in te vullen.

### 1.28 Client sided validatie

Pas client sided validatie toe in de form van de vorige oefening.

### 1.29 Snack wijzigen

Maak hyperlinks van de namen van snacks

- in de pagina met het alfabet
- in de pagina met het invoervak met beginletters

Als de gebruiker zo'n hyperlink aanklikt,

komt hij op een pagina waar hij de naam en de prijs van de snack kan wijzigen.

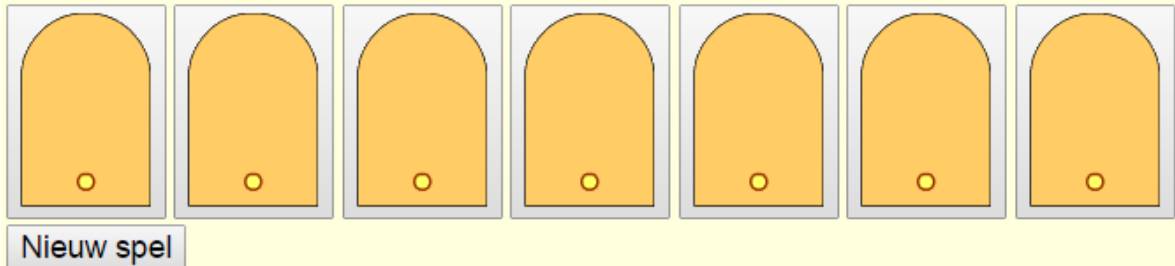
- de naam mag niet leeg zijn.
- de prijs mag niet leeg zijn en moet minstens 0 zijn.

Nadat de gebruiker correcte waarden typt en op de knop Opslaan klikt, toon je de welkom pagina.

### 1.30 Zoek de friet

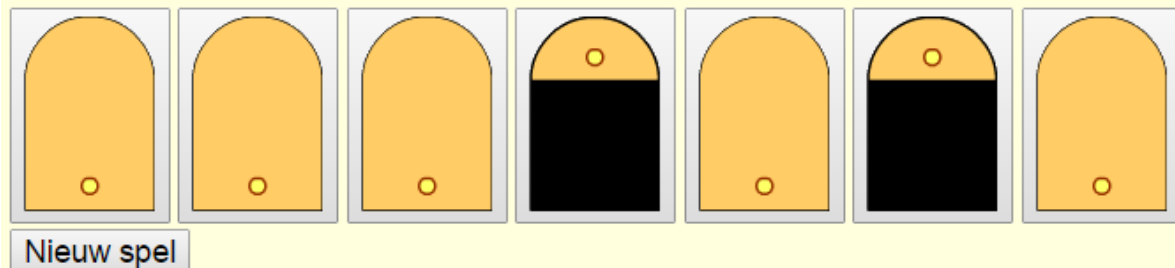
De gebruiker ziet in een pagina zeven gesloten deuren.

## Zoek de friet

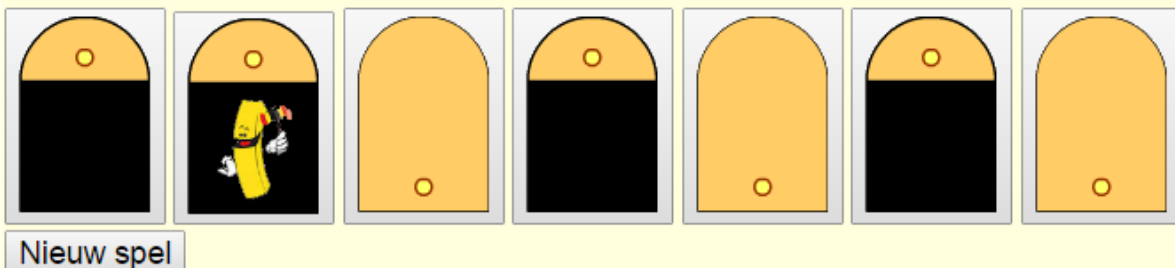


Achter één van de deuren bevindt zich een friet. De gebruiker zoekt die deur. De gebruiker kan de deuren één per één openen door op een deur te klikken. Eenmaal geopend blijft de deur open.

## Zoek de friet



## Zoek de friet



Als de gebruiker Nieuw spel kiest, maak je een nieuw spel met zeven gesloten deuren.

Gebruik de afbeeldingen deuropen.png, deurtoe.png en gevonden.png uit het takenmateriaal.

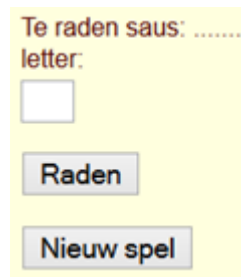
#### 1.30.1 Tip

Maak met volgende HTML code een afbeelding die een request parameter met de naam index en de waarde 3 doorgeeft bij het submitten van de form die de afbeelding bevat:

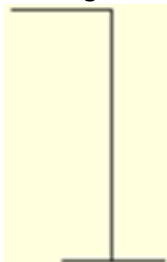
```
<button name="index" value="3">  
    
</button>
```

### 1.31 Raud de saus

Maak een nieuwe pagina waarin de gebruiker de naam van één van de sauzen van frituur Frida raadt.



- Toon naast Te raden saus: evenveel puntjes als de te raden saus. Kies de saus willekeurig uit de sauzen in de database.
- De gebruiker kan naast letter: een letter typen en op Raden klikken. De gebruiker ziet daarna dezelfde pagina.
  - Als de getypte letter voorkomt in de te raden saus, toon je op de plaats van die letter geen puntje, maar de letter. Als de gebruiker bijvoorbeeld *t* typte en de te raden saus is *cocktail*, toon je Te raden saus: . . . t . . .
  - Als de getypte letter niet voorkomt in de te raden saus, verhoog je een teller met het aantal verkeerde pogingen. Toon de teller aan de gebruiker via één van de afbeeldingen 0.png . . . 9.png bij de cursus. Als de gebruiker bijvoorbeeld 3 verkeerde pogingen deed, toon je 3.png:



- Als de gebruiker 10 verkeerde pogingen deed, verliest hij het spel. Toon dan de tekst U bent verloren, de saus was cocktail
- Als de gebruiker de saus raadt en nog geen 10 verkeerde pogingen deed, wint hij het spel. Toon dan de tekst U bent gewonnen, de saus was cocktail
- De gebruiker kan het spel herbeginnen met de knop Nieuw spel. Kies dan terug een willekeurige saus voor dit nieuwe spel.

### 1.32 Gastenboek

Voeg aan de website voor frituur Frida een gastenboek toe.

Als de gebruiker de pagina opent, ziet hij de berichten die gebruikers al toevoegden, in omgekeerde chronologische volgorde:



Voeg zelf een table toe aan de database om de gastenboek data bij te houden.

Als de gebruiker Toevoegen kiest, ziet hij de pagina opnieuw, maar kan hij een eigen bericht toevoegen. Hierbij zijn beide vakken verplicht in te vullen.

**Gastenboek**

Naam:

Bericht:

**Toevoegen**

- **28-8-19 hendrik** Frida's frikandellen, mum !
- **4-1-19 alexandra** Frida is the best.
- **1-8-18 hans** Leve Frida !

### 1.33 Gastenboekbeheer

De gebruiker ziet in het gastenboek bij ieder bericht een checkbox.

De gebruiker ziet ook een knop Verwijderen.

Als hij de knop aanklikt, verwijder jij alle aangevinkte berichten.

Tip: je maakt met volgend code fragment de form met checkboxen in de Thymeleaf pagina:

```
<form th:if="not ${gastenboek.empty}" th:action="@{/gastenboek/verwijderen}"
      method="post">
  <ul>
    <li th:each="entry : ${gastenboek}" th:object="${entry}">
      <strong th:text="|*{{datum}} *{{naam}}|"></strong>
      <th:block th:text="*{{bericht}}"></th:block>
      <input type="checkbox" name="id" th:value="*{{id}}">
    </li>
  </ul>
  <button>Verwijderen</button>
</form>
```

Verwerk de request van het submitten in volgende controller method:

```
@PostMapping("verwijderen")
public String delete(long[] id) {
  ...
}
```

De parameter id bevat null als geen enkel vinkje aangeklikt is.

Anders bevat hij de ids van de aangevinkte berichten.

### 1.34 Star Trek

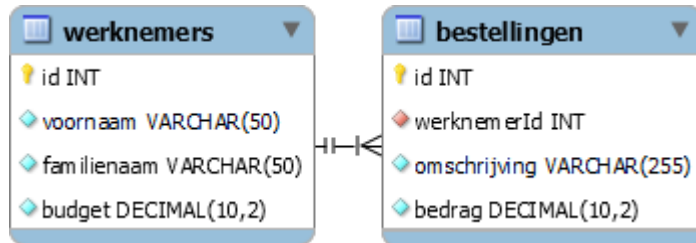
Je maakt in een **nieuw** project een website.

Bedienden typen de bestellingen van werknemers van de firma “Star Trek”.

#### 1.34.1 Database

Je maakt de database met het script `startrek.sql`.

De database bevat volgende tables:



Je website verbindt met de database met de gebruikersnaam `cursist` en het paswoord `cursist`.

#### 1.34.2 Welkom pagina

## Werknemers

- [Christine Chapel](#)
- [Hikaru Sulu](#)
- [James Kirk](#)
- [Janice Rand](#)
- [Montgomery Scott](#)
- [Nyota Uhura](#)
- [Pavel Chekov](#)
- [Tonia Barrows](#)

Je toont alle werknemers. Je sorteert op voornaam.

Als de gebruiker een werknemer aanklikt, toon je de werknemer pagina.

#### 1.34.3 Werknemer pagina

## Christine Chapel



Nummer  
**7**

Budget  
**1,000.00**

[Vorige bestellingen](#) [Nieuwe bestelling](#) [Startpagina](#)

Je vindt de foto's bij het takenmateriaal.

Als de gebruiker op [Vorige bestellingen](#) klikt, toon je de de Bestellingen pagina.

Als hij op [Nieuwe bestelling](#) klikt, toon je de Nieuwe bestelling pagina.



## 1.34.4 Bestellingen pagina

## Bestellingen van Christine Chapel

Nummer	Omschrijving	Bedrag
1	schietstoel	999.99
2	ruimtepak	500.00
3	tube astronautenvoedsel	17.50

[Startpagina](#)

Je sorteert de bestellingen op nummer.

## 1.34.5 Nieuwe bestelling pagina

## Bestelling voor Christine Chapel

Omschrijving

Bedrag

**Bestel**[Startpagina](#)

De omschrijving is verplicht in te vullen.

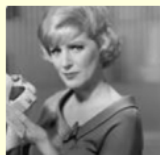
Het bedrag is verplicht in te vullen met een positief getal.

Als de gebruiker **Bestel** kiest, zijn er twee mogelijkheden:

1. Het budget van de werknemer is  $\geq$  bedrag.  
Je vermindert het budget met het bedrag.  
Je voegt een record toe aan de table bestellingen.  
Je toont de Bestellingen pagina van die werknemer.  
De gebruiker ziet op die pagina ook de nieuw toegevoegd bestelling.
2. Het budget van de werknemer is  $<$  bedrag.  
Je toont de Werknemer pagina met een foutmelding:

**Onvoldoende budget.**

## Christine Chapel



Nummer

7

Budget

1,000.00

[Vorige bestellingen](#) [Nieuwe bestelling](#) [Startpagina](#)

Hou er rekening mee dat *meerdere* bedienden tegelijk bestellingen typen en dat *meerdere* bedienden tegelijk bestellingen van *eenzelfde werknemer* kunnen typen.

## 2 VOORBEELDOPLOSSINGEN

### 2.1 GET of POST

een request geeft een werknemer opslag.	POST
een request toont de lonen van de werknemers.	GET
een request geeft een bonus aan alle werknemers.	POST

### 2.2 Project

### 2.3 Welkom pagina

#### 2.3.1 index.html

```
<!doctype html>
<html lang="nl">
  <head>
    <title>Frituur Frida</title>
    <link rel="icon" href="images/frida.ico" type="image/x-icon">
    <link rel="stylesheet" href="css/frida.css">
  </head>
  <body>
    <h1>Frituur Frida</h1>
  </body>
</html>
```

#### 2.3.2 frida.css

```
body {
  background-color: #C0BD92;
  font-family: sans-serif;
}
h1 {
  color: red;
}
```

### 2.4 Controller

Verwijder index.html.

#### 2.4.1 IndexController

```
package be.vdab.frida.controllers;
// enkele imports
@RestController
class IndexController {
  @GetMapping("/")
  public String index() {
    var openGesloten = LocalDate.now().getDayOfWeek() == DayOfWeek.MONDAY ?
      "gesloten" : "open";
    return "<!doctype html><html><title>Hallo</title><body>" + openGesloten
      + "</body></html>";
  }
}
```

## 2.5 Thymeleaf

### 2.5.1 IndexController

```
package be.vdab.frida.controllers;
// enkele imports
@Controller
class IndexController {
    @GetMapping("/")
    public ModelAndView index() {
        var openGesloten = LocalDate.now().getDayOfWeek() == DayOfWeek.MONDAY ?
            "gesloten" : "open";
        return new ModelAndView("index", "openGesloten", openGesloten);
    }
}
```

### 2.5.2 index.html

```
<!doctype html>
<html lang="nl" xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>Frituur Frida</title>
        <link rel="icon" href="images/frida.ico" type="image/x-icon">
        <link rel="stylesheet" href="css/frida.css">
    </head>
    <body>
        <h1>We zijn vandaag <span th:text="${openGesloten}"></span></h1>
    </body>
</html>
```

## 2.6 Adres

### 2.6.1 Gemeente

```
package be.vdab.frida.domain;
public class Gemeente {
    private final String naam;
    private final int postcode;
    // constructor met parameters, getters
}
```

### 2.6.2 Adres

```
package be.vdab.frida.domain;
public class Adres {
    private final String straat;
    private final String huisNr;
    private final Gemeente gemeente;
    // constructor met parameters, getters
}
```

### 2.6.3 IndexController

Gewijzigd return statement:

```
var modelAndView = new ModelAndView("index", "openGesloten", openGesloten);
modelAndView.addObject("adres",
    new Adres("Grote markt", "7", new Gemeente("Brussel", 1000)));
return modelAndView;
```

### 2.6.4 index.html

```
<h2>Adres</h2>
<div th:object="${adres}" th:text="|*{straat} *{huisNr} *{gemeente.postcode}
    *{gemeente.naam}|"></div>
```

## 2.7 Sauzen

### 2.7.1 Saus

```
package be.vdab.frida.domain;
public class Saus {
    private final long id;
    private final String naam;
    private final String[] ingredienten;
    // constructor met parameters, getters
}
```

### 2.7.2 SausController

```
package be.vdab.frida.controllers;
// enkele imports
@Controller
class SausController {
    private final Saus[] alleSauzen = {
        new Saus(3, "cocktail", new String[] {"mayonaise", "ketchup", "cognac"}),
        new Saus(6, "mayonaise", new String[] {"ei", "mosterd"}),
        new Saus(7, "mosterd", new String[] {"mosterd", "azijn", "witte wijn"}),
        new Saus(12, "tartare", new String[] {"mayonaise", "augurk", "tabasco"}),
        new Saus(44, "vinaigrette", new String[] {"olijfolie", "mosterd", "azijn"}});
    @GetMapping("/sauzen")
    public ModelAndView findAll() {
        return new ModelAndView("sauzen", "alleSauzen", alleSauzen);
    }
}
```

### 2.7.3 sauzen.html

```
<!doctype html>
<html lang="nl" xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>Sauzen</title>
        <link rel="icon" href="images/frida.ico" type="image/x-icon">
        <link rel="stylesheet" href="css/frida.css">
    </head>
    <body>
        <h1>Sauzen</h1>
        <ul>
            <li th:each="saus : ${alleSauzen}" th:object="${saus}">
                <span th:text="*{naam}"></span>
                <ul>
                    <li th:each="ingredient : *{ingredienten}"
                        th:text="${ingredient}"></li>
                </ul>
            </li>
        </ul>
    </body>
</html>
```

## 2.8 URL

### 2.8.1 index.html en sauzen.html

```
<link rel="icon" th:href="@{/images/frida.ico}" type="image/x-icon">
<link rel="stylesheet" th:href="@{/css/frida.css}">
```

## 2.9 Fragments

### 2.9.1 fragments.html

```
<div xmlns:th="http://www.thymeleaf.org">
  <nav th:fragment="menu">
    <ul>
      <li><a th:href="@{/}">Welkom</a></li>
      <li><a th:href="@{/sauzen}">Sauzen</a></li>
    </ul>
  </nav>
  <head th:fragment="head(title)">
    <link rel="icon" th:href="@{/images/frida.ico}" type="image/x-icon">
    <title th:text="${title}"></title>
    <link rel="stylesheet" th:href="@{/css/frida.css}">
  </head>
</div>
```

### 2.9.2 index.html

Vervang <head>...</head> door

```
<head th:replace="fragments::head(title='Frituur Frida')"></head>
```

Typ na <body>

```
<nav th:replace="fragments::menu"></nav>
```

### 2.9.3 sauzen.html

Vervang <head>...</head> door

```
<head th:replace="fragments::head(title='Sauzen')"></head>
```

Typ na <body>

```
<nav th:replace="fragments::menu"></nav>
```

### 2.9.4 frida.css

Voeg opmaak toe voor nav ul, nav li, nav li a en nav li a:hover zoals in de theorie.

## 2.10 URL's en methods

- |                              |                      |                        |
|------------------------------|----------------------|------------------------|
| • /landen/BE                 | land                 | BE hoort bij {code}    |
| • /landen                    | landen               |                        |
| • /landen/NL                 | land                 | NL hoort bij {code}    |
| • /landen/rijkste10          | rijkste10            |                        |
| • /landen/werelddelen/europa | vanWerelddeel europa | hoort bij {werelddeel} |

## 2.11 Path variabele

### 2.11.1 sauzen.html

Vervang

```
<li th:each="saus : ${alleSauzen}" th:object="${saus}">
...
</li>
```

door

```
<li th:each="saus : ${alleSauzen}" th:object="${saus}">
  <a th:href="@{/sauzen/{id}(id={id})}" th:text="{naam}"></a>
</li>
```

### 2.11.2 SausController

Extra methods:

```
private Optional<Saus> findByIdHelper(long id) {
    return Arrays.stream(alleSauzen).filter(saus->saus.getId() == id).findFirst();
}
@GetMapping("/sauzen/{id}")
public ModelAndView findById(@PathVariable long id) {
    var modelAndView = new ModelAndView("saus");
    findByIdHelper(id).ifPresent(saus -> modelAndView.addObject("saus", saus));
    return modelAndView;
}
```

### 2.11.3 saus.html

```
<!doctype html>
<html lang="nl" xmlns:th="http://www.thymeleaf.org">
  <head th:replace="fragments::head(title=${saus} == null ?
    'Saus niet gevonden' : ${saus.naam})"></head>
  <body>
    <nav th:replace="fragments::menu"></nav>
    <h1 th:if="not ${saus}">Saus niet gevonden:
      <th:block th:text="{id}"></th:block>
    </h1>
    <th:block th:if="${saus}" th:object="${saus}">
      <h1 th:text="{naam}"></h1>
      <dl>
        <dt>Nummer</dt><dd th:text="{id}"></dd>
        <dt>Ingrediënten</dt>
        <dd>
          <ul>
            <li th:each="ingredient : {ingredienten}" th:text="{ingredient}"></li>
          </ul>
        </dd>
      </dl>
      
    </th:block>
  </body>
</html>
```

### 2.11.4 frida.css

Extra code:

```
dd,dt {
    margin-left: 0;
}
dt {
    margin-top: 0.5em;
}
dd {
    font-weight: bold;
}
```

## 2.12 @RequestMapping

- Voor de class SausController: @RequestMapping("sauzen")
- @GetMapping("/sauzen")      wordt: @GetMapping
- @GetMapping("/sauzen/{id}")      wordt @GetMapping("id")

## 2.13 Taal

### 2.13.1 TaalController

```
package be.vdab.frida.controllers;
// enkele imports
@Controller
@RequestMapping("talen")
class TaalController {
    @GetMapping
    public ModelAndView nederlands(@RequestHeader("Accept-Language") String language) {
        return new ModelAndView("talen", "nederlands", language.startsWith("nl"));
    }
}
```

### 2.13.2 talen.html

```
<!doctype html>
<html lang="nl" xmlns:th="http://www.thymeleaf.org">
    <head th:replace="fragments::head(title='Taal')"></head>
    <body>
        <nav th:replace="fragments::menu"></nav>
        <h1>U spreekt
            <th:block th:if="not ${nederlands}">geen </th:block>Nederlands.
        </h1>
    </body>
</html>
```

### 2.13.3 fragments.html

Extra regel:

```
<li><a th:href="@{/talen}">Taal</a></li>
```

## 2.14 Bezocht

### 2.14.1 IndexController

Extra variabele:

```
private static final int EEN_JAAR_IN_SECONDS = 31_536_000;
```

Declaratie van de method index wijzgen:

```
@GetMapping
public ModelAndView index(@CookieValue Optional<Integer> aantalBezoeken,
    HttpServletResponse response) {
    return statement van de method index wijzgen:
    var nieuwAantalBezoeken = aantalBezoeken.orElse(0) + 1;
    var cookie = new Cookie("aantalBezoeken", String.valueOf(nieuwAantalBezoeken));
    cookie.setMaxAge(EEN_JAAR_IN_SECONDS);
    cookie.setPath("/");
    response.addCookie(cookie);
    modelAndView.addObject("aantalBezoeken", nieuwAantalBezoeken);
    return modelAndView;
}
```

### 2.14.2 index.html

```
<div>Je bezoekt ons voor de
<th:block th:text="${aantalBezoeken}"></th:block>° keer.</div>
```

## 2.15 Alfabet

### 2.15.1 SausController

Extra code:

```
private final char[] alfabet = "abcdefghijklmnopqrstuvwxyz".toCharArray();
@GetMapping("alfabet")
public ModelAndView alfabet() {
    return new ModelAndView("sausAlfabet", "alfabet", alfabet);
}
private Stream<Saus> findByBeginNaamHelper(char letter) {
    return Arrays.stream(allesauzen)
        .filter(saus -> saus.getNaam().charAt(0) == letter);
}
@GetMapping("alfabet/{letter}")
public ModelAndView findByBeginNaam(@PathVariable char letter) {
    return new ModelAndView("sausAlfabet", "alfabet", alfabet)
        .addObject("sauzen", findByBeginNaamHelper(letter).iterator());
}
```

### 2.15.2 sausAlfabet.html

```
<!doctype html>
<html lang="nl" xmlns:th="http://www.thymeleaf.org">
  <head th:replace="fragments::head(title='Saus alfabet')"></head>
  <body>
    <nav th:replace="fragments::menu"></nav>
    <h1>Alfabet</h1>
    <ul class="naastMekaar">
      <li th:each="eenLetter : ${alfabet}">
        <a th:href="@{/sauzen/alfabet/{letter}(letter=${eenLetter})}"
           th:text="${eenLetter}"></a>
      </li>
    </ul>
    <th:block th:if="${letter}">
      <h2>Sauzen die beginnen met <th:block th:text="${letter}"></th:block></h2>
      <ul><li th:each="saus : ${sauzen}" th:text="${saus.naam}"></li></ul>
    </th:block>
  </body>
</html>
```

### 2.15.3 frida.css

Extra regels:

```
.naastMekaar {
  padding-left: 0;
}
.naastMekaar li {
  display: inline;
}
```

### 2.15.4 fragments.html

Extra regel: <li><a th:href="@{/sauzen/alfabet}">Saus alfabet</a></li>



## 2.16 Sauzen.csv

### 2.16.1 SausRepository

### 2.16.2 SausRepositoryException

```
package be.vdab.frida.exceptions;
public class SausRepositoryException extends RuntimeException {
    private static final long serialVersionUID = 1L;
    public SausRepositoryException(String message) {
        super(message);
    }
    public SausRepositoryException(String message, Exception originelFout) {
        super(message, originelFout);
    }
}
```

### 2.16.3 CSVSausRepository

```
package be.vdab.frida.repositories;
// enkele imports
@Component
public class CSVSausRepository {
    private static final Path PAD = Paths.get("/data/sauzen.csv");
    public Stream<Saus> findAll() {
        try {
            return Files.lines(PAD).map(this::maakSaus);
        } catch (IOException ex) {
            throw new SausRepositoryException("Fout bij lezen " + PAD, ex);
        }
    }
    private Saus maakSaus(String regel) {
        var onderdelen = regel.split(",");
        if (onderdelen.length < 2) {
            throw new SausRepositoryException(PAD+": "+regel+": minder dan 2 onderdelen");
        }
        try {
            var ingredienten = Arrays.copyOfRange(onderdelen, 2, onderdelen.length);
            return new Saus(Long.parseLong(onderdelen[0]),onderdelen[1],ingredienten);
        } catch (NumberFormatException ex) {
            throw new SausRepositoryException(PAD + ":" + regel + ": verkeerde id", ex);
        }
    }
}
```

#### 2.16.4 CSVSausRepositoryTest

```
package be.vdab.frida.repositories;
// enkele imports
class CSVSausRepositoryTest {
    private static final Path PAD = Path.of("/data/sauzen.csv");
    private CSVSausRepository repository;
    @BeforeEach
    void beforeEach() {
        repository = new CSVSausRepository();
    }
    @Test
    void erZijnEvenveelSauzenAlsErRegelsZijnInHetCSVBestand() throws IOException {
        assertThat(repository.findAll()).hasSameSizeAs(Files.readAllLines(PAD));
    }
    @Test
    void deEersteSausBevatDeDataVanDeEersteRegelInHetCSVBestand()
        throws IOException {
        var eersteRegel = Files.lines(PAD).findFirst().get();
        var eersteSaus = repository.findAll().findFirst().get();
        assertThat(eersteSaus.getId() + "," + eersteSaus.getNaam() + "," +
            Arrays.stream(eersteSaus.getIngredienten())
                .collect(Collectors.joining(","))
            ).isEqualTo(eersteRegel);
    }
}
```

#### 2.16.5 SausService

```
package be.vdab.frida.services;
// enkele imports
@Service
public class SausService {
    private final CSVSausRepository sausRepository;
    // constructor met parameter
    public Stream<Saus> findAll() {
        return sausRepository.findAll();
    }
    public Stream<Saus> findByBeginNaam(char letter) {
        return sausRepository.findAll()
            .filter(saus -> saus.getNaam().charAt(0) == letter);
    }
    public Optional<Saus> findById(long id) {
        return sausRepository.findAll()
            .filter(saus -> saus.getId() == id)
            .findFirst();
    }
}
```

#### 2.16.6 SausController

```
package be.vdab.frida.controllers;
// enkele imports
@Controller
@RequestMapping("sauzen")
class SausController {
    private final char[] alfabet = "abcdefghijklmnopqrstuvwxyz".toCharArray();
    private final SausService sausService;
    // constructor met parameter
    @GetMapping
    public ModelAndView findAll() {
        return new ModelAndView("sauzen",
            "alleSauzen",sausService.findAll().iterator());
    }
}
```

```

@GetMapping("/{id}")
public ModelAndView findById(@PathVariable long id) {
    var modelAndView = new ModelAndView("saus");
    sausService.findById(id).ifPresent(saus -> modelAndView.addObject(saus));
    return modelAndView;
}
@GetMapping("alfabet")
public ModelAndView alfabet() {
    return new ModelAndView("sausAlfabet", "alfabet", alfabet);
}
@GetMapping("alfabet/{letter}")
public ModelAndView findByBeginNaam(@PathVariable char letter) {
    return new ModelAndView("sausAlfabet", "alfabet", alfabet)
        .addObject("sauzen", sausService.findByBeginNaam(letter).iterator());
}
}

```

## 2.17 Sauzen.properties

### 2.17.1 SausRepository

```

package be.vdab.frida.repositories;
// enkele imports
public interface SausRepository {
    Stream<Saus> findAll();
}

```

### 2.17.2 CSVSausRepository

Extra regel voor de class: `@Qualifier("CSV")`

Bij de class zelf: `implements SausRepository`

Voor de method `findAll`: `@Override`

### 2.17.3 PropertiesSausRepository

```

package be.vdab.frida.repositories;
// enkele imports
@Component @Qualifier("properties")
class PropertiesSausRepository implements SausRepository {
    private static final Path PAD = Paths.get("/data/sauzen.properties");
    @Override
    public Stream<Saus> findAll() {
        try {
            return Files.lines(PAD).map(this::maakSaus);
        } catch (IOException ex) {
            throw new SausRepositoryException("Fout bij lezen " + PAD, ex);
        }
    }
    private Saus maakSaus(String regel) {
        var onderdelen = regel.split(":");
        if (onderdelen.length < 2) {
            throw new SausRepositoryException(PAD+": "+regel+": minder dan 2 onderdelen");
        }
        try {
            var naamEnIngredienten = onderdelen[1].split(",");
            var ingredienten =
                Arrays.copyOfRange(naamEnIngredienten, 1, naamEnIngredienten.length);
            return new Saus(Long.parseLong(onderdelen[0]), naamEnIngredienten[0],
                ingredienten);
        } catch (NumberFormatException ex) {
            throw new SausRepositoryException(PAD + ":" + regel + ": verkeerde id", ex);
        }
    }
}

```

### 2.17.4 PropertiesSausRepositoryTest

```
package be.vdab.frida.repositories;
// enkele imports
class PropertiesSausRepositoryTest {
    private static final Path PAD = Path.of("/data/sauzen.properties");
    private PropertiesSausRepository repository;
    @BeforeEach
    void beforeEach() {
        repository = new PropertiesSausRepository();
    }
    @Test
    void erZijnEvenveelSauzenAlsErRegelsZijnInHetCSVBestand() throws IOException {
        assertThat(repository.findAll()).hasSameSizeAs(Files.readAllLines(PAD));
    }
    @Test
    void deEersteSausBevatDeDataVanDeEersteRegelInHetCSVBestand()
        throws IOException {
        var eersteRegel = Files.lines(PAD).findFirst().get();
        var eersteSaus = repository.findAll().findFirst().get();
        assertThat(eersteSaus.getId() + ":" + eersteSaus.getNaam() + "," +
            Arrays.stream(eersteSaus.getIngrediënten())
                .collect(Collectors.joining(","))
            ).isEqualTo(eersteRegel);
    }
}
```

### 2.17.5 SausService

Private variabele sausRepository krijgt type SausRepository en gewijzigde constructor:

```
public SausService(@Qualifier("properties") SausRepository sausRepository) {
    this.sausRepository = sausRepository;
}
```

## 2.18 Application.properties

### 2.18.1 application.properties

Extra regels:

```
CSVSausenPad=/data/sauzen.csv
propertiesSausenPad=/data/sauzen.properties
```

### 2.18.2 CSVSausRepository

```
package be.vdab.frida.repositories;
// enkele imports
@Component
@Qualifier("CSV")
class CSVSausRepository implements SausRepository {
    private final Path pad;
    CSVSausRepository(@Value("${CSVSausenPad}") Path pad) {
        this.pad = pad;
    }
    // vervang in de rest van de code PAD door pad
}
```

### 2.18.3 CSVSausRepositoryTest

```
package be.vdab.frida.repositories;
// enkele imports
@ExtendWith(SpringExtension.class)
@PropertySource("application.properties")
@Import(CSVSausRepository.class)
class CSVSausRepositoryTest {
    private final CSVSausRepository repository;
    private final Path pad;
```

```

CSVSausRepositoryTest(CSVSausRepository repository,
    @Value("${CSVSausenPad}") Path pad) {
    this.repository = repository;
    this.pad = pad;
}
@Test
void erZijnEvenveelSausenAlsErRegelsZijnInHetCSVBestand() throws IOException {
    assertThat(repository.findAll()).hasSameSizeAs(Files.readAllLines(pad));
}
@Test
void deEersteSausBevatDeDataVanDeEersteRegelInHetCSVBestand()
    throws IOException {
    var eersteRegel = Files.lines(pad).findFirst().get();
    var eersteSaus = repository.findAll().findFirst().get();
    assertThat(eersteSaus.getId() + "," + eersteSaus.getNaam() + "," +
        Arrays.stream(eersteSaus.getIngredienten())
            .collect(Collectors.joining(",")))
        .isEqualTo(eersteRegel);
}
}

```

#### 2.18.4 PropertiesSausRepository

```

package be.vdab.frida.repositories;
// enkele imports
@Component
@Qualifier("properties")
class PropertiesSausRepository implements SausRepository {
    private final Path pad;
    PropertiesSausRepository(@Value("${propertiesSausenPad}") Path pad) {
        this.pad = pad;
    }
    // vervang in de rest van de code PAD door pad
}

```

#### 2.18.5 PropertiesSausRepositoryTest

```

package be.vdab.frida.repositories;
// enkele imports
@ExtendWith(SpringExtension.class)
@PropertySource("application.properties")
@Import(PropertiesSausRepository.class)
class PropertiesSausRepositoryTest {
    private final PropertiesSausRepository repository;
    private final Path pad;
    PropertiesSausRepositoryTest(PropertiesSausRepository repository,
        @Value("${propertiesSausenPad}") Path pad) {
        this.repository = repository;
        this.pad = pad;
    }
    @Test
    void erZijnEvenveelSausenAlsErRegelsZijnInHetBestand() throws IOException {
        assertThat(repository.findAll()).hasSameSizeAs(Files.readAllLines(pad));
    }
    @Test
    void deEersteSausBevatDeDataVanDeEersteRegelInHetBestand() throws IOException {
        var eersteRegel = Files.lines(pad).findFirst().get();
        var eersteSaus = repository.findAll().findFirst().get();
        assertThat(eersteSaus.getId() + ":" + eersteSaus.getNaam() + "," +
            Arrays.stream(eersteSaus.getIngredienten())
                .collect(Collectors.joining(",")))
            .isEqualTo(eersteRegel);
    }
}

```

## 2.19 Database toegang

Zelfde werkwijze als in theorie.

## 2.20 DataSource

### 2.20.1 pom.xml

Zelfde dependencies toevoegen als in theorie.

### 2.20.2 application.properties

```
spring.datasource.url=jdbc:mysql://localhost/frida
spring.datasource.username=cursist
spring.datasource.password=cursist
```

### 2.20.3 DataSourceTest

Zelfde als in theorie, maar "luigi" wordt "frida".

## 2.21 Repository

### 2.21.1 Snack

```
package be.vdab.frida.domain;
import java.math.BigDecimal;
public class Snack {
    private final long id;
    private final String naam;
    private final BigDecimal prijs;
    // constructor met parameters, getters
}
```

### 2.21.2 SnackNietGevondenException

```
package be.vdab.frida.exceptions;
public class SnackNietGevondenException extends RuntimeException {
    private static final long serialVersionUID = 1L;
}
```

### 2.21.3 SnackRepository

```
package be.vdab.frida.repositories;
// enkele imports
@Repository
public class SnackRepository {
    private final JdbcTemplate template;
    // constructor met parameter
    private final RowMapper<Snack> snackRowMapper =
        (result, rowNum) -> new Snack(result.getLong("id"),
            result.getString("naam"), result.getBigDecimal("prijs"));
    public void update(Snack snack) {
        var sql = """
            update snacks
            set naam = ?, prijs = ?
            where id = ?
            """;
        if (template.update(sql, snack.getNaam(), snack.getPrijs(), snack.getId()) == 0){
            throw new SnackNietGevondenException();
        }
    }
}
```

```

public List<Snack> findByBeginNaam(String beginNaam) {
    var sql = """
        select id, naam, prijs
        from snacks
        where naam like ?
        order by naam
        """;
    return template.query(sql, snackRowMapper, beginNaam + '%');
}
public Optional<Snack> findById(long id) {
    try {
        var sql = """
            select id, naam, prijs
            from snacks
            where id = ?
            """;
        return Optional.of(template.queryForObject(sql, snackRowMapper, id));
    } catch (IncorrectResultSizeDataAccessException ex) {
        return Optional.empty();
    }
}
}
}

```

## 2.22 Repository test

### 2.22.1 application.properties

spring.test.database.replace=none

### 2.22.2 spring.properties

In src/test/resources: spring.test.constructor.autowire.mode=all

### 2.22.3 insertSnacks.sql

In src/test/resources: insert into snacks(naam,prijs) values('test', 10);

### 2.22.4 SnackRepositoryTest

```

package be.vdab.frida.repositories;
// enkele imports
@JdbcTest @Sql("/insertSnacks.sql")
@Import(SnackRepository.class)
class SnackRepositoryTest extends AbstractTransactionalJUnit4SpringContextTests{
    private final static String SNACKS = "snacks";
    private final SnackRepository repository;
    // constructor met parameter
    private long idVanTestSnack() {
        return jdbcTemplate.queryForObject(
            "select id from snacks where naam='test'", Long.class);
    }
    @Test
    void update() {
        var id = idVanTestSnack();
        var snack = new Snack(id, "test", BigDecimal.TEN);
        repository.update(snack);
        assertThat(countRowsInTableWhere(SNACKS, "prijs=10 and id=" + id).isOne();
    }
    @Test
    void updateOnbestaandeSnack() {
        assertThatExceptionOfType(SnackNietGevondenException.class)
            .isThrownBy(() -> repository.update(new Snack(-1, "test", BigDecimal.TEN)));
    }
}

```

```

@Test
void findById() {
    assertThat(repository.findById(idVanTestSnack()))
        .hasValueSatisfying(snack -> assertThat(snack.getNaam()).isEqualTo("test"));
}
@Test
void findByOnbestaandeIdVindtGeenSnack() {
    assertThat(repository.findById(-1)).isEmpty();
}
@Test
void findByBeginNaam() {
    assertThat(repository.findByBeginNaam("t"))
        .hasSize(countRowsInTableWhere(SNACKS, "naam like 't%'"))
        .extracting(Snack::getNaam)
        .allSatisfy(naam -> assertThat(naam.toLowerCase()).startsWith("t"))
        .isSortedAccordingTo(String::compareToIgnoreCase);
}
}

```

### 2.22.5 application.properties

```

logging.level.org.springframework.jdbc.core.JdbcTemplate=DEBUG
logging.level.org.springframework.jdbc.core.simple.SimpleJdbcInsert=DEBUG
logging.level.org.springframework.jdbc.core.StatementCreatorUtils=TRACE

```

## 2.23 Services

### 2.23.1 application.properties

```

spring.datasource.hikari.transaction-isolation=TRANSACTION_READ_COMMITTED

```

### 2.23.2 SnackService

```

package be.vdab.frida.services;
// enkele imports
@Service
@Transactional(readOnly = true)
public class SnackService {
    private final SnackRepository snackRepository;
    // constructor met parameter
    @Transactional
    public void update(Snack snack) {
        snackRepository.update(snack);
    }
    public List<Snack> findByBeginNaam(String beginNaam) {
        return snackRepository.findByBeginNaam(beginNaam);
    }
    public Optional<Snack> findById(long id) {
        return snackRepository.findById(id);
    }
}

```



### 2.23.3 SnackController

```
package be.vdab.frida.controllers;
// enkele imports
@Controller
@RequestMapping("snacks")
class SnackController {
    private final char[] alfabet = "abcdefghijklmnopqrstuvwxyz".toCharArray();
    private final SnackService snackService;
    // constructor met parameter
    @GetMapping("alfabet")
    public ModelAndView alfabet() {
        return new ModelAndView("snackAlfabet", "alfabet", alfabet);
    }
    @GetMapping("alfabet/{letter}")
    public ModelAndView findByBeginNaam(@PathVariable char letter) {
        return new ModelAndView("snackAlfabet", "alfabet", alfabet)
            .addObject("snacks", snackService.findByBeginNaam(String.valueOf(letter)));
    }
}
```

### 2.23.4 snackAlfabet.html

```
<!doctype html>
<html lang="nl" xmlns:th="http://www.thymeleaf.org">
    <head th:replace="fragments::head(title='Snack alfabet')"></head>
    <body>
        <nav th:replace="fragments::menu"></nav>
        <h1>Alfabet</h1>
        <ul class="naastMekaar">
            <li th:each="eenLetter : ${alfabet}">
                <a th:href="@{/snacks/alfabet/{deLetter}(deLetter=${eenLetter})}"
                    th:text="${eenLetter}"></a></li>
            </ul>
            <th:block th:if="${letter}">
                <h2>Snacks die beginnen met <th:block th:text="${letter}"></th:block></h2>
                <ul><li th:each="snack : ${snacks}" th:text="${snack.naam}"></li></ul>
            </th:block>
        </body>
    </html>
```

### 2.23.5 fragments.html

Extra regel: <li><a th:href="@{/snacks/alfabet}">Snack alfabet</a></li>

## 2.24 Dagverkopen

### 2.24.1 VerkochtAantalPerSnack

```
package be.vdab.frida.dto;
public record VerkochtAantalPerSnack(long id, String naam, int totaalAantal) {
}
```

### 2.24.2 SnackRepository

Extra method:

```
public List<VerkochtAantalPerSnack> findVerkochteAantallenPerSnack() {
    var sql = """
        select id, naam, sum(aantal) as totaalAantal
        from snacks left outer join dagverkopen on snacks.id = dagverkopen.snackId
        group by id, naam
        order by id
        """;
    RowMapper<VerkochtAantalPerSnack> mapper = (result, rowNum) ->
        new VerkochtAantalPerSnack(result.getLong("id"), result.getString("naam"),
            result.getInt("totaalAantal"));
    return template.query(sql, mapper);
}
```

### 2.24.3 insertDagverkopen.sql

```
insert into dagverkopen(snackid, datum, aantal)
values ((select id from snacks where naam='test'), curdate(), 10);
```

### 2.24.4 SnackRepositoryTest

Gewijzigde regel:

```
@Sql({"insertSnacks.sql", "insertDagverkopen.sql"})
```

Extra method:

```
@Test
void findVerkochtAantalPerSnack() {
    var verkochteAantallenPerSnack = repository.findVerkochteAantallenPerSnack();
    assertThat(verkochteAantallenPerSnack).hasSize(countRowsInTable(SNACKS));
    var rij1 = verkochteAantallenPerSnack.get(0);
    assertThat(rij1.totaalAantal()).isEqualTo(jdbcTemplate.queryForObject(
        "select sum(aantal) from dagVerkopen where snackId = " + rij1.id(),
        Integer.class));
}
```

### 2.24.5 SnackService

Extra method:

```
public List<VerkochtAantalPerSnack> findVerkochteAantallenPerSnack() {
    return snackRepository.findVerkochteAantallenPerSnack();
}
```

### 2.24.6 SnackController

Extra method:

```
@GetMapping("verkochteaantallenpersnack")
ModelAndView findVerkochteAantallenPerSnack() {
    return new ModelAndView("verkochteaantallenpersnack",
        "verkochteAantallenPerSnack", snackService.findVerkochteAantallenPerSnack());
}
```

### 2.24.7 verkochteaantallenpersnack.html

```
<!doctype html>
<html lang="nl" xmlns:th="http://www.thymeleaf.org">
<head th:replace="fragments::head(title='Verkochte aantallen per snack')"></head>
<body>
  <nav th:replace="fragments::menu"></nav>
  <h1>Verkochte aantallen per snack</h1>
  <table>
    <thead>
      <tr>
        <th>Nummer</th><th>Naam</th><th>Totaal aantal</th>
      </tr>
    </thead>
    <tbody>
      <tr th:each="verkochtAantalPerSnack:${verkochteAantallenPerSnack}"
        th:object="${verkochtAantalPerSnack}">
        <td th:text="*{id}"></td>
        <td th:text="*{naam}"></td>
        <td th:text="*{totaalAantal}"></td>
      </tr>
    </tbody>
  </table>
</body>
</html>
```

### 2.24.8 fragments.html

Extra regel: `<li><a th:href="@{/snacks/verkochteaantallenpersnack}">Verkochte aantallen per snack</a></li>`

## 2.25 Form

### 2.25.1 BeginNaamForm

```
package be.vdab.frida.forms;
public record BeginNaamForm(String begin) {}
```

### 2.25.2 SnackController

Extra code:

```
@GetMapping("beginnaam/form")
public ModelAndView beginNaamForm() {
    return new ModelAndView("beginNaam").addObject(new BeginNaamForm(""));
}
@GetMapping("beginnaam")
public ModelAndView findByBeginNaam(BeginNaamForm form) {
    return new ModelAndView("beginNaam")
        .addObject("snacks", snackService.findByBeginNaam(form.begin()));
}
```

### 2.25.3 beginNaam.html

```
<!doctype html>
<html lang="nl" xmlns:th="http://www.thymeleaf.org">
  <head th:replace="fragments::head(title='Begin naam')"></head>
  <body>
    <nav th:replace="fragments::menu"></nav>
    <h1>Begin naam</h1>
    <form th:object="${beginNaamForm}" method="get" th:action="@{/snacks/beginnaam}">
      <label>Begin naam:<input th:field="*{begin}" autofocus></label>
      <button>Zoeken</button>
    </form>
    <ul th:if="${snacks}">
      <li th:each="snack : ${snacks}" th:text="${snack.naam}"></li>
    </ul>
  </body>
</html>
```

### 2.25.4 fragments.html

Extra regel: `<li><a th:href="@{/snacks/beginnaam/form}">Begin naam</a></li>`

## 2.26 Bean Validation

### 2.26.1 BeginNaamForm

`@NotBlank` voor String begin;

### 2.26.2 SnackController

Gewijzigde method:

```
@GetMapping("beginnaam")
public ModelAndView findByBeginNaam(@Valid BeginNaamForm form, Errors errors) {
    var modelAndView = new ModelAndView("beginNaam");
    if (errors.hasErrors()) {
        return modelAndView;
    }
    return modelAndView.addObject("snacks",
        snackService.findByBeginNaam(form.begin()));
}
```

### 2.26.3 messages.properties

Voeg de foutboodschappen toe die horen bij bean validation.

### 2.26.4 beginNaam.html

Gewijzigde regel:

`<label>Begin naam:<span th:errors="*{begin}"></span> ...`

### 2.26.5 frida.css

Extra regels:

```
label span, .fout {
    background-color: red;
    color:white;
    font-weight: bold;
    padding-left:0.5em;
    padding-right: 0.5em;
}
label span {
    margin-left: 0.5em;
}
label {
    cursor: pointer;
}
input {
    display: block;
    margin-top: 0.2em;
    margin-bottom: 1em;
    font-size: 1.1em;
}
```

## 2.27 Client sided validatie

### 2.27.1 beginNaam.html

Uitbreid `<input>` element: `<input th:field="*{begin}" autofocus required>`

## 2.28 Snack wijzigen

### 2.28.1 snackAlfabet.html en beginNaam.html

Gewijzigde `<li>...</li>` regel:

```
<li th:each="snack : ${snacks}">
  <a th:href="@{/snacks/{id}/wijzigen/form(id=${snack.id})}"
    th:text="${snack.naam}"></a></li>
```

### 2.28.2 Snack

```
...
@NotBlank
private String naam;
@NotNull @PositiveOrZero
private BigDecimal prijs;
...
```

### 2.28.3 SnackController

Extra code:

```
@GetMapping("/{id}/wijzigen/form")
public ModelAndView wijzigenForm(@PathVariable long id) {
    var modelAndView = new ModelAndView("wijzigSnack");
    snackService.findById(id).ifPresent(snack -> modelAndView.addObject(snack));
    return modelAndView;
}
@PostMapping("wijzigen")
public String wijzigen(@Valid Snack snack, Errors errors, RedirectAttributes redirect){
    if (errors.hasErrors()) {
        return "wijzigSnack";
    }
    try {
        snackService.update(snack);
        return "redirect:/";
    } catch (SnackNietGevondenException ex) {
        redirect.addAttribute("snackNietGevonden", snack.getId());
        return "redirect:/";
    }
}
```

### 2.28.4 wijzigSnack.html

```
<!doctype html>
<html lang="nl" xmlns:th="http://www.thymeleaf.org">
  <head th:replace="fragments::head(title='Snack wijzigen')"></head>
  <body>
    <script th:src="@{/js/preventDoubleSubmit?js}" defer></script>
    <nav th:replace="fragments::menu"></nav>
    <h1>Snack wijzigen</h1>
    <form th:if="${snack}" th:object="${snack}" method="post"
      th:action="@{/snacks/wijzigen}">
      <label>Naam:<span th:errors="*{naam}"></span>
      <input th:field="*{naam}" autofocus required></label>
      <label>Prijs:<span th:errors="*{prijs}"></span>
      <input th:field="*{prijs}" type="number" required min="0" step="0.01">
      </label>
      <input type="hidden" name="id" th:value="*{id}">
      <button>Wijzigen</button>
    </form>
    <div class="fout" th:if="not ${snack}">Snack niet gevonden</div>
  </body>
</html>
```

### 2.28.5 preventDoubleSubmit.js

```
document.querySelector("form").onsubmit = function() {
    this.querySelector("button").disabled = true;
};
```

### 2.28.6 messages.properties

typeMismatch.java.math.BigDecimal=typ een bedrag

### 2.28.7 Index.html

Extra regel:

```
<div th:if="${param.snackNietGevonden}" class="fout">Snack
    <th:block th:text="${param.snackNietGevonden}"></th:block> niet gevonden</div>
```

## 2.29 Zoek de friet

### 2.29.1 Deur

```
package be.vdab.frida.sessions;
import java.io.Serializable;
public class Deur implements Serializable {
    private static final long serialVersionUID = 1L;
    private final int index;
    private final boolean metFriet;
    private boolean open;
    public Deur(int index, boolean metFriet) {
        this.index = index;
        this.metFriet = metFriet;
    }
    public void open() { open = true; }
    public int getIndex() { return index; }
    public boolean isOpen() { return open; }
    public boolean isMetFriet() { return metFriet; }
}
```

### 2.29.2 ZoekDeFriet

```
package be.vdab.frida.sessions;
// enkele imports
@Component
@SessionScope
public class ZoekDeFriet implements Serializable {
    private static final long serialVersionUID = 1L;
    private static final int AANTAL_DEUREN = 7;
    private final Deur[] deuren = new Deur[AANTAL_DEUREN];
    public ZoekDeFriet() {
        reset();
    }
    public void openDeur(int index) {
        deuren[index].open();
    }
    public Deur[] getDeuren() {
        return deuren;
    }
    public void reset() {
        var random = new Random();
        var indexMetFriet = random.nextInt(AANTAL_DEUREN);
        for (var index = 0; index != AANTAL_DEUREN; index++) {
            deuren[index] = new Deur(index, index == indexMetFriet);
        }
    }
}
```

### 2.29.3 FrietController

```
package be.vdab.frida.controllers;
// enkele imports
@Controller
@RequestMapping("frieten")
class FrietController {
    private final ZoekDeFriet zoekDeFriet;
    // constructor met parameter
    @GetMapping("zoeken")
    public ModelAndView zoekDeFriet() {
        return new ModelAndView("zoekDeFriet").addObject(zoekDeFriet);
    }
    @PostMapping("zoeken/nieuwspel")
    public String nieuwSpel() {
        zoekDeFriet.reset();
        return "redirect:/frieten/zoeken";
    }
    @PostMapping("zoeken/opendeur")
    public String openDeur(int index) {
        zoekDeFriet.openDeur(index);
        return "redirect:/frieten/zoeken";
    }
}
```

### 2.29.4 zoekDeFriet.html

```
<!doctype html>
<html lang="nl" xmlns:th="http://www.thymeleaf.org">
    <head th:replace="fragments::head(title='Zoek de friet')"></head>
    <body>
        <nav th:replace="fragments::menu"></nav>
        <h1>Zoek de friet</h1>
        <form th:action="@{/frieten/zoeken/opendeur}" method="post">
            <button th:each="deur : ${zoekDeFriet.deuren}" name="index"
                th:object="${deur}" th:value="*{index}">
                
                
                
            </button>
        </form>
        <form th:action="@{/frieten/zoeken/nieuwspel}" method="post">
            <button>Nieuw spel</button>
        </form>
    </body>
</html>
```

### 2.29.5 fragments.html

Extra regel:

```
<li><a th:href="@{/frieten/zoeken}">Zoek de friet</a></li>
```

### 2.29.6 application.properties

```
server.servlet.session.tracking-modes=cookie
```

## 2.30 Raad de saus

### 2.30.1 SausRaden

```
package be.vdab.frida.sessions;
// enkele imports ...
@Component
@SessionScope
public class SausRaden implements Serializable {
    private static final long serialVersionUID = 1L;
    private static final int MAX_VERKEERDE_POGINGEN = 10;
    private String saus;
    private StringBuilder puntjes;
    private int verkeerdePogingen;
    public void reset(String saus) {
        this.saus = saus;
        puntjes = new StringBuilder(".").repeat(saus.length());
        verkeerdePogingen = 0;
    }
    public void gok(char letter) {
        var letterIndex = saus.indexOf(letter);
        if (letterIndex == -1) {
            verkeerdePogingen++;
        } else {
            do {
                puntjes.setCharAt(letterIndex, letter);
                letterIndex = saus.indexOf(letter, letterIndex + 1);
            } while (letterIndex != -1);
        }
    }
    public String getPuntjes() {
        return puntjes.toString();
    }
    public String getSaus() {
        return saus;
    }
    public int getVerkeerdePogingen() {
        return verkeerdePogingen;
    }
    public boolean isGewonnen() {
        return puntjes.indexOf(".") == -1;
    }
    public boolean isVerloren() {
        return verkeerdePogingen == MAX_VERKEERDE_POGINGEN;
    }
}
```

### 2.30.2 SausRadenTest

Gezien de class SausRaden een algoritme bevat in de method gok, is een test aangewezen:

```
package be.vdab.frida.sessions;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import static org.assertj.core.api.Assertions.assertThat;
class SausRadenTest {
    private SausRaden raden;
    @BeforeEach
    void beforeEach() {
        raden = new SausRaden();
        raden.reset("lol");
    }
}
```



```

@Test
void eenNieuwSpel() {
    assertThat(raden.getSaus()).isEqualTo("lol");
    assertThat(raden.getPuntjes()).isEqualTo("...");
    assertThat(raden.getVerkeerdePogingen()).isZero();
    assertThat(raden.isGewonnen()).isFalse();
    assertThat(raden.isVerloren()).isFalse();
}

@Test
void eenJuisteLetterRaden() {
    raden.gok('l');
    assertThat(raden.getPuntjes()).isEqualTo("l.l");
    assertThat(raden.getVerkeerdePogingen()).isZero();
    assertThat(raden.isGewonnen()).isFalse();
    assertThat(raden.isVerloren()).isFalse();
}

@Test
void eenVerkeerdeLetterRaden() {
    raden.gok('z');
    assertThat(raden.getPuntjes()).isEqualTo("...");
    assertThat(raden.getVerkeerdePogingen()).isOne();
    assertThat(raden.isGewonnen()).isFalse();
    assertThat(raden.isVerloren()).isFalse();
}

@Test
void deVolledigeSausRaden() {
    raden.gok('l');
    raden.gok('o');
    assertThat(raden.isGewonnen()).isTrue();
    assertThat(raden.isVerloren()).isFalse();
}

@Test
void jeVerliestBijTeVeelPogingen() {
    for (var poging = 1; poging <= 10; poging++) {
        raden.gok('|'); // teken dat niet in een saus voorkomt;
    }
    assertThat(raden.isGewonnen()).isFalse();
    assertThat(raden.isVerloren()).isTrue();
}
}

```

### 2.30.3 SausRadenForm

```

package be.vdab.frida.forms;
public record SausRadenForm(@NotNull Character letter) {}

```

### 2.30.4 SausController

Extra private variabele:

```
private final SausRaden sausRaden;
```

Uitgebreide constructor:

```

SausController(SausService sausService, SausRaden sausRaden) {
    this.sausService = sausService;
    this.sausRaden = sausRaden;
}

```

Extra methods:

```

private String randomSaus() {
    var sauzen = sausService.findAll().toList();
    var random = new Random();
    var randomIndex = random.nextInt(sauzen.size());
    return sauzen.get(randomIndex).getNaam();
}

```

```

@GetMapping("raden")
public ModelAndView radenForm() {
    sausRaden.reset(randomSaus());
    return new ModelAndView("sausRaden").addObject(sausRaden)
        .addObject(new SausRadenForm(null));
}
@PostMapping("raden/nieuwspel")
public String radenNieuwSpel() {
    return "redirect:/sauzen/raden";
}
@PostMapping("raden")
public ModelAndView raden(@Valid SausRadenForm form, Errors errors) {
    if (errors.hasErrors()) {
        return new ModelAndView("sausRaden").addObject(sausRaden);
    }
    sausRaden.gok(form.letter());
    return new ModelAndView("redirect:/sauzen/raden/volgendegok");
}
@GetMapping("raden/volgendegok")
public ModelAndView volgendeGok() {
    return new ModelAndView("sausRaden").addObject(sausRaden)
        .addObject(new SausRadenForm(null));
}

```

### 2.30.5 sausRaden.html

```

<!doctype html>
<html lang="nl" xmlns:th="http://www.thymeleaf.org">
    <head th:replace="fragments::head(title='Raad de saus')"></head>
    <body th:object="${sausRaden}">
        <nav th:replace="fragments::menu"></nav>
        <h1>Raad de saus</h1>
        <div th:if="*{verloren}">U bent verloren, de saus was
            <th:block th:text="*{saus}"></th:block>.</div>
        <div th:if="*{gewonnen}">U bent gewonnen, de saus was
            <th:block th:text="*{saus}"></th:block>.</div>
        <div th:if="not *{verloren} and not *{gewonnen}">Te raden saus:
            <th:block th:text="*{puntjes}"></th:block>
            <form th:object="${sausRadenForm}" th:action="@{/sauzen/raden}" method="post">
                <label>letter:<span th:errors="*{letter}"></span>
                    <input th:field="*{letter}" autofocus required size="1"maxlength="1">
                </label>
                <button>Raden</button>
            </form>
        </div>
        <form th:action="@{/sauzen/raden/nieuwspel}" method="post">
            <button>Nieuw spel</button>
        </form>
        
        </body>
    </html>

```

### 2.30.6 fragments.html

Extra regel:

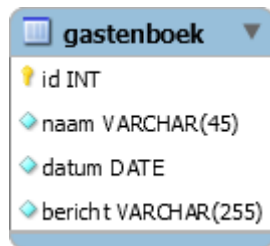
```

<li><a th:href="@{/sauzen/raden}">Raad de saus</a></li>

```

## 2.31 Gastenboek

### 2.31.1 Nieuwe table in de database



### 2.31.2 Rechten in de database

```
use frida;
grant insert,select on gastenboek to cursist
```

### 2.31.3 GastenBoekEntry

```
package be.vdab.frida.domain;
// enkele imports
public class GastenBoekEntry {
    private final long id;
    @NotBlank
    private final String naam;
    @NotNull
    @DateTimeFormat (style = "S-")
    private final LocalDate datum;
    @NotBlank
    private final String bericht;
    // constructor met parameters, getters
}
```

### 2.31.4 GastenBoekRepository

```
package be.vdab.frida.repositories;
// enkele imports
@Repository
public class GastenBoekRepository {
    private final SimpleJdbcInsert insert;
    private final JdbcTemplate template;
    public GastenBoekRepository(JdbcTemplate template) {
        this.template = template;
        insert = new SimpleJdbcInsert(template)
            .withTableName("gastenboek")
            .usingGeneratedKeyColumns("id");
    }
    public long create(GastenBoekEntry entry) {
        return insert.executeAndReturnKey(Map.of("naam", entry.getNaam(),
            "datum", entry.getDatum(), "bericht", entry.getBericht())).longValue();
    }
    private final RowMapper<GastenBoekEntry> entryRowMapper =
        (result, rowNum) -> new GastenBoekEntry(result.getLong("id"),
            result.getString("naam"), result.getObject("datum", LocalDate.class),
            result.getString("bericht"));
    public List<GastenBoekEntry> findAll() {
        var sql = """
            select id, naam, datum, bericht
            from gastenboek
            order by datum desc
            """;
        return template.query(sql, entryRowMapper);
    }
}
```

### 2.31.5 insertGastenBoek.sql

```
insert into gastenboek(naam,datum,bericht) values
('test','2019-01-01', 'test'),
('test2','2019-02-02', 'test2');
```

### 2.31.6 GastenBoekRepositoryTest

```
package be.vdab.frida.repositories;
// enkele imports
@JdbcTest
@Import(GastenBoekRepository.class)
@Sql("/insertGastenBoek.sql")
class GastenBoekRepositoryTest
    extends AbstractTransactionalJUnit4SpringContextTests {
    private static final String GASTENBOEK = "gastenboek";
    private final GastenBoekRepository repository;
    // constructor met parameter
    @Test
    void create() {
        var id = repository.create(
            new GastenBoekEntry(0, "test3", LocalDate.now(), "test"));
        assertThat(id).isPositive();
        assertThat(countRowsInTableWhere(GASTENBOEK, "id = " + id)).isOne();
    }
    @Test
    void findAll() {
        assertThat(repository.findAll())
            .hasSize(countRowsInTable(GASTENBOEK))
            .extracting(GastenBoekEntry::getDatum)
            .isSortedAccordingTo(Comparator.reverseOrder());
    }
}
```

### 2.31.7 GastenBoekService

```
package be.vdab.frida.services;
// enkele imports
@Service
@Transactional(readOnly = true)
public class GastenBoekService {
    private final GastenBoekRepository gastenBoekRepository;
    // constructor met parameter
    @Transactional
    public long create(GastenBoekEntry entry) {
        return gastenBoekRepository.create(entry);
    }
    public List<GastenBoekEntry> findAll() {
        return gastenBoekRepository.findAll();
    }
}
```

### 2.31.8 GastenBoekEntryForm

```
package be.vdab.frida.forms;
// enkele imports
public class GastenBoekEntryForm extends GastenBoekEntry {
    public GastenBoekEntryForm(String naam, String bericht) {
        super(0, naam, LocalDate.now(), bericht);
    }
}
```

### 2.31.9 GastenBoekController

```
package be.vdab.frida.controllers;
// enkele imports
@Controller
@RequestMapping("gastenboek")
public class GastenBoekController {
    private final GastenBoekService gastenBoekService;
    // constructor met parameter
    @GetMapping
    public ModelAndView findAll() {
        return new ModelAndView("gastenboek",
            "gastenboekEntries", gastenBoekService.findAll());
    }
    @GetMapping("toevoegen/form")
    public ModelAndView toevoegForm() {
        return new ModelAndView("gastenboek",
            "gastenboekEntries", gastenBoekService.findAll())
            .addObject(new GastenBoekEntryForm("", ""));
    }
    @PostMapping("toevoegen")
    public ModelAndView toevoegen(@Valid GastenBoekEntryForm form, Errors errors){
        if (errors.hasErrors()) {
            return new ModelAndView("gastenboek",
                "gastenboekEntries", gastenBoekService.findAll());
        }
        gastenBoekService.create(form);
        return new ModelAndView("redirect:/gastenboek");
    }
}
```

### 2.31.10 gastenboek.html

```
<!doctype html>
<html lang="nl" xmlns:th="http://www.thymeleaf.org">
    <head th:replace="fragments::head(title='Gastenboek')"></head>
    <body>
        <nav th:replace="fragments::menu"></nav>
        <h1>Gastenboek</h1>
        <a th:if="not ${gastenBoekEntryForm}"
            th:href="@{/gastenboek/toevoegen/form}">Toevoegen</a>
        <form th:if="${gastenBoekEntryForm}" method="post"
            th:object="${gastenBoekEntryForm}" th:action="@{/gastenboek/toevoegen}">
            <label>Naam:<span th:errors="*{naam}"></span>
                <input th:field="*{naam}" autofocus required></input>
            <label>Bericht:<span th:errors="*{bericht}"></span>
                <input th:field="*{bericht}" required></input>
            <button>Toevoegen</button>
        </form>
        <ul th:if="not ${gastenboekEntries.empty}">
            <li th:each="entry : ${gastenboekEntries}" th:object="${entry}">
                <strong th:text="|*{datum}| *{naam}|"></strong>
                <th:block th:text="*{bericht}"></th:block>
            </li>
        </ul>
    </body>
</html>
```

### 2.31.11 fragments.html

Extra regel: <li><a th:href="@{/gastenboek}">Gastenboek</a></li>

## 2.32 Gastenboekbeheer

### 2.32.1 Rechten in database

```
use frida;
grant delete on gastenboek to cursist
```

### 2.32.2 GastenBoekRepository

```
public void delete(Long[] ids) {
    // de JdbcTemplate update method verwacht een Long[] array, geen Long[] array
    if (ids.length != 0) {
        var sql = ""
            delete from gastenboek
            where id in (
                ""
                + "?,".repeat(ids.length - 1)
                + "?)";
        template.update(sql, ids);
    }
}
```

### 2.32.3 GastenBoekRepositoryTest

```
private long idVanTestEntry() {
    return jdbcTemplate.queryForObject(
        "select id from gastenboek where naam='test'", Long.class);
}
private long idVanTest2Entry() {
    return jdbcTemplate.queryForObject(
        "select id from gastenboek where naam='test2'", Long.class);
}
@Test
void delete() {
    var id = idVanTestEntry();
    var id2 = idVanTest2Entry();
    repository.delete(new Long[]{id, id2});
    assertThat(countRowsInTableWhere(GASTENBOEK,
        "id in (" + id + ',' + id2 + ')'))
        .isZero();
}
@Test
void deleteMetLegeArrayGeeftGeenException() {
    repository.delete(new Long[]{});
}
```

### 2.32.4 GastenBoekService

```
public void delete(Long[] ids) {
    gastenBoekRepository.delete(ids);
}
```

### 2.32.5 GastenBoekController

```
@PostMapping("verwijderen")
public String delete(Optional<Long[]> id) {
    id.ifPresent(ids -> gastenBoekService.delete(ids));
    // als de gebruikt geen enkele entry selecteerde, is id een lege Optional
    return "redirect:/gastenboek";
}
```

### 2.32.6 gastenboek.html

Vervang <ul>...</ul> door:

```
<form th:if="not ${gastenboek.empty}" th:action="@{/gastenboek/verwijderen}"
  method="post">
  <ul>
    <li th:each="entry : ${gastenboek}" th:object="${entry}">
      <strong th:text="|*{{datum}} *{{naam}}|"></strong>
      <th:block th:text="*{{bericht}}"></th:block>
      <input type="checkbox" name="id" th:value="*{{id}}">
    </li>
  </ul>
  <button>Verwijderen</button>
</form>
```

### 2.32.7 frida.css

```
input[type=checkbox] {
  display: inline;
}
```

## 2.33 Star Trek

### 2.33.1 application.properties

```
spring.datasource.url=jdbc:mysql://localhost/startrek
spring.datasource.username=cursist
spring.datasource.password=cursist
logging.level.org.springframework.jdbc.core.JdbcTemplate=DEBUG
logging.level.org.springframework.jdbc.core.simple.SimpleJdbcInsert=DEBUG
logging.level.org.springframework.jdbc.core.StatementCreatorUtils=TRACE
spring.test.database.replace=none
spring.datasource.hikari.transaction-isolation=TRANSACTION_READ_COMMITTED
```

### 2.33.2 spring.properties

```
spring.test.constructor.autowire.mode=all
```

### 2.33.3 DataSourceTest

```
package be.vdab.budget.repositories;
// enkele imports
@JdbcTest class DataSourceTest {
  private final DataSource dataSource; // en constructor met parameter
  @Test void getConnection() throws SQLException {
    try (var connection = dataSource.getConnection()) {
      assertThat(connection.getCatalog()).isEqualTo("startrek");
    }
  }
}
```

### 2.33.4 Werknemer

```
package be.vdab.budget.domain;
// enkele imports
public class Werknemer {
  private final long id;
  private final String voornaam;
  private final String familienaam;
  @NumberFormat(pattern = "#,##0.00")
  private final BigDecimal budget;
  // constructor met parameters
  public String getNaam() {
    return voornaam + ' ' + familienaam;
  }
  // getters
}
```

### 2.33.5 Bestelling

```
package be.vdab.budget.domain;
// enkele imports
public class Bestelling {
    private final long id;
    private final long werknemerId;
    @NotBlank
    private final String omschrijving;
    @NotNull
    @Positive
    @NumberFormat(pattern = "#,##0.00")
    private final BigDecimal bedrag;
    // constructor met parameters
    // getters
}
```

### 2.33.6 OnvoldoendeBudgetException

```
package be.vdab.budget.exceptions;
public class OnvoldoendeBudgetException extends RuntimeException {
    private static final long serialVersionUID = 1L;
}
```

### 2.33.7 WerknemerNietGevondenException

```
package be.vdab.budget.exceptions;
public class WerknemerNietGevondenException extends RuntimeException {
    private static final long serialVersionUID = 1L;
}
```

### 2.33.8 WerknemerRepository

```
package be.vdab.budget.repositories;
// enkele imports
@Repository
public class WerknemerRepository {
    private final JdbcTemplate template;
    private final RowMapper<Werknemer> werknemerMapper =
        (result, rowNum) ->
            new Werknemer(result.getLong("id"), result.getString("voornaam"),
                result.getString("familienaam"), result.getBigDecimal("budget"));
    // constructor met parameter
    public List<Werknemer> findAll() {
        var sql = """
            select id, voornaam, familienaam, budget
            from werknemers
            order by voornaam
            """;
        return template.query(sql, werknemerMapper);
    }
    public Optional<Werknemer> findById(long id) {
        try {
            var sql = """
                select id, voornaam, familienaam, budget from werknemers
                where id = ?
                """;
            return Optional.of(
                template.queryForObject(sql, werknemerMapper, id));
        } catch (IncorrectResultSizeDataAccessException ex) {
            return Optional.empty();
        }
    }
}
```



```

public void verlaagBudget(long id, BigDecimal bedrag) {
    var sql = """
        update werknemers set budget = budget - ?
        where id = ? and budget >= ?
        """;
    var aantalAangepasteRecords =
        template.update(sql, bedrag, id, bedrag);
    if (aantalAangepasteRecords == 0) {
        if (findById(id).isEmpty()) {
            throw new WerknemerNietGevondenException();
        } else {
            throw new OnvoldoendeBudgetException();
        }
    }
}
}

```

### 2.33.9 insertWerknemers.sql

```

insert into werknemers(voornaam, familienaam, budget) values
('test1voornaam', 'test1familienaam', 1000),
('test2voornaam', 'test2familienaam', 2000);

```

### 2.33.10 WerknemerRepositoryTest

```

package be.vdab.budget.repositories;
// enkele imports
@JdbcTest
@Import(WerknemerRepository.class)
@Sql("/insertWerknemers.sql")
class WerknemerRepositoryTest
    extends AbstractTransactionalJUnit4SpringContextTests {
    private static final String WERKNEMERS = "werknemers";
    private final WerknemerRepository repository;
    // constructor met parameter
    @Test
    void findAllGeeftAlleWerknemersGesorteerdOpVoornaam() {
        assertThat(
            repository.findAll()
                .hasSize(countRowsInTable(WERKNEMERS))
                .extracting(Werknemer::getVoornaam)
                .isSortedAccordingTo(String::compareToIgnoreCase);
        );
    }
    private long idVanTestWerknemer() {
        return jdbcTemplate.queryForObject(
            "select id from werknemers where voornaam='test1Voornaam'", Long.class);
    }
}

```

```

@Test
void findById() {
    assertThat(repository.findById(idVanTestWerknemer()))
        .hasValueSatisfying(werknemer ->
            assertThat(werknemer.getNaam())
                .isEqualTo("test1voornaam test1familienaam"));
}
@Test
void findByIdOnBestaandeIdVindtGeenWerknemer() {
    assertThat(repository.findById(-1)).isEmpty();
}
@Test
void verlaagBudgetMetVoldoendeBudget() {
    repository.verlaagBudget(idVanTestWerknemer(), BigDecimal.TEN);
}
@Test
void verlaagBudgetMetOnvoldoendeBudget() {
    assertThatExceptionOfType(OnvoldoendeBudgetException.class)
        .isThrownBy(() -> repository.verlaagBudget(idVanTestWerknemer(),
            BigDecimal.valueOf(100_000)));
}
@Test
void verlaagBudgetVanOnbestaandeWerknemer() {
    assertThatExceptionOfType(WerknemerNietGevondenException.class)
        .isThrownBy(() -> repository.verlaagBudget(-1, BigDecimal.TEN));
}
}

```

### 2.33.11 BestellingRepository

```

package be.vdab.budget.repositories;
// enkele imports
@Repository
public class BestellingRepository {
    private final JdbcTemplate template;
    private final SimpleJdbcInsert insert;
    private final RowMapper<Bestelling> bestellingMapper =
        (result, rowNum) ->
            new Bestelling(result.getLong("id"), result.getLong("werknemerId"),
                result.getString("omschrijving"), result.getBigDecimal("bedrag"));
    public BestellingRepository(JdbcTemplate template) {
        this.template = template;
        insert = new SimpleJdbcInsert(template)
            .withTableName("bestellingen");
    }
    public List<Bestelling> findByIdByWerknemerId(long id) {
        var sql = """
            select id, werknemerId, omschrijving, bedrag
            from bestellingen
            where werknemerId = ?
            order by id
            """;
        return template.query(sql, bestellingMapper, id);
    }
    public void create(Bestelling bestelling) {
        insert.execute(Map.of(
            "werknemerId", bestelling.getWerknemerId(),
            "omschrijving", bestelling.getOmschrijving(),
            "bedrag", bestelling.getBedrag()));
    }
}

```

**2.33.12 insertBestellingen.sql**

```
insert into bestellingen(werknemerId, omschrijving, bedrag) values
((select id from werknemers where voornaam='test1voornaam'),
'test1omschrijving', 100),
((select id from werknemers where voornaam='test1voornaam'),
'test2omschrijving', 200);
```

**2.33.13 BestellingRepositoryTest**

```
package be.vdab.budget.repositories;
// enkele imports
@JdbcTest
@Import(BestellingRepository.class)
@Sql({"insertWerknemers.sql", "insertBestellingen.sql"})
class BestellingRepositoryTest
    extends AbstractTransactionalJUnit4SpringContextTests {
    private static final String BESTELLINGEN = "bestellingen";
    private final BestellingRepository repository;
    // constructor met parameter
    private long idVanTestWerknemer() {
        return jdbcTemplate.queryForObject(
            "select id from werknemers where voornaam='test1Voornaam'",
            Long.class);
    }
    @Test
    void findByWerknemerIdGeeftAlleWerknemersGesorteerdOpVoornaam() {
        var werknemerId = idVanTestWerknemer();
        assertThat(repository.findByWerknemerId(werknemerId))
            .hasSize(countRowsInTableWhere(BESTELLINGEN, "werknemerId=" + werknemerId))
            .allSatisfy(bestelling ->
                assertThat(bestelling.getWerknemerId()).isEqualTo(werknemerId))
            .extracting(Bestelling::getId)
            .isSorted();
    }
    @Test
    void findByOnbestaandeWerknemerIdGeeftEenLegeLijst() {
        assertThat(repository.findByWerknemerId(-1L)).isEmpty();
    }
}
```

**2.33.14 tWerknemerService**

```
package be.vdab.budget.services;
// enkele imports
@Service
@Transactional(readOnly = true)
public class WerknemerService {
    private final WerknemerRepository werknemerRepository;
    // constructor met parameter
    public List<Werknemer> findAll() {
        return werknemerRepository.findAll();
    }
    public Optional<Werknemer> findById(long id) {
        return werknemerRepository.findById(id);
    }
}
```

### 2.33.15 tBestellingService

```
package be.vdab.budget.services;
// enkele imports
@Service
@Transactional(readOnly = true)
public class BestellingService {
    private final BestellingRepository bestellingRepository;
    private final WerknemerRepository werknemerRepository;
    // constructor met parameters
    public List<Bestelling> findByWerknemerId(long id) {
        return bestellingRepository.findByWerknemerId(id);
    }
    @Transactional
    public void bestel(Bestelling bestelling) {
        // volgende opdracht wijzigt het werknemer record.
        // dit record blijft gelocked tot het einde van de transactie
        // zo kan een andere gebruiker niet tegelijk voor deze werknemer bestellen
        werknemerRepository.verlaagBudget(
            bestelling.getWerknemerId(), bestelling.getBedrag());
        // volgende opdracht voegt de bestelling toe:
        bestellingRepository.create(bestelling);
    }
}
```

### 2.33.16 BestellingServiceTest

Als een service method maar één opdracht bevat, heeft een test van zo'n method weinig meerwaarde. De method bestel bevat echter *meerdere* opdrachten. De bijbehorende test:

```
package be.vdab.startrek.services;
import static org.mockito.Mockito.verify;
// enkele andere imports
@ExtendWith(MockitoExtension.class)
class BestellingServiceTest {
    private BestellingService service;
    @Mock
    BestellingRepository bestellingRepository;
    @Mock
    WerknemerRepository werknemerRepository;
    @BeforeEach
    void beforeEach() {
        service = new BestellingService(bestellingRepository, werknemerRepository);
    }
    @Test
    void bestel() {
        var bestelling = new Bestelling(0, 1, "test", BigDecimal.ONE);
        service.bestel(bestelling);
        // riep de service method de correcte BestellingRepository method op:
        verify(bestellingRepository).create(bestelling);
        // riep de service method de correcte WerknemerRepository method op:
        verify(werknemerRepository).verlaagBudget(1, BigDecimal.ONE);
    }
}
```

### 2.33.17 IndexController

```
package be.vdab.budget.controllers;
// enkele imports
@Controller
@RequestMapping("/")
class IndexController {
    private final WerknemerService werknemerService;
    // constructor met parameters
    @GetMapping
    public ModelAndView index() {
        return new ModelAndView("index",
            "werknemers", werknemerService.findAll());
    }
}
```

### 2.33.18 WerknemerController

```
package be.vdab.budget.controllers;
// enkele imports
@Controller
@RequestMapping("werknemers")
class WerknemerController {
    private final WerknemerService werknemerService;
    private final BestellingService bestellingService;
    // constructor met parameters
    @GetMapping("{id}")
    public ModelAndView findById(@PathVariable long id) {
        var modelAndView = new ModelAndView("werknemer");
        werknemerService.findById(id).ifPresent(werknemer ->
            modelAndView.addObject(werknemer));
        return modelAndView;
    }
    @GetMapping("{id}/vorigebestellingen")
    public ModelAndView findVorigeBestellingen(@PathVariable long id) {
        var modelAndView = new ModelAndView("vorigeBestellingen");
        werknemerService.findById(id).ifPresent(werknemer ->
            modelAndView.addObject(werknemer)
                .addObject("bestellingen",
                    bestellingService.findByWerknemerId(id)));
        return modelAndView;
    }
    @GetMapping("{id}/nieuwebestelling")
    public ModelAndView nieuweBestelling(@PathVariable long id) {
        var modelAndView = new ModelAndView("nieuweBestelling");
        werknemerService.findById(id).ifPresent(werknemer ->
            modelAndView.addObject(werknemer)
                .addObject(new Bestelling(0, id, null, null)));
        return modelAndView;
    }
}
```

```

@PostMapping("bestel")
public ModelAndView bestel(@Valid Bestelling bestelling, Errors errors,
    RedirectAttributes redirect) {
    if (errors.hasErrors()) {
        var modelAndView = new ModelAndView("nieuweBestelling");
        werknemerService.findById(bestelling.getWerknemerId())
            .ifPresent(werknemer ->
                modelAndView.addObject(werknemer));
        return modelAndView;
    }
    try {
        bestellingService.bestel(bestelling);
        redirect.addAttribute("id", bestelling.getWerknemerId());
        return new ModelAndView(
            "redirect:/werknemers/{id}/vorigebestellingen");
    } catch (OnvoldoendeBudgetException ex) {
        redirect.addAttribute("onvoldoendeBudget", true);
        redirect.addAttribute("id", bestelling.getWerknemerId());
        return new ModelAndView("redirect:/werknemers/{id}");
    } catch (WerknemerNietGevondenException ex) {
        redirect.addAttribute("werknemerNietGevonden", true);
        return new ModelAndView("redirect:/");
    }
}
}

```

### 2.33.19 fragments.html

```

<head th:fragment="head(title)" xmlns:th="http://www.thymeleaf.org">
    <link rel="icon" th:href="@{/images/startrek.ico}" type="image/x-icon">
    <title th:text="${title}"></title>
    <link rel="stylesheet" th:href="@{/css/startrek.css}">
</head>

```

### 2.33.20 index.html

```

<!doctype html>
<html lang="nl" xmlns:th="http://www.thymeleaf.org">
<head th:replace="fragments::head(title='Werknemers')"></head>
<body>
<div th:if="${param.werknemerNietGevonden}" class="fout">
    Werknemer niet gevonden.
</div>
<h1>Werknemers</h1>
<ul>
    <li th:each="werknemer : ${werknemers}" th:object="${werknemer}">
        <a th:text="*{naam}" th:href="@{/werknemers/{id}(id=*{id})}"></a>
    </li>
</ul>
</body>
</html>

```

### 2.33.21 werknemer.html

```

<!doctype html>
<html lang="nl" xmlns:th="http://www.thymeleaf.org">
<head
    th:replace="fragments::head(title=${werknemer}?${werknemer.naam}:
        'Werknemer niet gevonden')"></head>
<body>
<div th:if="${param.onvoldoendeBudget}" class="fout">
    Onvoldoende budget.
</div>

```

```

<h1 th:if="not ${werknemer}" class="fout">
  Werknemer niet gevonden: <span th:text="${id}"></span>
</h1>
<th:block th:if="${werknemer}" th:object="${werknemer}">
  <h1 th:text="*{naam}"></h1>
  
  <dl>
    <dt>Nummer</dt><dd th:text="*{id}"></dd>
    <dt>Budget</dt><dd th:text="*{budget}"></dd>
  </dl>
  <a th:href="@{/werknemers/{id}/vorigebestellingen(id=*{id})}">Vorige
    bestellingen</a>
  <a th:href="@{/werknemers/{id}/nieuwebestelling(id=*{id})}">Nieuwe
    bestelling</a>
</th:block>
<a th:href="@{/}">Startpagina</a>
</body>
</html>

```

### 2.33.22 vorigeBestellingen.html

```

<!doctype html>
<html lang="nl" xmlns:th="http://www.thymeleaf.org">
<head th:replace="fragments::head(title=${werknemer}? 'Bestellingen van
  '+${werknemer.naam}: 'Werknemer niet gevonden')"></head>
<body>
<h1 th:if="not ${werknemer}" class="fout">
  Werknemer niet gevonden: <span th:text="${id}"></span>
</h1>
<th:block th:if="${werknemer}" th:object="${werknemer}">
  <h1 th:text="|Bestellingen van *{naam}|"></h1>
  <table id="bestellingen">
    <thead>
      <tr>
        <th>Nummer</th>
        <th>Omschrijving</th>
        <th>Bedrag</th>
      </tr>
    </thead>
    <tbody>
      <tr th:each="bestelling: ${bestellingen}" th:object="${bestelling}">
        <td th:text="*{id}"></td>
        <td th:text="*{omschrijving}"></td>
        <td th:text="*{{bedrag}}"></td>
      </tr>
    </tbody>
  </table>
</th:block>
<a th:href="@{/}">Startpagina</a>
</body>
</html>

```

### 2.33.23 nieuweBestelling.html

```

<!doctype html>
<html lang="nl" xmlns:th="http://www.thymeleaf.org">
<head th:replace="fragments::head(title='Nieuwe bestelling')"></head>
<body>
<h1 th:if="not ${werknemer}" class="fout">
  Werknemer niet gevonden: <span th:text="${id}"></span>
</h1>

```

```

<th:block th:if="${werknemer}">
  <h1 th:text="|Bestelling voor ${werknemer.naam}|"></h1>
  <form th:object="${bestelling}" method="post"
        th:action="@{/werknemers/bestel}">
    <input th:field="*{id}" type="hidden">
    <input th:field="*{werknemerId}" type="hidden">
    <label>Omschrijving
      <span th:errors="*{omschrijving}"></span>
      <input th:field="*{omschrijving}" required autofocus>
    </label>
    <label>Bedrag
      <span th:errors="*{bedrag}"></span>
      <input th:field="*{bedrag}" required type="number" min="0.01" step="0.01">
    </label>
    <button>Bestel</button>
  </form>
  <a th:href="@{/}">Startpagina</a>
</th:block>
</body>
</html>

```

### 2.33.24 messages.properties

Voeg de foutboodschappen toe die horen bij bean validation.

### 2.33.25 startrek.css

```

body {
  background-color: #FFFFDE;
  font-family: sans-serif;
}
dd, dt {
  margin-left: 0;
}
dt {
  margin-top: 0.5em;
}
dd {
  font-weight: bold;
}
label span, .fout {
  background-color: red;
  color: white;
  font-weight: bold;
  padding-left: 0.5em;
  padding-right: 0.5em;
}
label span {
  margin-left: 0.5em;
}
label {
  cursor: pointer;
}
input, button {
  display: block;
  font-size: 1.1em;
  margin-bottom: 1em;
  margin-top: 0.2em;
}
input:focus {
  background-color: #FFFFD6
}

```



```
button {
  background-color: #4CAF50;
  border: none;
  color: white;
  cursor: pointer;
  font-weight: bold;
  outline: 0;
  padding: 8px 16px;
}
button:hover {
  background-color: #3e8e41;
}
button:active {
  background-color: #3ecc41;
}
button:disabled {
  background-color: gray;
  cursor: not-allowed;
}
table, td, th {
  border: 1px solid silver;
  border-collapse: collapse;
  padding: 5px 5px;
}
th {
  color: white;
  background-color: #4CAF50;
}
td {
  padding-left: 0.2em;
  padding-right: 0.2em;
}
tr:nth-child(even) {
  background-color: #FAFABA;
}
tr:nth-child(odd) {
  background-color: #FAFAFA;
}
tr:hover {
  background-color: gold;
}
#bestellingen td:nth-child(1), #bestellingen td:nth-child(3) {
  text-align: right;
}
```

### 3 COLOFON

<b>Domeinexpertisemanager:</b>	Jean Smits
<b>Moduleverantwoordelijke:</b>	Jean Smits
<b>Medewerkers:</b>	Hans Desmet
<b>Versie:</b>	11/2/2022
<b>Nummer dotatielijst:</b>	