



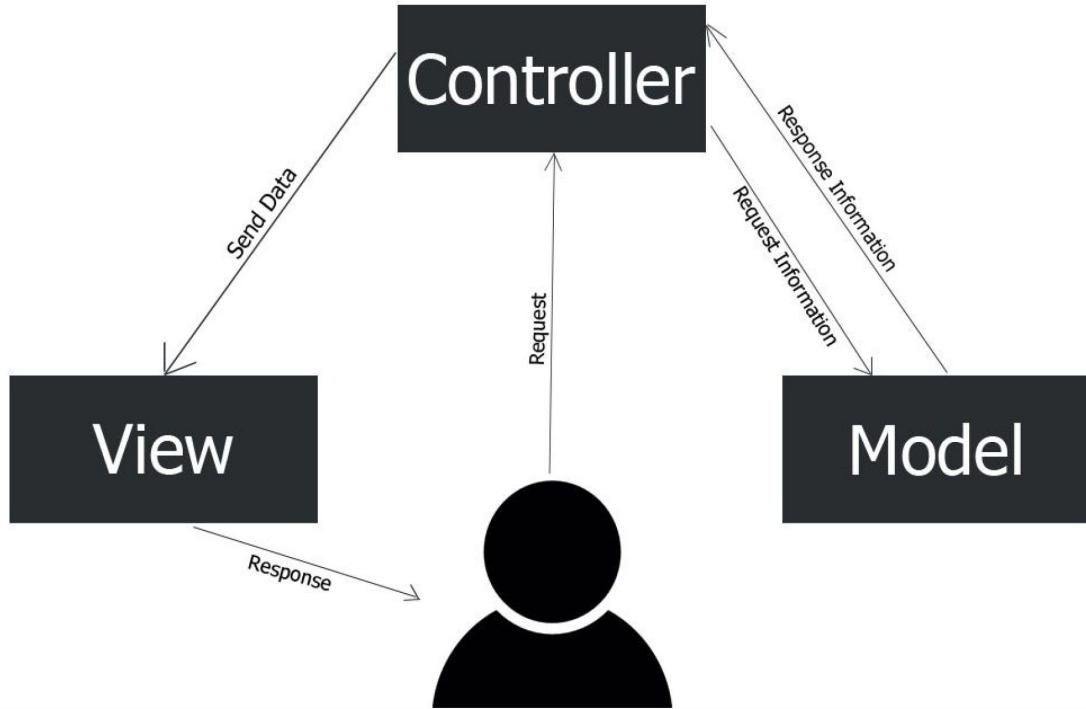
DEV4

Spring boot

MVC



Model - View - Controller



Controller

- A control center of your application
- Receives a request, processes it and sends a response back
- Example flow
 - User visits /users
 - This triggers the get() function in the UserController
 - That functions gets all requested users and sends the data back



Model



- A class to contain a single entity, like a user
- Bundles all the data → Often matches the database structure for that entity
- Easier to work with compared to an array with data
- Can also have it's own functions
- Example usage
 - All users are requested
 - They are fetched from the database and the data is stored in model class User
 - Each instance of that User class is a single, separate user

Model: DTO

- Data Transfer Object
- Extra class, related to a model
- Only contains fields for the user to view / change (depends on the context)



Model: DTO example

- A user can change their email and password, but not their id
- The data from the form is saved in the DTO
- The DTO will have a email and password field, but no id field
- This prevents the user from changing the id (even when extra forms fields are added maliciously)



Model: DTO

```
// Location: dto > CreateEmployeeRequest.kt  
data class CreateEmployeeRequest(  
    // No ID (user should not pick the id)  
    var fullName: String, var email: String,  
    var position: Position  
){}  

```



Model: DTO usage in controller

```
// Location: controllers/EmployeeController.kt
fun hireEmployee(@RequestBody employeeRequest:
CreateEmployeeRequest): {
    // Convert DTO to actual model
    // Can happen where you want: controller, service, ...
    val employee = Employee(fullName = employeeRequest.fullName)
}
```



View

- The visual representation
- We'll focus on API's, so no need for the view just yet



Spring Boot



Spring Boot

- MVC based framework
- Originally for Java, also available for Kotlin
- Create your project via IntelliJ → **Ultimate edition** needed



New Project

Search

New Project


Empty Project

Generators

- ✓ Maven Archetype
- 🔥 Jakarta EE
- 🌱 **Spring Initializr**
- 📁 JavaFX
- 📦 Quarkus
- ⚡ Micronaut
- ⚡ Ktor
- 📦 Kotlin Multiplatform
- 📦 Compose for Desktop
- 📄 HTML
- ⚙️ React
- ex Express
- 📄 Angular CLI
- 📦 IDE Plugin
- 🤖 Android
- 🌐 Vue.js

Server URL: start.spring.io ⋮

Name:


Location: 

Project will be created in: ~/Projects/_ehb/dev4/name

☐ Create Git repository


Language:

Type:

Group: 


Artifact:

Package name:

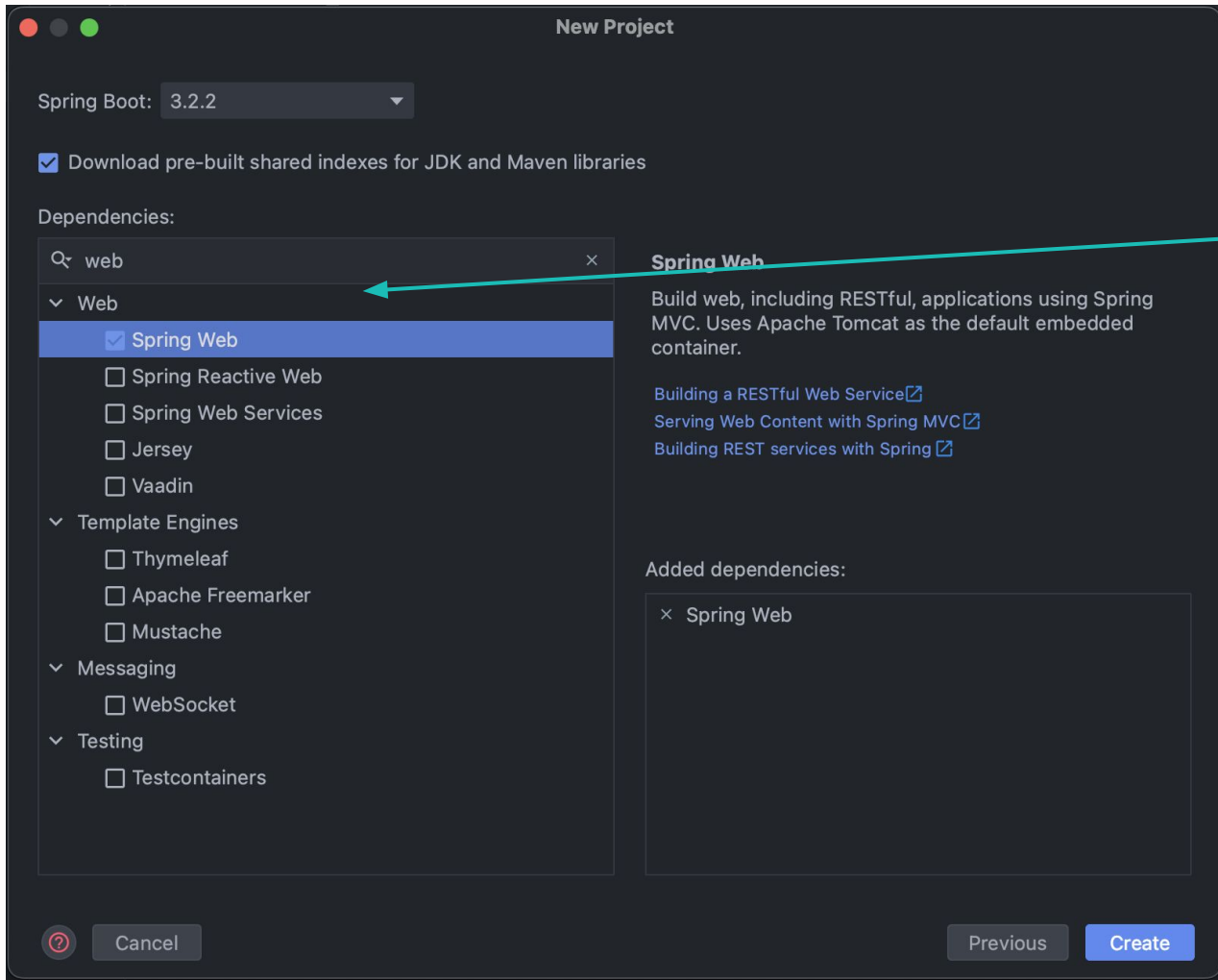
JDK:  ▼

Java: ▼

Packaging:



Your organisation URL in reverse = unique identifier for the owner of the project



Pick any libraries
you need

Annotations

- Way of using built-in behaviour from the framework
- Start with an @
- Lots of different annotations are available



Annotations

```
@RestController
@RequestMapping("numbers")
class NumbersController {
    @GetMapping("pi")
    fun pi(): Double {
        return Math.PI
    }
}
```

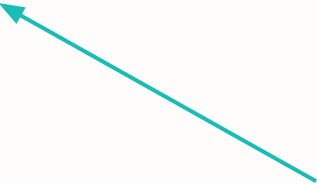
Mark class as controller



Prefix urls for this controller



/numbers/pi will be handled by that function



Maven



Maven

- The dependency manager we will use
- Similar to npm or composer
- Pom.xml for config (similar to package.json)
- Most dependencies are added during IntelliJ project setup
- Open source
 - Maintained by the community
 - Owned by the Apache software foundation



