VS Code: Spring boot projektin luominen

9.1.2025, Minna Pellikka

Spring boot projektin luominen

Tässä lyhyesti ohjeet**, seuraavalla sivulla tarkemmat ohjeet** näyttökuvien kera.

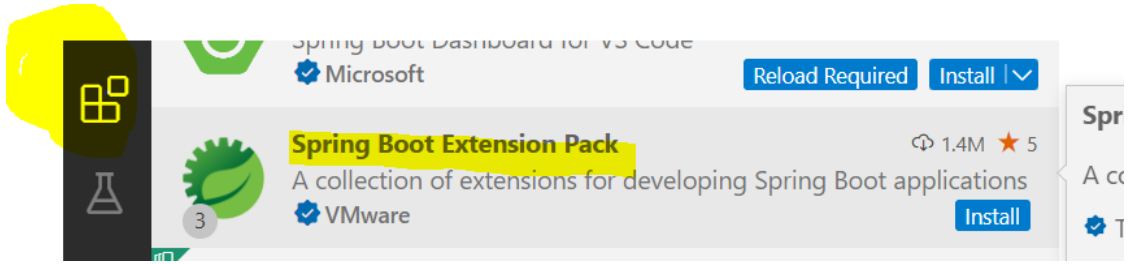
1. Spring boot projekti luodaan suoraan Visual Studio Codessa. Sinulla pitää olla kaksi laajennosta (extensions) asennettuna: Spring Boot Extension Pack sekä Extension Pack for Java.

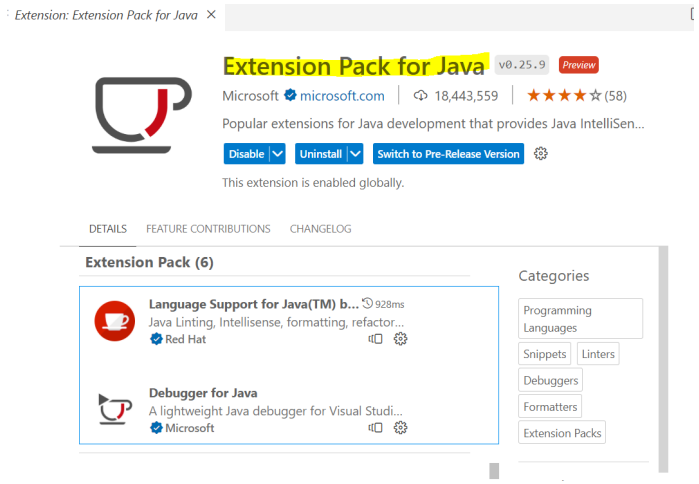
2. Projektin tietojen syöttäminen:
    a) Mene kansioon, jonne haluat projektin luoda
    b) ctrl + shift + p näppäinyhdistelmällä saat vs code ruudun yläreunaan command valikon -> kirjoita sinne ”Spring initialzer: Create a Maven project”
    c) Valitse SB versio on 3.4.1
    d) Valitse tämän jälkeen kieleksi Java
    e) Anna projektillesi group id (paketti, package), esim. kevat25
    f) Anna artifact (projektin nimi), esim. backend
    g) Valitse paketointityypiksi JAR
    h) Javan versio on 17
    i) Valitse dependecyt (riippuvuudet, kirjastopaketit) ja hyväksy ne enterillä
    j) Hyväksy projektin generointi kansioon, jonka valitsit a-kohdassa

3. Koodin lisääminen projektiin ja sovelluksen käynnistäminen

1 Käynnistä Visual Studio Code ja asenna tarvittavat laajennokset

Valitse kuvien mukaiset laajennokset (extensios) Spring Boot projektia varten





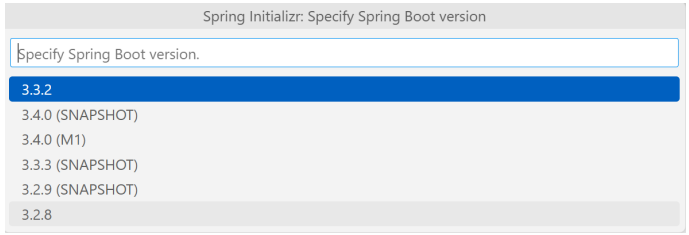
2 Syötä projektin tiedot

Projektin tietojen syöttäminen:

a) Mene kansioon, jonne haluat projektin luoda (VS Code: open folder)

b) ctrl + shift + p näppäinyhdistelmällä saat vs code ruudun yläreunaan command valikon -> kirjoita sinne ”Spring initialzer: Create a Maven project”

c) Valitse opettajan ohjeistama Spring boot versio (esim. keväällä 2025 se on 3.4.1)



d) Valitse tämän jälkeen kieleksi Java

e) Anna projektillesi group id (paketti, package), esim. kevat25

f) Anna artifact (projektin nimi), esim. backend

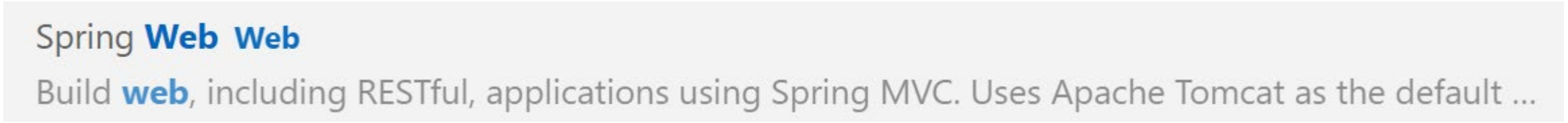
g) Valitse paketointityypiksi JAR

h) Javan versio on 17

2 Syötä projektin tiedot, jatkuu…

i) Valitse oikeat dependecyt (Spring web, Spring Boot Dev Tools)

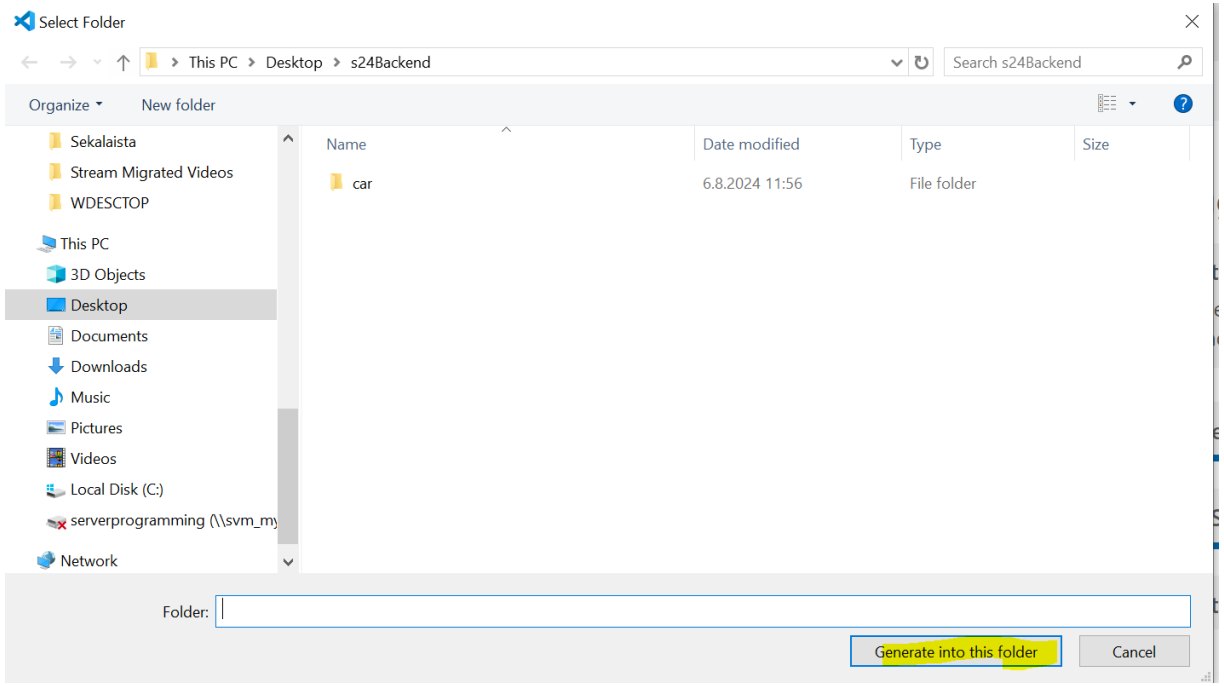
Paketin avulla voimme luoda webbisovelluksia, käyttää MVC-arkkitehtuuria, sisältää sisäisen Tomcatin



Paketin avulla voimme esim. tehdä koodiin muunnoksia ilman, että meidän pitää käynnistää tomcat palvelin uudelleen. Muunnokset ovat siten heti ajonaikana käytettävissä.

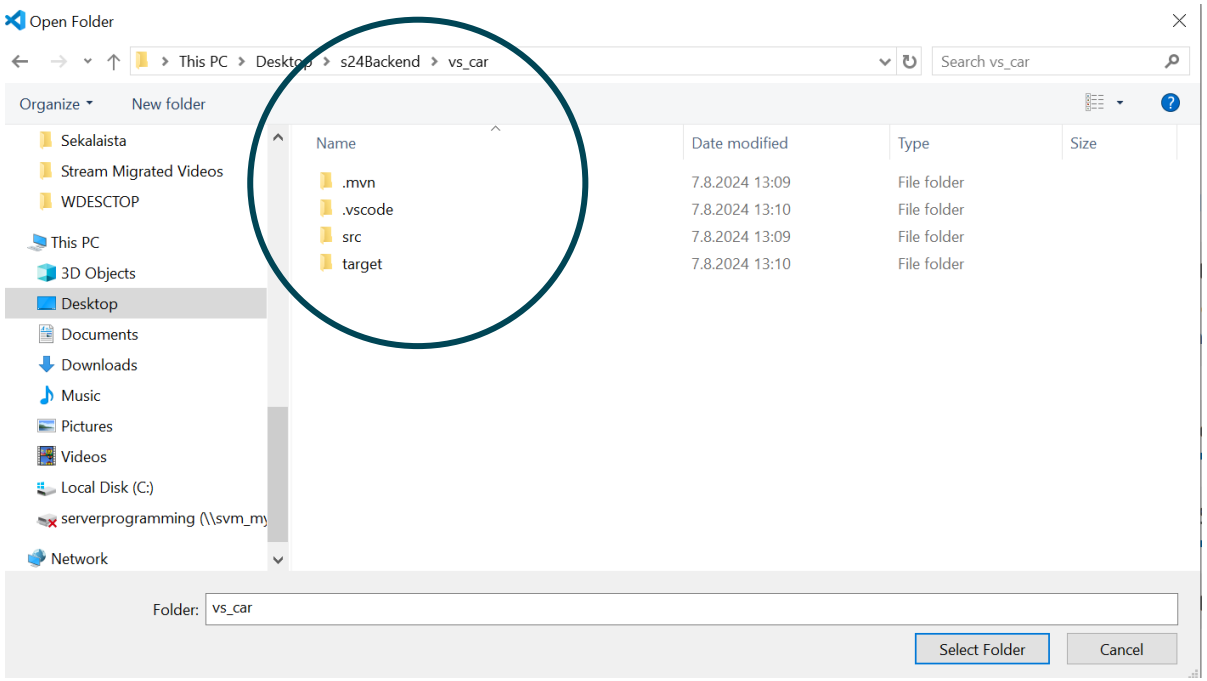
2 Syötä projektin tiedot, jatkuu…

j) Generoi projekti a-kohdassa valittuun kansioon



3 Koodin lisääminen projektiisi

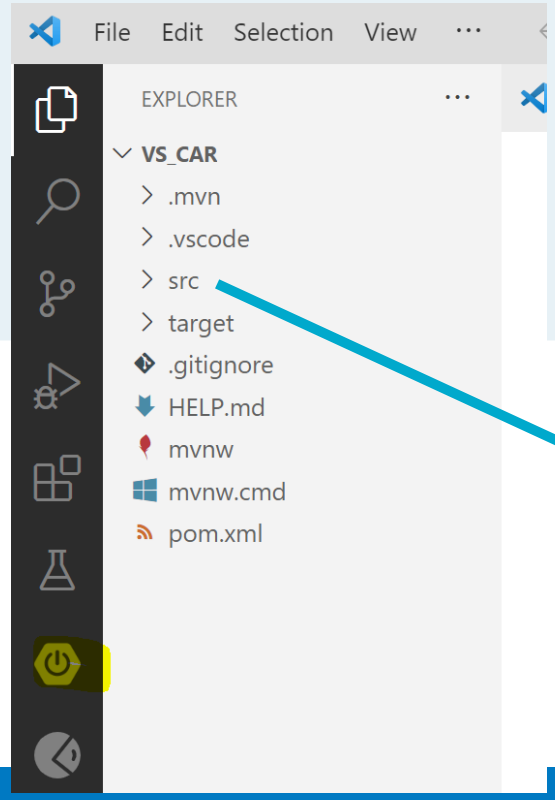
VS Code ruudulla näkyy uusi projekti. Jos ei näy, niin avaa kansio seuraavasti: VS Code -> File -> Open Folder)

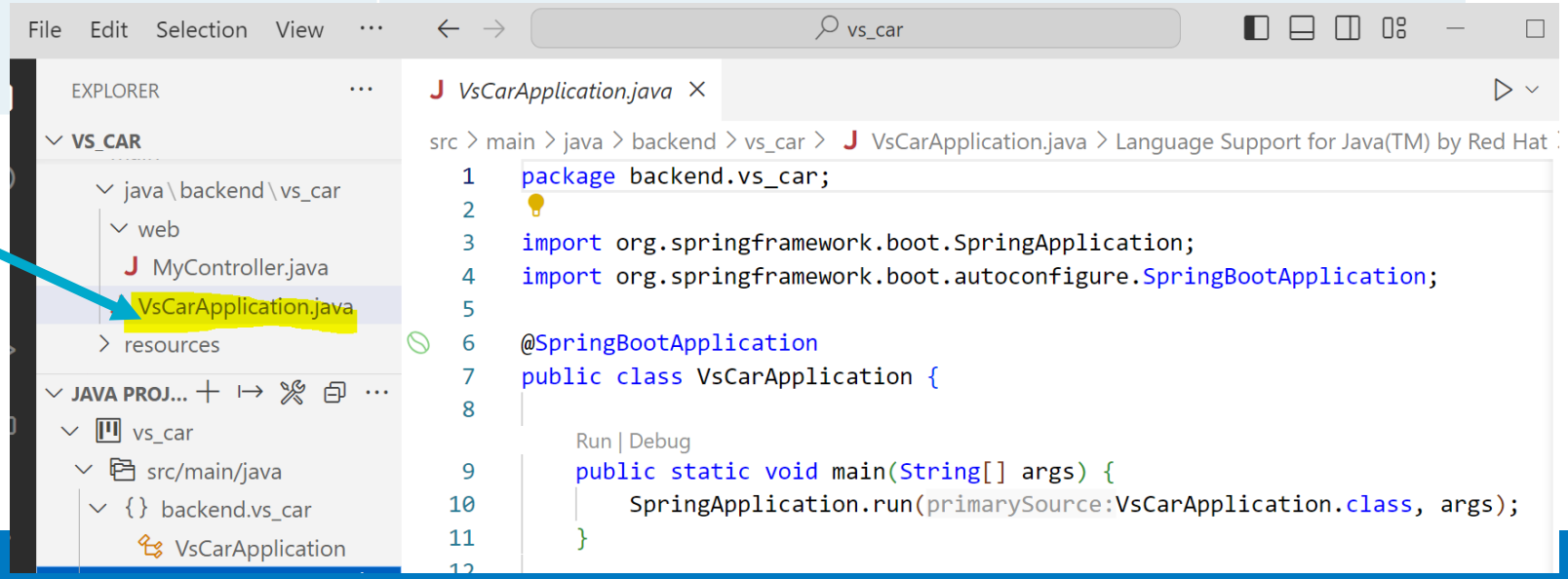


3 Koodin lisääminen projektiisi, jatkuu

Ruudulla näkyy nyt Backend projektisi pohja, johon voit koodata toiminnallisuutta

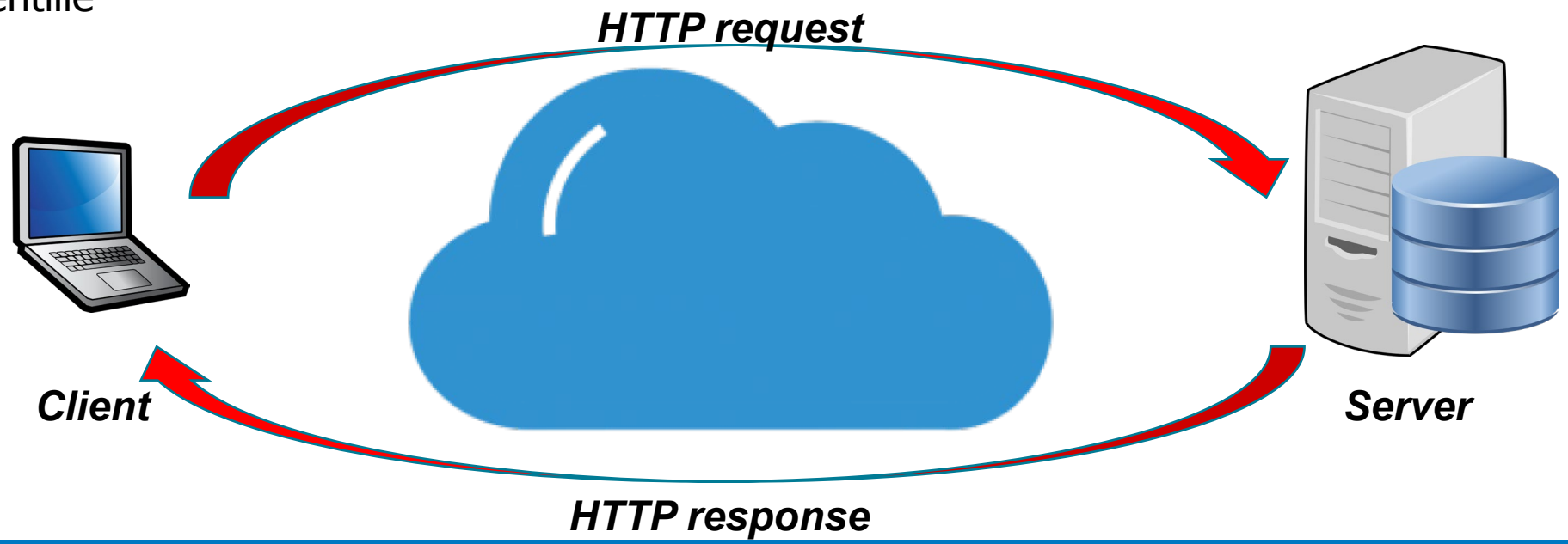
Ns pääluokka on generoitunut automaattisesti Spring Boot kehikon toimesta. Annotaatio @SpringBootApplication varmistaa mm. sen, että sovellus lukee projektin kaikki kontrollerit ja palvelut. Huolehtii, että ”lennossa” lisätyt paketit (esim. validation) toimivat ilman sovelluksen uudelleen käynnistämistä





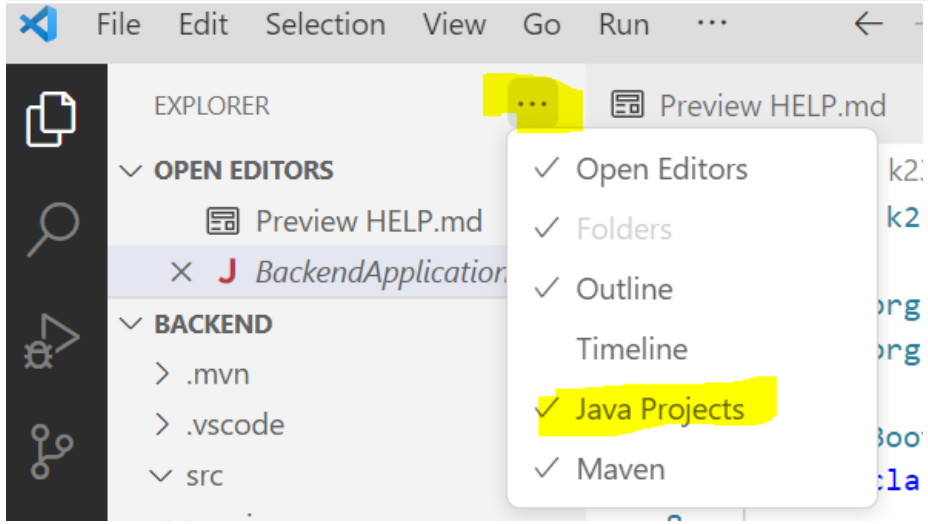
Sovelluksen toiminnallisuuden periaatteet

- Tällä hetkellä meillä on sovellus, jossa on runko valmiina, muttei toiminnallisuutta.
- Lisätään toiminnallisuus, jonka tuloksena nettisivulle tulostuu lause ”Spring Boot sovellukseni!”
- Luodaan tätä varten palvelinohjelmaamme ns. controller luokka. Controller luokka vastaanottaa palvelimeen (server) kohdistuvat asiakkaan (client) pyynnöt. Controller luokka tekee – tai mieluummin pyytää jotain muuta tahoa tekemään työt ja palauttaa palautteen (response) clientille

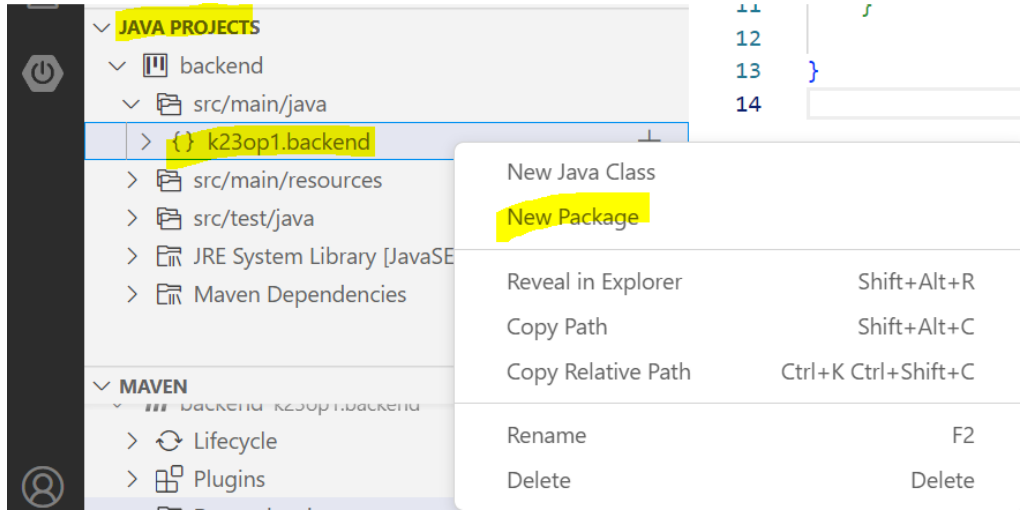


3 Koodin lisääminen projektiisi, jatkuu

a) Tarkista, että olet aktivoinut Java Projects näkymän

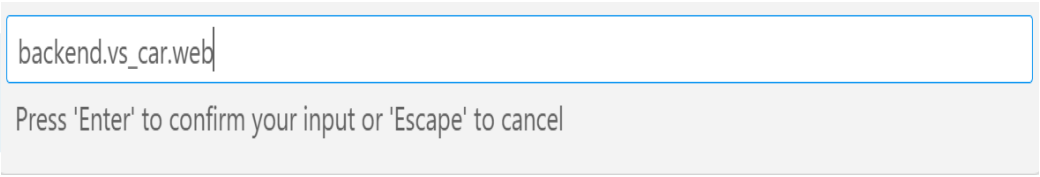


b) Luo uusi paketti ao kuvan mukaisesti



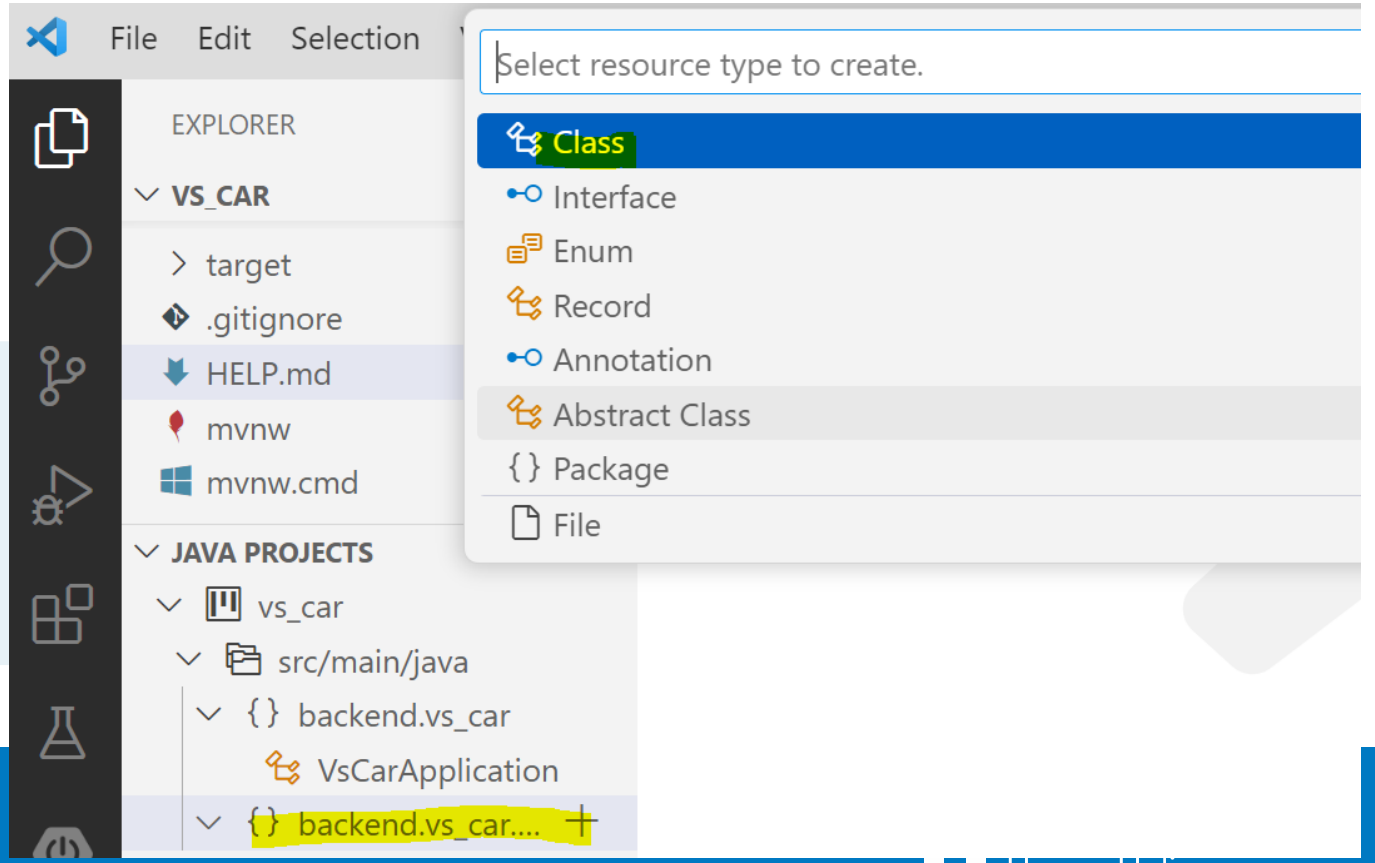
3 Koodin lisääminen projektiisi, jatkuu

c) Anna paketille nimeksi web, laitamme sinne ns controller luokkia



d) Luo uusi java luokka. Anna
Luokalle nimeksi **`MyController.java`**.

Luokkaan lisättävä koodi esitetään seuraavalla sivulla.



3 Koodin lisääminen projektiisi, jatkuu

MyController.java

_car > web > MyController.java > Language Support for Java(TM) by Red Hat > MyContro

```
1   package backend.vs_car.web;
2
3   import org.springframework.stereotype.Controller;
4   import org.springframework.web.bind.annotation.RequestMapping;
5   import org.springframework.web.bind.annotation.ResponseBody;
6
7   @Controller
8   public class MyController {
9
10      @RequestMapping("/main")
11      @ResponseBody
12      public String returnMessage() {
13          return "Eka SB sovellukseni";
14      }
15
16  }
```

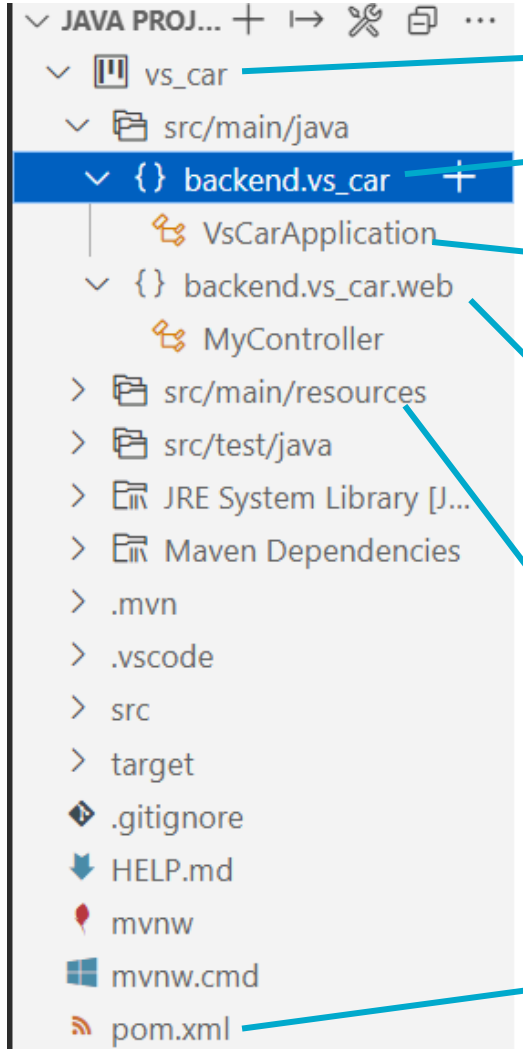
Spring Bootissa kerrotaan @ annotaatioilla seuraavat asiat

`@Controller:` Tämän ns. kontrolleriluokan metodit pystyvät käsittelemään clientin pyynnöt/requestit ja välittämään niille vastaukset/responset.

`@RequestMapping:` Kerrotaan, mistä osoitteesta tulevat pyynnöt käsitellään kyseisessä metodissa. Tässä esimerkissä käytetään ”main” merkkijonoa. Joten, jos kirjoitat selaimeen urliin ” http://localhost:8080/main”, niin kutsu ohjautuu tähän **`returnMessage`**-metodiin.

`@ResponseBody:` Palaute clientille esitetään sellaisenaan eli esimerkissä ”Eka SB sovellukseni”.

Ensimmäisen projektisi rakenne



Projektisi nimi (artifact)

Paketin-nimi (group)

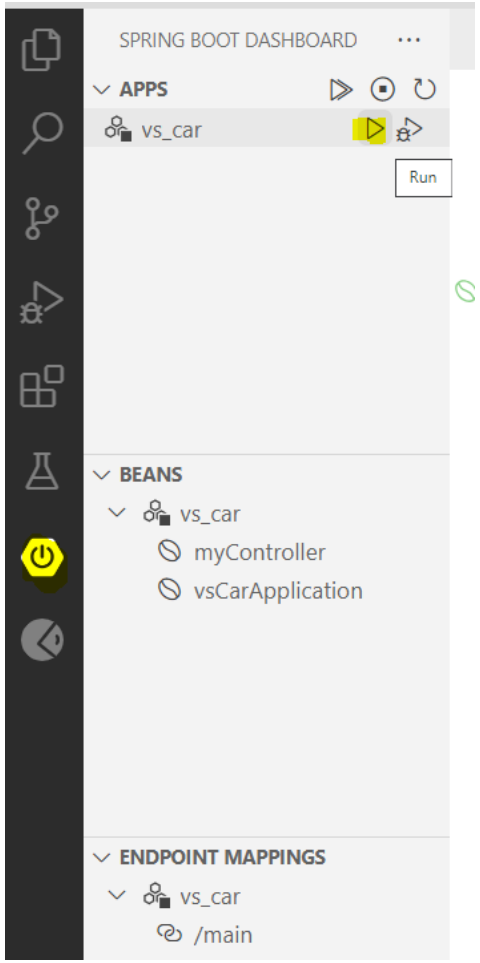
Spring Boot projektin pääluokka, joka sisältää main-metodin. Main-metodin avulla käynnistetään sovellus. Pääluokan pitää olla ”juuressa”. Muut paketit luokkineen sen alapuolella. Kiinnitä rakenteeseen huomiota, muuten sovellus ei toimi kunnolla.

Tämän paketin alla sijaitsevat Controller–luokat, jotka vastaanottavat clientin requestit

Resources kansion alla on Templates-hakemisto, jonne laitetaan View-tason tiedostot. Me käytämme käyttöliittymän rakentamisessa Thymeleafia.

Mavenin konfigurointitiedosto.
Sisältää kirjastomoduulit, dependency.

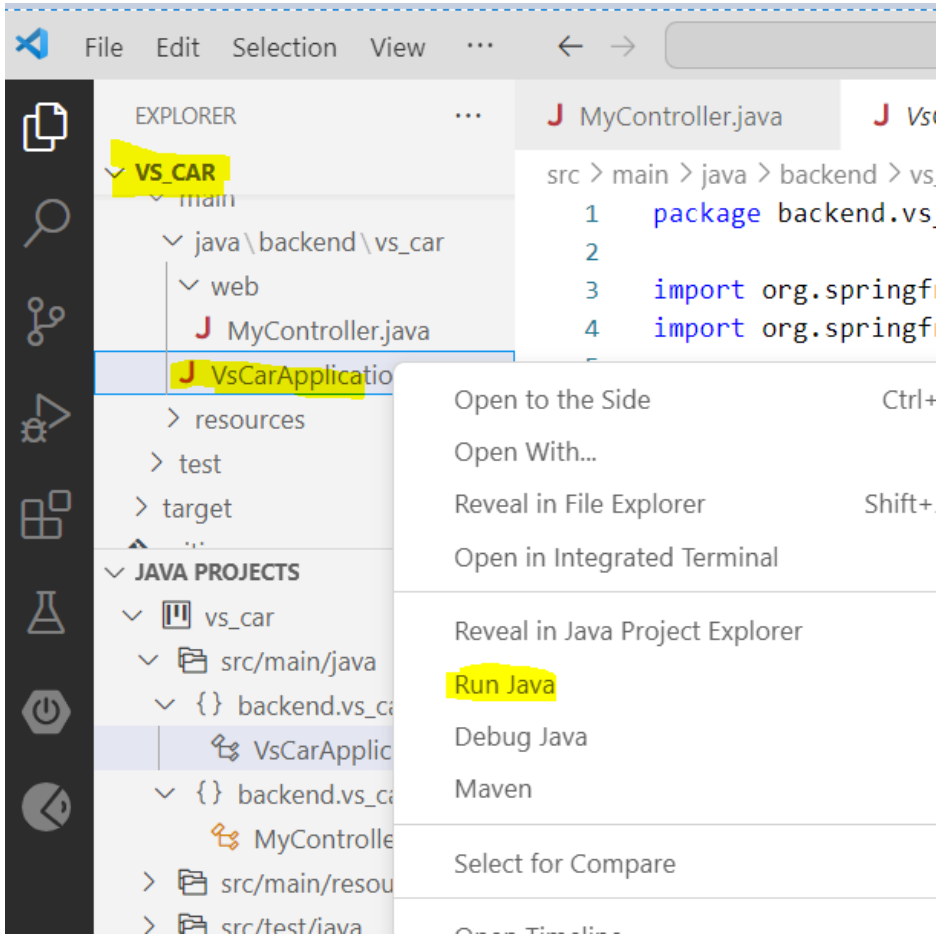
Sovelluksen käynnistäminen



Spring Boot sovelluksen käynnistäminen voi tapahtua esim. vasemmalla olevan kuvan mukaisessa näkymässä. Ks keltaisella väritetyt.

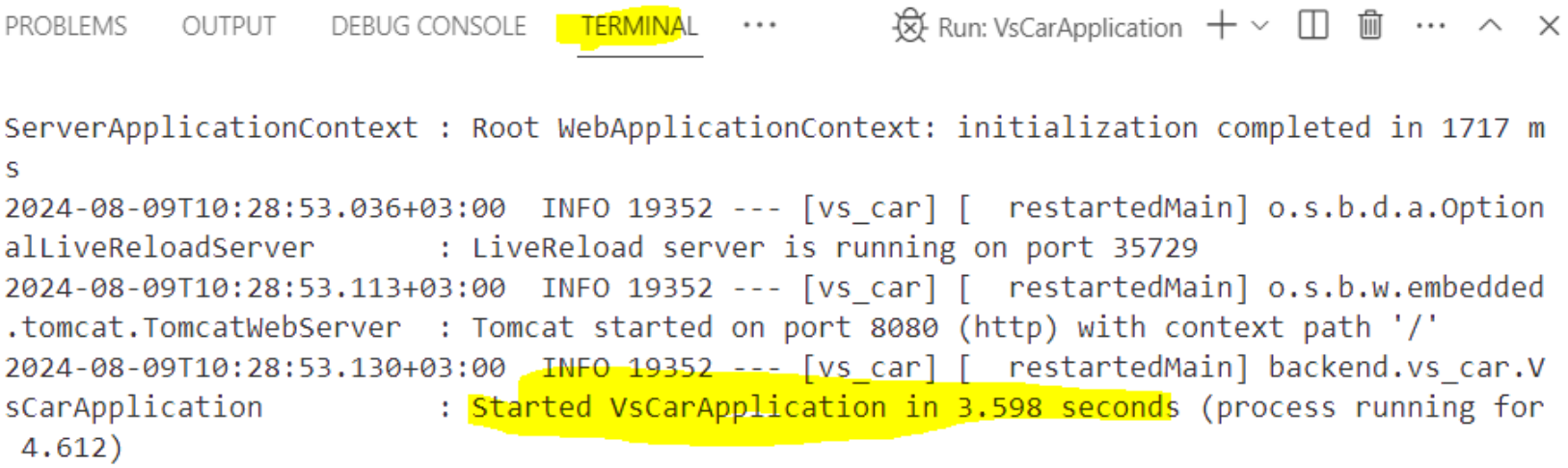
TAI

Sovelluksen voi käynnistää myös menemällä `Explorer` näkymässä projektin pääluokan päälle ja valitsemalla pop-up valikosta Run Java. Ks kuva oikealla.

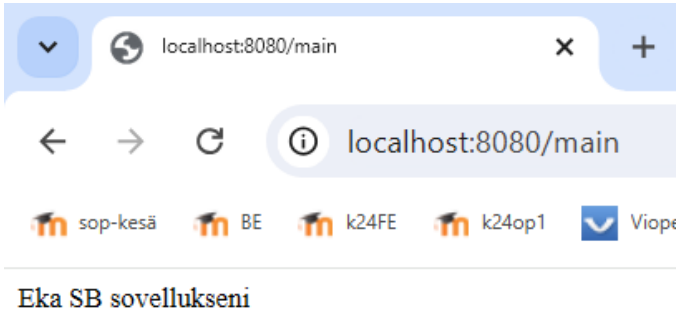


Sovelluksen ajaminen

Terminal näkymästä näet, milloin Tomcat ja sovelluksesi ovat käynnistyneet.



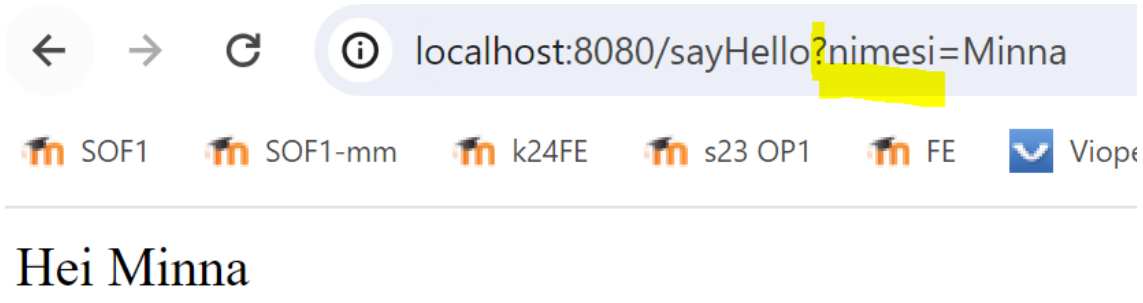
Kirjoittamalla selaimen osoitekenttään (url:iin) http://localhost:8080/main lähetät requestin/kutsun sovelluksellesi ja näet responsen ruudulla



Lisää toiminnallisuutta

Lisätään **`MyController`** luokkaan toiminnallisuus, jonka tuloksena nettisivulle tulostuu ”Hei <nimi>!”

<nimi> paikalle tulee nimi, jonka annat url:ssa parametrinä

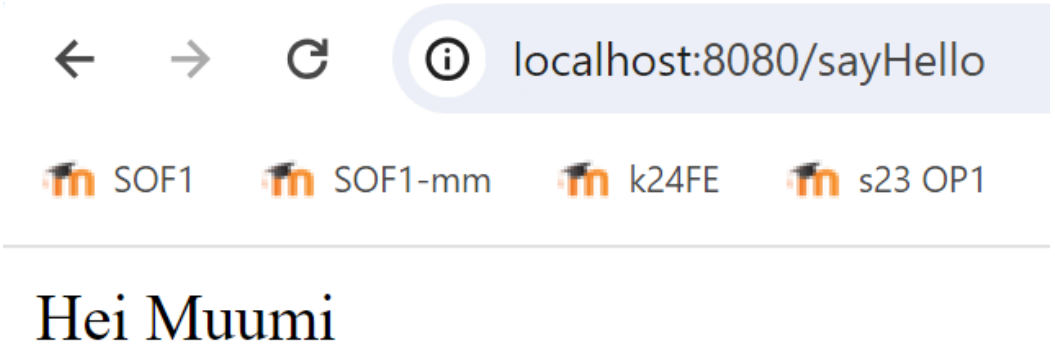



```
@RequestMapping("sayHello")
@ResponseBody
public String returnGreeting(@RequestParam (name="nimesi") String etunimi) {
    return "Hei " + etunimi;
}
```

`@RequestParam` annotaation avulla sidomme urlin mukana tulevan parametrimuutujan **`nimesi`** arvon (esimerkissä Minna) Java puolella muuttujaan **`etunimi`**

Lisää toiminnallisuutta

Muokataan toiminnallisuus siten, että jos nimeä ei anneta, niin tulostetaan jokin oletusarvo.



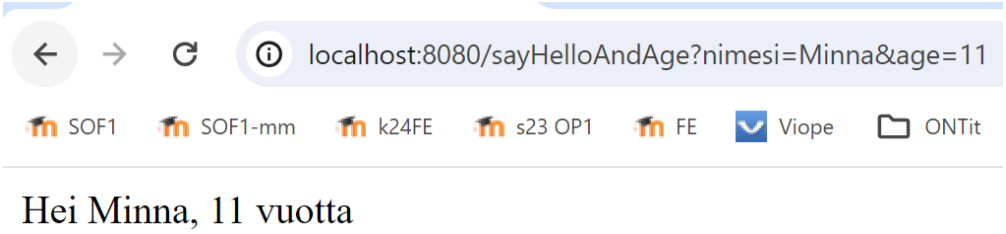
`MyControllerissa` on tätä vastaava koodi

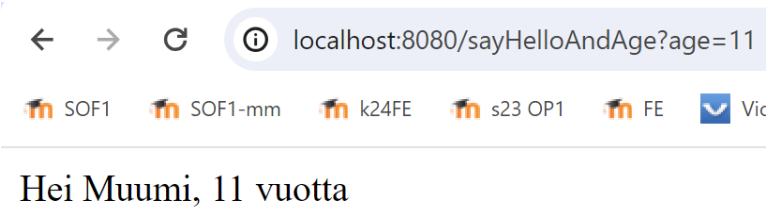
```
@RequestMapping("sayHello")
@ResponseBody
public String returnGreeting(@RequestParam (name="nimesi", required=false, defaultValue="Muumi") String etunimi) {
    return "Hei " + etunimi;
}
```

`@RequestParam`:lle voidaan antaa attribuutteja. Esim. **`required`**-attribuutilla kerrotaan, ettei **`nimi`**-parametri ei ole pakollinen, **`defaultValue`** kertoo oletusarvon.

Lisää toiminnallisuutta

Muokataan toiminnallisuutta siten, että nimen lisäksi annetaan myös ikä





`MyControllerissa` on tätä vastaava koodi

```
@RequestMapping("sayHelloAndAge")
@ResponseBody
public String returnGreeting(@RequestParam (name="nimesi", required=false, defaultValue="Muumi") String etunimi,
        @RequestParam int age) {
    return "Hei " + etunimi + ", " + age + " vuotta";
}
```