ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

ΚΕΝΤΡΟ ΕΠΙΜΟΡΦΩΣΗΣ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗΣ

Spring
Spring Boot

Αθ. Ανδρούτσος

Spring Framework

- Το Spring (https://spring.io/projects/spring-framework) είναι ένα open-source framework, που δημιουργήθηκε αρχικά από τον Rod Johnson, το 2002 και περιγράφεται στο βιβλίο του "*Expert One-on-One J2EE Design and Development*", *Wrox, 2002.*
- Το Spring Framework δημιουργήθηκε για να αντιμετωπίσει την πολυπλοκότητα στην ανάπτυξη επιχειρησιακών εφαρμογών (enterprise applications)
- Είναι εναλλακτικό framework του Java EE / Jakarta EE

Spring Versions

Release	Released	OSS support	Commercial Support	Latest
6.1	10 months ago (16 Nov 2023)	Ends in 10 months (31 Aug 2025)	Ends in 2 years and 2 months (31 Dec 2026)	6.1.13 (12 Sep 2024)
~~6.0~~	1 year and 10 months ago (16 Nov 2022)	Ended 1 month and 1 week ago (31 Aug 2024)	Ends in 1 year and 2 months (31 Dec 2025)	~~6.0.23 (14 Aug 2024)~~
~~5.3 (LTS)~~	3 years and 11 months ago (27 Oct 2020)	Ended 1 month and 1 week ago (31 Aug 2024)	Ends in 2 years and 2 months (31 Dec 2026)	~~5.3.39 (14 Aug 2024)~~
~~5.2~~	5 years ago (30 Sep 2019)	Ended 2 years and 9 months ago (31 Dec 2021)	Ended 9 months ago (31 Dec 2023)	~~5.2.25 (13 Jul 2023)~~

Show more unmaintained releases

- https://endoflife.date/spring-framework

Spring / JDK Compatibility

JDK/Jakarta EE Compatibility

Release	JDK	Jakarta EE
6.1	17 - 23	9 - 10
6.0	17 - 21	9 - 10
5.3	8 - 21	7 - 8

- Το Spring καθώς εξελίσσεται χρησιμοποιεί νέα στοιχεία της Java και η LTS έκδοση είναι συμβατή με JDK 8 – JDK 21 και Jakarta EE 7-8 (javax namespace), ενώ η έκδοση 6.1 είναι συμβατή με Java 17-23 και Jakarta 9-10 (Jakarta namespace)

Software Development

- Ο τρόπος που αναπτύσσουμε εφαρμογές σήμερα έχει αλλάξει
- Ενώ παλαιότερα στα αρχικά στάδια, ο πιο κοινός τύπος εφαρμογών ήταν browser-based εφαρμογές με relational databases, σήμερα αναπτύσσουμε επίσης (και κυρίως) Single Page Applications (SPA) με REST microservices που γίνονται deploy στο cloud και κάνουν persist τα data σε διάφορους τύπους Βάσεων Δεδομένων (SQL Databases, NoSQL Databases, Blockchain, κ.λπ.)

Spring Modules (1)

- Το Spring ακολουθώντας τις τάσεις στην τεχνολογία λογισμικού δεν είναι πια ένα απλό framework αλλά **ένα σύνολο από modules** που παρέχουν λειτουργικότητα out-of-the-box για κάθε concern (Spring Core, , Spring Data, Spring MVC, Spring Security, κ.λπ.)
- Επίσης, το Spring έχει απλοποιήσει το config που χρειάζεται για να γίνουν integrate τα διάφορα modules, εισάγοντας το Spring Boot

Spring Modules (2)

- Το *Spring* περιλαμβάνει διάφορα modules, όπως:
  - ***Spring Core,***
  - ***Spring Data,***
  - ***Spring Boot,***
  - ***Spring MVC,***
  - ***Spring Security***, κλπ.

Χαρακτηριστικά Spring (1)

- Η βασική φιλοσοφία του Spring είναι το ***non-invasive programming***, δηλαδή η χρησιμοποίηση απλών **POJO classes** (Spring Beans η απλά **Beans**) και η χρήση **annotations** για καθετί που θέλουμε να δηλώσουμε και όχι η χρήση *implements* ή *extends* ειδικών interfaces ή κλάσεων γεγονός που θα καθιστούσε το framework βαρύ και non-portable

Χαρακτηριστικά Spring (2)

- Τα βασικά πλεονεκτήματα και χαρακτηριστικά του Spring σήμερα είναι:
  - Lightweight μοντέλο ανάπτυξης με **POJOs**
  - Loosely coupling με **Dependency Injection** και χρήση interfaces
  - Declarative programming με χρήση **annotations** και Aspect-Oriented Programming (AOP)
  - Auto-configuration με **Spring Boot**

Dependency Injection / Beans

- Κάθε εφαρμογή συντίθεται από διάφορα συνεργαζόμενα components ή αλλιώς **Beans**
- Όταν η εφαρμογή ξεκινάει να τρέχει θα πρέπει τα components να δημιουργηθούν και να συνεργαστούν (**wiring**)
- Το Spring παρέχει ένα *container*, που ονομάζεται **Spring Application context (AppicationContext)** και υλοποιεί τον IoC Container, που διαχειρίζεται αυτά τα components και τα κάνει inject όπου έχει οριστεί
- Αυτός ο μηχανισμός συνεργασίας των beans (wiring) βασίζεται σε ένα design pattern που έχουμε δει και ονομάζεται **Dependency Injection (DI)**

Wiring beans & Auto-wiring

- Παραδοσιακά υπήρχαν δύο τρόποι για να δηλώσουμε το DI, ποια beans δηλαδή πρέπει να συνεργαστούν, *XML* και *Java*
- Αν για παράδειγμα έχουμε δύο beans το *TeacherServiceimpl* και το *CourseServiceImpl*, όπου το *CourseServiceImpl* χρειάζεται ως dependency το *TeacherServiceImpl* θα μπορούσε τα xml και Java-based configuration files να είναι το παρακάτω:

```
<bean>
   <bean id="TeacherServiceImpl"
    class="gr.aueb.thanos.TeacherSeviceImpl"
     <contructor-arg ref="TeacherServiceImpl"
</bean>
<bean>
   <bean id="CourseServiceImpl"
   class="gr.aueb.thanos.CourseSeviceImpl">
   <contructor-arg ref="CourseServiceImpl"
</bean>
```

```
@Configuration
public class ServiceConfiguration {

   @Bean
   public TeacherServiceImpl teacherServiceImpl() {
       return new TeacherServiceImpl();
   }

   @Bean
   public CourseServiceImpl courseServiceImpl(TeacherServiceImpl
                                                teacherServiceImpl) {
       return new CourseServiceImpl(teacherServiceImpl);
   }
}
```

Spring Boot (1)

- Ωστόσο με τον ερχομό του **Spring Boot** το config γίνεται αυτόματα
- Δηλαδή γίνεται αυτόματα *component scanning* και auto-configuration
- Στη συνέχεια μπορούμε να κάνουμε wiring τα συνεργαζόμενα components με το annotation ***@Autowired*** και να γίνεται bean injection (Αντίστοιχο του @Inject , @Ejb της Java EE)

Spring Boot (2)

- Το *Spring Boot* επιταχύνει την ανάπτυξη του κώδικα και τελικά το μόνο που χρειάζεται να γράψουμε εμείς για το configuration είναι μία γραμμή κώδικα!
- Το *Spring Boot* επίσης παρέχει μηχανισμούς dependency management μέσω του Maven ή του Gradle.
- Δεν μπορούμε λοιπόν να φανταστούμε την ανάπτυξη Spring εφαρμογών χωρίς το Spring Boot.
- **Για αυτό θεωρούμε πως το Spring είναι άρρηκτα συνδεδεμένο με το Spring Boot**

Spring Boot versions

Release	Released	OSS support	Commercial Support	Latest
3.3	4 months and 2 weeks ago (23 May 2024)	Ends in 7 months (23 May 2025)	Ends in 1 year and 10 months (23 Aug 2026)	3.3.4 (19 Sep 2024)
3.2	10 months ago (23 Nov 2023)	Ends in 1 month and 2 weeks (23 Nov 2024)	Ends in 1 year and 4 months (23 Feb 2026)	3.2.10 (19 Sep 2024)
~~3.1~~	1 year and 4 months ago (18 May 2023)	Ended 4 months and 3 weeks ago (18 May 2024)	Ends in 10 months (18 Aug 2025)	~~3.1.12 (23 May 2024)~~
~~3.0~~	1 year and 10 months ago (24 Nov 2022)	Ended 10 months ago (24 Nov 2023)	Ends in 4 months and 2 weeks (24 Feb 2025)	~~3.0.13 (23 Nov 2023)~~
~~2.7~~	2 years and 4 months ago (19 May 2022)	Ended 10 months ago (24 Nov 2023)	Ends in 2 years and 2 months (31 Dec 2026)	~~2.7.18 (23 Nov 2023)~~
~~2.6~~	2 years and 10 months ago (17 Nov 2021)	Ended 1 year and 10 months ago (24 Nov 2022)	Ended 7 months ago (24 Feb 2024)	~~2.6.15 (18 May 2023)~~

- Η πιο πρόσφατη έκδοση είναι η έκδοση 3.3.4 που είναι συμβατή με JDK 17-21

@Autowired (1) - Constructors

- Με το **@Autowired** annotation επιτρέπουμε στο Spring να ικανοποιήσει αυτόματα τα dependencies με DI. Και παρότι μπορούμε να χρησιμοποιήσουμε κλάσεις (implementations) ως πεδία μιας άλλης κλάσης, **ως καλή πρακτική χρησιμοποιούμε interfaces ως ιδιότητες κλάσεων ώστε, όπως έχουμε δει, να έχουμε loosely coupling** μεταξύ των beans. Ας υποθέσουμε την κλάση **MyPc**, όπου το πεδίο IUsb είναι interface και θεωρούμε ένα μόνο implementation (κλάση) του UsbStick

```
1  @Component
2  public class MyPc implements Ipc {
3
4      private IUsb      usbStick;
5
6      @Autowired
7      public MyPc( IUsb     usbStick) {
8          this.usbStick = usbStick;
9      }
10 }
```

Αν δεν υπάρχουν matching beans θα δημιουργηθεί Exception καθώς θα δημιουργείται το ApplicationContext

- Στη γραμμή 6, ο constructor έχει γίνει annotate με **@Autowired**, προστάζοντας το Spring να αρχικοποιήσει το usbStick με ένα *instance* μίας **μοναδικής** κλάσης που κάπου έχουμε ορίσει και υλοποιεί το interface
- Αυτό θα γίνει όταν το Spring θα φορτώσει την κλάση MyPc καλώντας τον constructor της MyPc. Τότε θα ψάξει να βρει στο ApplicationContext μία κλάση που να είναι assignable (θα υλοποιεί δηλαδή) στο IUsb και θα την κάνει inject

@Autowired (2) - Setters

- Το **@Autowired** δεν περιορίζεται στους constructors. Μπορεί να χρησιμοποιηθεί και σε *setters*. Στο παρακάτω παράδειγμα, αφού το bean MyPc έχει γίνει instantiate από το Spring με τη χρήση του default constructor (δεν έχουμε ορίσει κάτι άλλο), στη συνέχεια το Spring θα προσπαθήσει να ικανοποιήσει τα dependencies που έχουμε ορίσει με το @Autowired

```
1  @Component
2  public class MyPc implements Ipc {
3
4      private  IUsb     usbStick;
5
6      @Autowired
7      public setUsbStick( IUsb     usbStick) {
8          this.usbStick = usbStick;
9      }
10 }
```

@Autowired (3) - Properties

- Σε παλαιότερες εκδόσεις του Spring μπορούσαμε να **χαρακτηρίσουμε @Autowired κατευθείαν τα properties** και τότε θα αρχικοποιηθούν -όταν φορτωθεί η κλάση- με τον default constructor

```
1  @Component
2  public class MyPc implements Ipc {
3
4      @Autowired
5      private  IUsb      usbStick;
6
7  }
```

@Autowired (4)

- Δεν υπάρχει ωστόσο κάτι ιδιαίτερο στους Constructors, setters, και properties. Το @Autowired μπορεί να χρησιμοποιηθεί σε οποιαδήποτε μέθοδο, όπως η play()

```
1  @Component
2  public class MyPc implements Ipc {
3
4      @Autowired
5      private  IUsb    usbStick;
6
7      @Autowired
8      public void play( IUsb    usbStick) {
9          usbStick.play();
10     }
11 }
```

- Σε κάθε περίπτωση το Spring θα προσπαθήσει να ικανοποιήσει τα dependencies
- Αν δεν υπάρχει το matching bean θα δημιουργηθεί exception
- Αν υπάρχουν πολλαπλά implementations τότε μπορούμε να χρησιμοποιήσουμε Qualifiers (δείτε επόμενη διαφάνεια)

@Autowired (5)


```
1  @Component
2  @Qualifier("usb")
3  public class UsbImpl implements  IUsb     {
4      ...
5  }
6
7  @Component
8  @Qualifier("ssd")
9  public class SsdImpl implements  IUsb     {
10     ...
11 }
12
13 @Component
14 public class MyPc implements Ipc {
15
16     @Autowired
17     private  IUsb    usb;
18
19     @Autowired
20     private  IUsb    ssd;
21     }
22 }
```

- Αφού ορίσουμε τις υλοποιήσεις με @Qualifier μπορούμε να κάνουμε στη συνέχεια inject με το όνομα του Qualifier

Stereotype annotations (1)

- Το Spring κάνει **auto scan κλάσεις που βρίσκονται στο ίδιο package που βρίσκεται η main ή σε subpackages**
- Κάνει scan κλάσεις που έχουν γίνει mark με κάποια stereotype annotations. Το Spring χρησιμοποιεί stereotype annotations για να προσδιορίσει το ρόλο κάθε κλάσης σε μία εφαρμογή
- Τα stereotypes annotations είναι **@Component @Controller @RestController, @Repositiry, @Service, @Configuration** και ο ρόλος τους είναι να προσδιορίσουν τον σκοπό της κλάσης
- Μέσα σε κλάση που έχει χαρακτηριστεί ως **@Configuration** μπορούμε να χαρακτηρίσουμε μεθόδους που επιστρέφουν beans ως **@Bean**. Η @Configuration κλάση αρχικοποιεί τα beans που επιστρέφουν οι @Bean μέθοδοι στην αρχικοποίηση του IoC κατά την έναρξη του προγράμματος

@SpringBootApplication

- Μία Spring Boot εφαρμογή τυπικά ξεκινάει με μία main σε μία κλάση που έχει σημανθεί ως ***@SpringBootApplication***
- Στην πραγματικότητα το annotation **@SpringBootApplication** περιλαμβάνει τρία annotations: @Configuration, @ComponentScan και @EnableAutoConfiguration
- Με αυτό το annotation το SpringBoot κάνει scan για Components στο current package που περιέχει τη main καθώς και στα subpackages

Spring MVC

Spring on the Web

- Παρότι με το Spring μπορούμε να αναπτύξουμε διάφορες εφαρμογές, πιο συχνά το Spring χρησιμοποιείται για την ανάπτυξη enterprise full-stack web εφαρμογών
- Το βασικό Spring module που μας δίνει τη δυνατότητα ανάπτυξης web εφαρμογών είναι το **Spring MVC**

Spring MVC (1)

- Το **Spring MVC** δεν είναι ένα μόνο dependency αλλά ένα σύνολο από jars και είναι διαθέσιμο μέσω του Spring Boot αν εισάγουμε στο Gradle ή στο Maven το **spring-boot-starter-web** dependency
- Στο αρχείο **build.gradle του Gradle** (αντίστοιχο του *pom.xml*) μπορούμε να ορίσουμε το παρακάτω:

```
implementation 'org.springframework.boot:spring-boot-starter-web'
```

Spring MVC (2)

- Το **spring-boot-starter-web** περιέχει επίσης transitive dependencies για
  - **Servlet API**
  - **Embedded Apache Tomcat**
  - **Logging με Logback και SLF4j**
  - **Jackson για JSON serialization / deserialization**

Spring Boot dependencies

- Όλα τα dependencies του Spring Boot έχουν το **spring-boot-starter** στο artifact id. Τα **starter** dependencies δεν έχουν library code από μόνα τους, αλλά μεταβατικά φέρνουν άλλα libraries
- Έτσι **σκεφτόμαστε τα dependencies σε όρους της λειτουργικότητας** που προσφέρουν (π.χ. spring-boot-starter-validation) και όχι σε όρους τεχνικής ονομασίας του κάθε library (π.χ. hibernate-validation)
- Επίσης, δεν χρειάζεται να ανησυχούμε για το dependency version compatibility, παρά μόνο για το Spring Boot version, το οποίο οργανώνει τα versions των dependencies (βλ. παρακάτω)

Gradle (1)

- Θα πρέπει επίσης να εισάγουμε στο Gradle και το **plugin του Spring Boot** το οποίο επιτρέπει την εκτέλεση των εφαρμογών μας καθώς και τη διαχείριση των spring-boot dependencies
- Θα χρησιμοποιήσουμε την έκδοση 3.*
- Η έκδοση 3.* απαιτεί Java 17 – 21
- Στο build.gradle εισάγουμε το παρακάτω

```
id 'org.springframework.boot' version '3.3.4'
```

Gradle (2)

- Επίσης χρειαζόμαστε support για **Java** compilation, εισάγοντας το **java plugin**

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.3.4'
```

- Με id **'war'** plugin μπορούμε να επεκτείνουμε το Java plugin και υποστηρίζεται η δημιουργία WAR files για να τα χρησιμοποιήσουμε σε ένα Apache Tomcat για παράδειγμα σε άλλο installation. Κάνει disable το default jar (σε αυτή την περίπτωση πρέπει να γίνει exclude ο default Apache Tomcat)

```
implementation 'org.springframework.boot:spring-boot-starter-web' {
    exclude module: 'spring-boot-starter-tomcat'
}
```

Gradle (3)

- Για να μπορεί το Gradle να διαχειρίζεται αυτόματα τα versions των dependencies του Spring μπορούμε να εισάγουμε και το ***io.spring.dependency-management*** plugin
- Έτσι δεν θα χρειάζεται να δίνουμε versions στα dependencies μιας και θα τα διαχειρίζεται αυτόματα το Gradle μέσω του *io.spring.dependency-management*

```
1    plugins {
2        id 'java'
3        id 'org.springframework.boot' version '3.3.4'
4        id 'io.spring.dependency-management' version '1.1.6'
5    }
```

Maven Central


```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.2.1'
    id 'io.spring.dependency-management' version '1.1.4'
}

group = 'gr.aueb.cf'
version = '0.0.1-SNAPSHOT'

java {
    sourceCompatibility = '17'
}

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
    mavenCentral()
}
```

- Το **sourceCompatibility** στο Gradle λειτουργεί όπως το maven.compiler.source στο Maven. Αν το targetCompatibilty δεν ορίζεται, αντιστοιχεί στο sourceCompatibility
- Για να ορίσουμε στο Gradle το Maven Central repository για να κατεβάζουμε dependencies θα πρέπει να ορίσουμε το mavenCentral() στα repositories

Java toolchain


```
1   plugins {
2       id 'java'
3       id 'org.springframework.boot' version '3.3.4'
4       id 'io.spring.dependency-management' version '1.1.6'
5   }
6
7   group = 'gr.aueb.cf'
8   version = '0.0.1-SNAPSHOT'
9
10  java {
11      toolchain { JavaToolchainSpec it ->
12          languageVersion = JavaLanguageVersion.of(17)
13          vendor = JvmVendorSpec.AMAZON
14      }
15  }
```

- Το toolchain υπάρχει στο Gradle από το version 6.7 αλλά μόλις τελευταία χρησιμοποιείται παραγωγικά. Επιτρέπει σε ομάδες να χρησιμοποιούν την ίδια version του JDK ανεξάρτητα από το αν την έχουν εγκαταστήσει στο σύστημά τους ή όχι. Αν δεν είναι εγκατεστημένη τοπικά, τότε το Gradle κατεβάζει το JDK και το εγκαθιστά

Maven


```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-toolchains-plugin</artifactId>
            <version>3.0.0</version>
            <executions>
                <execution>
                    <goals>
                        <goal>toolchain</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

- Στο Maven μπορούμε να ορίσουμε το Maven toolchain plugin ενώ χρειάζεται να ορίσουμε και ένα αρχείο toolchains.xml στο ${user.home}/.m2 directory (βλ. επόμενη διαφάνεια)

toolchains.xml


```
<toolchains>
  <toolchain>
    <type>jdk</type>
    <provides>
      <version>17</version>
      <vendor>Amazon Corretto</vendor>
    </provides>
    <configuration>
      <jdkHome>/path/to/corretto-17</jdkHome>
    </configuration>
  </toolchain>
</toolchains>
```

- Το Maven δεν κατεβάζει (downloads) όπως το Gradle το JDK που ορίζεται, αν δεν υπάρχει. Θα πρέπει το JDK να είναι εγκατεστημένο και να οριστεί το path του στο toolchains.xml
- Αυτό βοηθάει στο να μην χρειάζεται να αλλάζουμε το JAVA_HOME κάθε φορά που χρειαζόμαστε διαφορετική JDK version

build.gradle


```
17  repositories {
18      mavenCentral()
19  }
20
21  dependencies {  Edit Starters...
22      implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'
23      implementation 'org.springframework.boot:spring-boot-starter-web'
24      testImplementation 'org.springframework.boot:spring-boot-starter-test'
25      testRuntimeOnly 'org.junit.platform:junit-platform-launcher'
26      compileOnly 'org.projectlombok:lombok'
27      annotationProcessor 'org.projectlombok:lombok'
28  }
29
30  tasks.named('test', Test) {  Test it ->
31      useJUnitPlatform()
32  }
```

- Περαιτέρω στο build.gradle ορίζουμε το mavenCentral ως το repository που θα χρησιμοποιείται για τα dependencies.

Repositories


```
repositories {
    mavenCentral() // Use Maven Central repository
    maven {
        url "https://your-custom-repo.com/repository/maven-public/"
    }
}
```

- Εκτός από το maven central μπορούμε να ορίσουμε και άλλα repositories όπως παραπάνω αν χρειαζόμαστε dependencies που δεν υπάρχουν στο maven central

Maven Repositories


```
<project>
  <!-- Other configuration goes here -->

  <repositories>
    <!-- Maven Central (Maven uses this by default, but you can explicitly declare it) -->
    <repository>
      <id>central</id>
      <url>https://repo.maven.apache.org/maven2</url>
      <releases>
        <enabled>true</enabled>
      </releases>
      <snapshots>
        <enabled>false</enabled> <!-- Maven Central typically doesn't host snapshots -->
      </snapshots>
    </repository>

    <repository>
      <id>custom-snapshots</id>
      <url>https://your-custom-repo.com/repository/maven-snapshots/</url>
      <snapshots>
        <enabled>true</enabled>
      </snapshots>
    </repository>
  </repositories>

</project>
```

- Τα Snapshots είναι dev versions των dependencies

Gradle

- Μπορούμε να κάνουμε install το Gradle από το https://gradle.org/install/
- Δεν είναι απαραίτητο για να τρέξουμε τα project μας από IntelliJ Ultimate / Eclipse μιας και τα δύο έχουν Gradle plugin και project runner
- Αλλά είναι απαραίτητο για να τρέχουμε τα apps με Gradle χωρίς να χρειάζεται να έχουμε IDE με **./gradlew bootRun**
- Ή **./gradlew build** και μετά **java –jar name.jar** στον φάκελο /build/libs (με java –jar name.jar & τρέχει σαν process σε UNIX-like συστήματα)

Gradle Installation

- Τα βήματα του installation όπως αναφέρονται στη σελίδα του Gradle https://gradle.org/install/

1. Πάμε στα Releases https://gradle.org/releases/ (η τρέχουσα έκδοση είναι η 8.10.2 – SEP 2024) και κατεβάζουμε το **binary-only**
2. **Extract** σε ένα φάκελό μας (π.χ. **C:\Gradle**) το φάκελο που βρίσκεται μέσα στο .zip (**gradle-8.10.2**)
3. Ρυθμίζουμε το Path environmental variable (**Windows + R** και μετά **Για προχωρημένους / Μεταβλητές περιβάλλοντος** στα Windows) ώστε να περιέχει το bin του Gradle (**C:\gradle-8.10.2\bin**)

Επιβεβαίωση Εγκατάστασης

- **gradle -v** (στο παρόν είναι εγκατεστημένη η έκδοση 8.5)

```
C:\Users\a8ana>gradle -v

------------------------------------------------------------
Gradle 8.5
------------------------------------------------------------

Build time:   2023-11-29 14:08:57 UTC
Revision:     28aca86a7180baa17117e0e5ba01d8ea9feca598

Kotlin:       1.9.20
Groovy:       3.0.17
Ant:          Apache Ant(TM) version 1.10.13 compiled on January 4 2023
JVM:          17.0.6 (Amazon.com Inc. 17.0.6+10-LTS)
OS:           Windows 10 10.0 amd64
```

Spring-boot-starter-web


```
26        <dependency>
27            <groupId>org.springframework.boot</groupId>
28            <artifactId>spring-boot-starter-web</artifactId>
29        </dependency>
```

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web'
```

- Από το mvn repository εισάγουμε το starter-web στα dependencies του build.gradle. Δεν χρειάζεται η έκδοση μιας και το plugin spring.dependency-management φέρνει την τελευταία έκδοση
- Το implementation χαρακτηρίζει τα dependencies που χρειάζονται και @CompileTime και @Runtime

Maven - Spring Boot Starter Parent


```
5     <parent>
6         <groupId>org.springframework.boot</groupId>
7         <artifactId>spring-boot-starter-parent</artifactId>
8         <version>3.3.4</version>
9         <relativePath/> <!-- lookup parent from repository -->
10    </parent>
```

- Αν στο Maven εισάγουμε το **spring-boot-starter-parent** δεν χρειάζεται να εισάγουμε versions στα dependencies, όπως το *spring-boot-starter-web*

Pom.xml (1)


```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="h
3           xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
4      <modelVersion>4.0.0</modelVersion>
5      <parent>
6          <groupId>org.springframework.boot</groupId>
7          <artifactId>spring-boot-starter-parent</artifactId>
8          <version>3.3.4</version>
9          <relativePath/> <!-- lookup parent from repository -->
10     </parent>
11     <groupId>gr.aueb.cf</groupId>
12     <artifactId>spring-starter6-test-mvn</artifactId>
13     <version>0.0.1-SNAPSHOT</version>
14     <name>spring-starter6-test-mvn</name>
15     <description>spring-starter6-test-mvn</description>
16     <url/>
17
18     <properties>
19         <java.version>17</java.version>
20     </properties>
```

- Το **parent POM του spring-boot** παρέχει αυτόματα versions στα dependencies (by default επίσης θέτει το version της Java στο 1.8)
- Μέσα στα properties όπως έχουμε δει αν θέλουμε να κάνουμε override το Java version μπορούμε ρητά, όπως στις γραμμές 18-20 να θέσουμε java.version 17 (που περιλαμβάνει source και target)

Pom.xml (2)


```
21      <dependencies> Edit Starters...
22          <dependency>
23              <groupId>org.springframework.boot</groupId>
24              <artifactId>spring-boot-starter-thymeleaf</artifactId>
25          </dependency>
26          <dependency>
27              <groupId>org.springframework.boot</groupId>
28              <artifactId>spring-boot-starter-web</artifactId>
29          </dependency>
30
31          <dependency>
32              <groupId>org.projectlombok</groupId>
33              <artifactId>lombok</artifactId>
34              <optional>true</optional>
35          </dependency>
36          <dependency>
37              <groupId>org.springframework.boot</groupId>
38              <artifactId>spring-boot-starter-test</artifactId>
39              <scope>test</scope>
40          </dependency>
41      </dependencies>
```

- Εισάγουμε dependencies

Pom.xml (3)

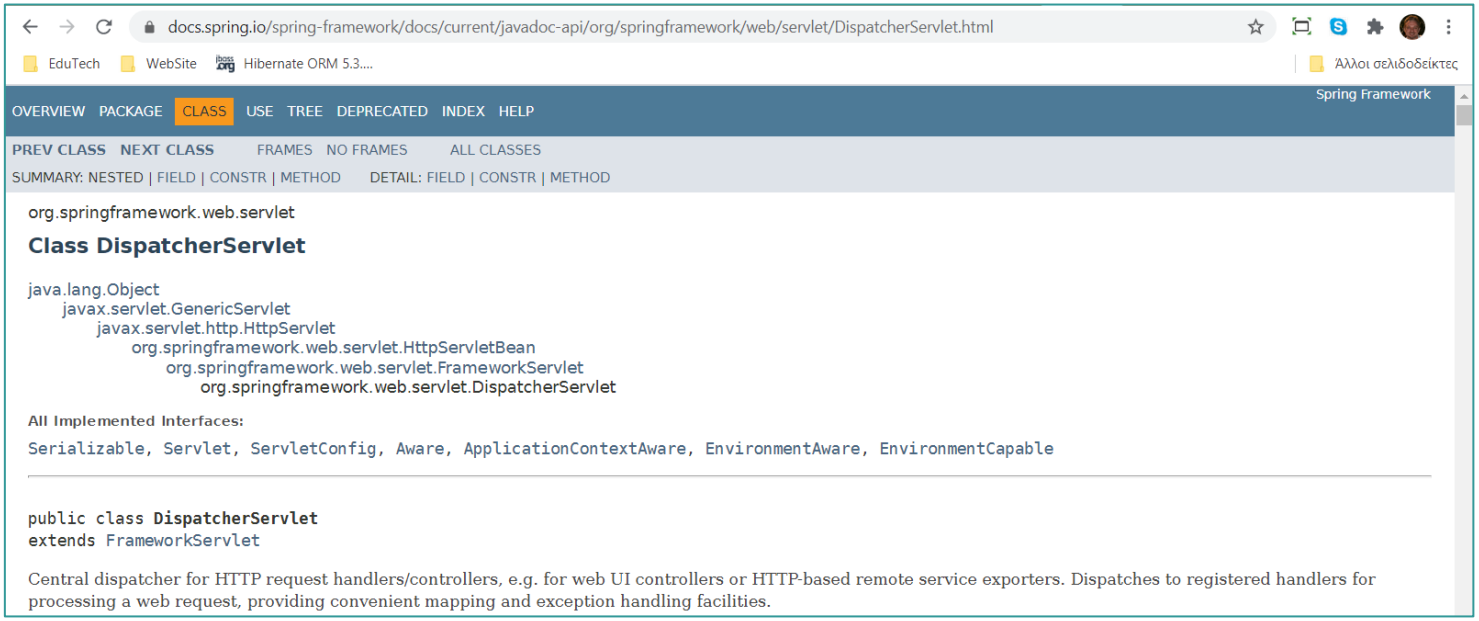

```
43      <build>
44          <plugins>
45              <plugin>
46                  <groupId>org.springframework.boot</groupId>
47                  <artifactId>spring-boot-maven-plugin</artifactId>
48                  <configuration>
49                      <excludes>
50                          <exclude>
51                              <groupId>org.projectlombok</groupId>
52                              <artifactId>lombok</artifactId>
53                          </exclude>
54                      </excludes>
55                  </configuration>
56              </plugin>
57          </plugins>
58      </build>
59
60  </project>
```

- Το **spring-boot maven-plugin** εισάγει το spring-boot στο maven και δίνει τη δυνατότητα για δημιουργία **executable jar**

Dispatcher Servlet (1)

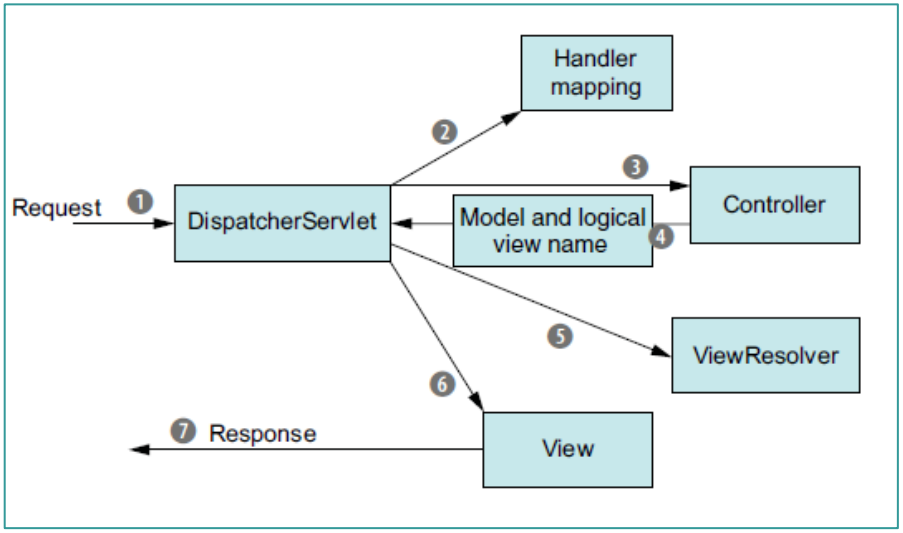
- Το Spring MVC έχει φτιαχτεί on top of Servlet technology
- Βασικό αρχιτεκτονικό χαρακτηριστικό είναι η ύπαρξη ενός **κεντρικού Dispatcher Servlet, δηλαδή ενός μοναδικού Front Controller** που δρομολογεί όλη την κίνηση (requests) προς τους επιμέρους Controllers
- Η κλάση ***DispatcherServlet*** παρέχεται από το Spring και υλοποιεί το *Front Controller Pattern*, όπου όπως αναφέραμε ένας Front Controller λαμβάνει όλα τα requests και τα στέλνει προς τους υπεύθυνους Controllers

Dispatcher Servlet (2)



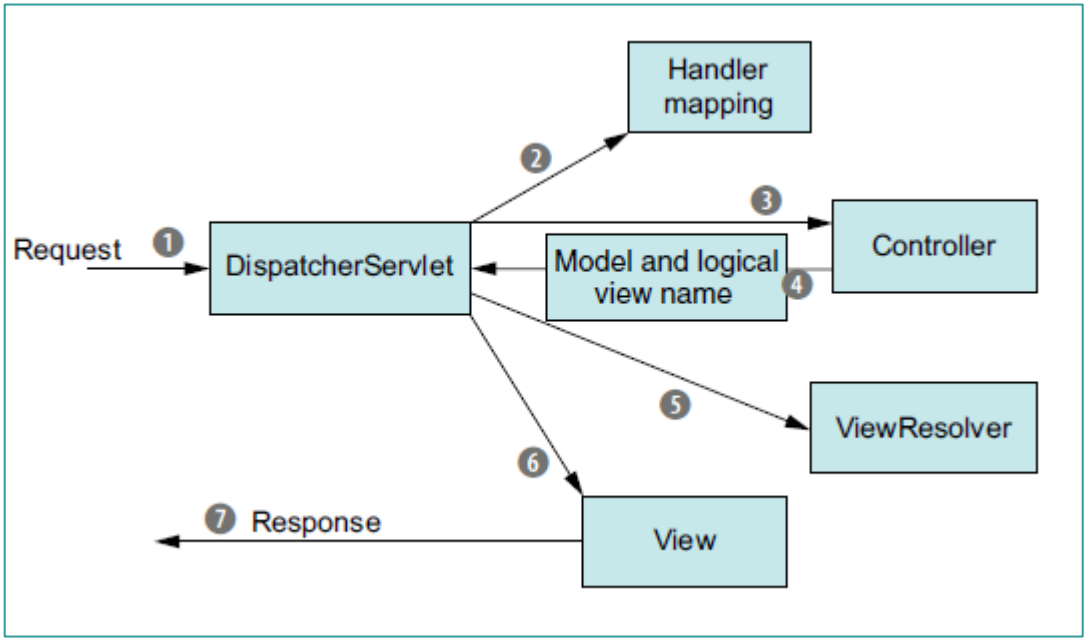
- Εδώ βλέπουμε τα Javadoc του Spring για το Dispatcher Servlet. Γενικά στο **docs.spring.io** είναι όλη η τεκμηρίωση του Spring
- Το **Spring Boot κάνει αυτόματα όλο το configuration** για τον Dispatcher Controller. Κάθε controller πρέπει να συνδεθεί με μία κλάση –την κλάση του Controller- καθώς και να συνδεθεί με κάποιο URL Mapping στο web.xml αρχείο. Αυτό γίνεται πλέον αυτόματα από το Spring Boot, προγραμματιστικά με Java και όχι στο web.xml
- **By default το Spring-boot-starter-web χρησιμοποιεί embedded Apache Tomcat και κάνει configure το *DispatcherServlet* στο URL mapping "/"**

Αρχιτεκτονική Spring MVC (1)



- Στο βήμα (1) ο ***DispatcherServlet*** δέχεται αιτήματα και τα δρομολογεί, στο βήμα (3) στον κατάλληλο Controller, μέσω του αντίστοιχου request-mapping-resolver (Handler Mapping) στο βήμα (2)
- Ο Controller αποστέλλει πίσω ένα view (JSP ή HTML) δηλώνοντας ένα λογικό όνομα (π.χ. index.html) στο βήμα (4). Στο βήμα (5) ο αντίστοιχος name-resolver (ViewResolver) αντιστοιχεί το λογικό όνομα στο φυσικό όνομα του αρχείου.
- Στο βήμα (6) ο ***DispatcherServlet*** βρίσκει το φυσικό view και το επιστρέφει στο Response στο βήμα (7)
- Εναλλακτικά αν χρησιμοποιούμε Rest Controller, επιστρέφουμε data (π.χ. JSON) και όχι σελίδα

Αρχιτεκτονική Spring MVC (2)



- Στα βήματα (2) και (3) για να γίνει το routing χρησιμοποιούμε ειδικά annotations στους Controller και στις μεθόδους μας (π.χ. *@Controller*, *@RestController*, *@GetMapping*, *@PostMapping*)
- Στα βήματα (4), (5), (6) για να γίνει το logical-name-mapping πρέπει απλά ο Controller να κάνει return το όνομα του αρχείου (π.χ. return "index" για index.html)

Response (1)

- Όσο αφορά το τι μπορούμε να επιστρέψουμε ως Response όπως έχουμε πει μπορεί να έχει δύο μορφές:
  - είτε html αρχείο όταν απευθυνόμαστε σε ανθρώπους (human-readable), ή
  - ή JSON / XML όταν επιστρέφουμε δεδομένα

Response (2)

- Τα JSP αρχεία όπως έχουμε δει έχουν μία πολυπλοκότητα λόγω του Taglib με τα ειδικά tags, τα οποία δεν είναι κοντά στην HTML αλλά πιο κοντά στην Java. Επίσης δεν υποστηρίζουν jar αρχεία, αλλά μόνο war
- Περισσότερο αποδοτικό είναι το **Thymeleaf** που είναι server-side Java template engine και παρέχει όχι ειδικά tags αλλά **απλές ιδιότητες στα ήδη υπάρχοντα HTML tags** με αποτέλεσμα να είναι πιο κοντά στην HTML και να παρέχει επομένως ***natural templates***

Response (3) Thymeleaf – Spring EL

- Επιπλέον το Thymeleaf όταν συνδυάζεται με το Spring μπορούμε να χρησιμοποιήσουμε την **Spring Expression Language (Spring EL / SpEL)** για να εισάγουμε δεδομένα (context variables) στα HTML Templates, δεδομένα που ονομάζονται **Model attributes** στην γλώσσα του Spring μιας και η κλάση **Model** στο Spring, όπως θα δούμε, είναι το αντίστοιχο του *RequestDispatcher* στην JEE
- Αυτό μας επιτρέπει να κάνουμε εύκολα **data binding μεταξύ των φορμών (View) και των Controllers**

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'
    implementation 'org.springframework.boot:spring-boot-starter-web'
```

Response (4)

- Όσο αφορά το REST/JSON, στο Spring η μετατροπή γίνεται αυτόματα όταν χαρακτηρίσουμε ένα Controller ως **@RestController και κάνουμε return ένα instance ή Collection της εφαρμογής μας**
- Η αυτόματη μετατροπή γίνεται, με **Jackson** (Java Library για την μετατροπή από Java objects σε JSON και το αντίθετο)
- Το **spring-boot-starter-web** dependency εισάγει και τα Jackson libraries

Embedded vs External Web Server (1)

- Έχουμε δει στο προηγούμενο JAX-RS project που αναπτύξαμε ότι **έπρεπε πρώτα να εγκαταστήσουμε τον Apache Tomcat** ως τον Container της Web εφαρμογής μας
- Το μειονέκτημα αυτής της προσέγγισης είναι ότι ο 'πελάτης' θα πρέπει να εγκαταστήσει τον δικό του Apache Tomcat και το παραγόμενο αρχείο πρέπει να είναι σε μορφή war ώστε ο «πελάτης» να το παραλάβει να το εισάγει στο root folder του δικού του Apache Tomcat που θα πρέπει όπως είπαμε να έχει ήδη εγκαταστήσει

Embedded vs External Web Server (2)

- Για μεγαλύτερη ευελιξία, ιδιαίτερα όταν λειτουργούμε σε περιβάλλοντα cloud όπως *Docker/microservices* αλλά και σε περιβάλλοντα όπου δεν απαιτείται από τον «πελάτη» η εγκατάσταση web server, μπορούμε με το Spring να δημιουργήσουμε ένα embedded Web Server, δηλαδή ένα Apache Tomcat (ή άλλο container) που να ξεκινά αυτόματα με την εφαρμογή μας
- Με αυτό τον τρόπο **μπορεί το παραγόμενο αρχείο της εφαρμογής μας να είναι jar** (και όχι war)

Embedded vs External Web Server (3)

- Το Servlet API της Java παραδοσιακά επικεντρωνόταν στο *external web server model*
- Ωστόσο τα τελευταία χρόνια και ιδιαίτερα με την έλευση των microservices σε πλατφόρμες όπως Docker, ***το embedded web server model είναι πολύ δημοφιλές***

Embedded Web Server - Spring

- Με το spring-boot-starter-web dependency εισάγεται και η embedded έκδοση του Tomcat

```
21  dependencies {
22      implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'
23      implementation 'org.springframework.boot:spring-boot-starter-web'
24  //    implementation 'org.springframework.boot:spring-boot-starter-tomcat'
25
```

Start spring io

- Το https://start.spring.io/ είναι μία εφαρμογή του Spring για να **δημιουργούμε τη δομή ενός Spring Project** και μετά να κάνουμε import είτε στο Eclipse ή στο IntelliJ τη δομή των φακέλων μέσα από το .zip file που δημιουργείται
- Στο **start.spring.io** θα δημιουργήσουμε ένα αρχικό set-up της εφαρμογής μας και θα δημιουργηθεί ένα zip αρχείο με την δομή του project μας το οποίο θα κάνουμε import στο Eclipse και στο IntelliJ community edition
- Θα δημιουργήσουμε ένα **απλό model, controllers και μία HTML page με το Thymeleaf**

Νέο Project

start.spring.io

spring initializr

Project
- Gradle - Groovy
- Gradle - Kotlin
- Maven

Language
- Java
- Kotlin
- Groovy

Spring Boot
- 3.4.0 (SNAPSHOT)
- 3.4.0 (M3)
- 3.3.5 (SNAPSHOT)
- 3.3.4
- 3.2.11 (SNAPSHOT)
- 3.2.10

Project Metadata

Group	gr.aueb.cf
Artifact	hello-world
Name	hello-world
Description	Demo project for Spring Boot
Package name	gr.aueb.cf.hello-world
Packaging	Jar / War

Dependencies — ADD DEPENDENCIES... CTRL + B

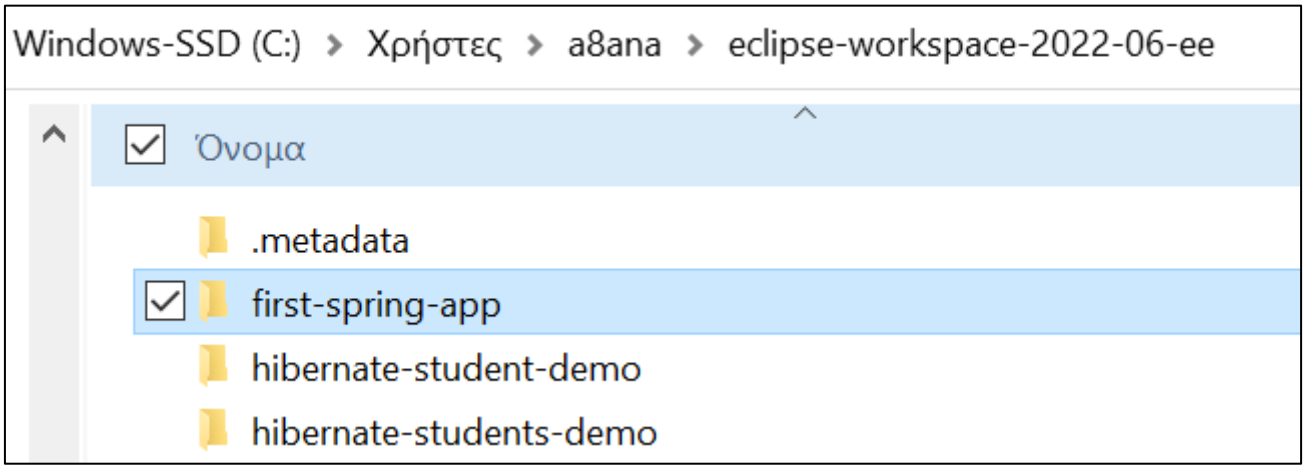
Thymeleaf TEMPLATE ENGINES
A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

GENERATE CTRL + ⏎ | EXPLORE CTRL + SPACE | SHARE...

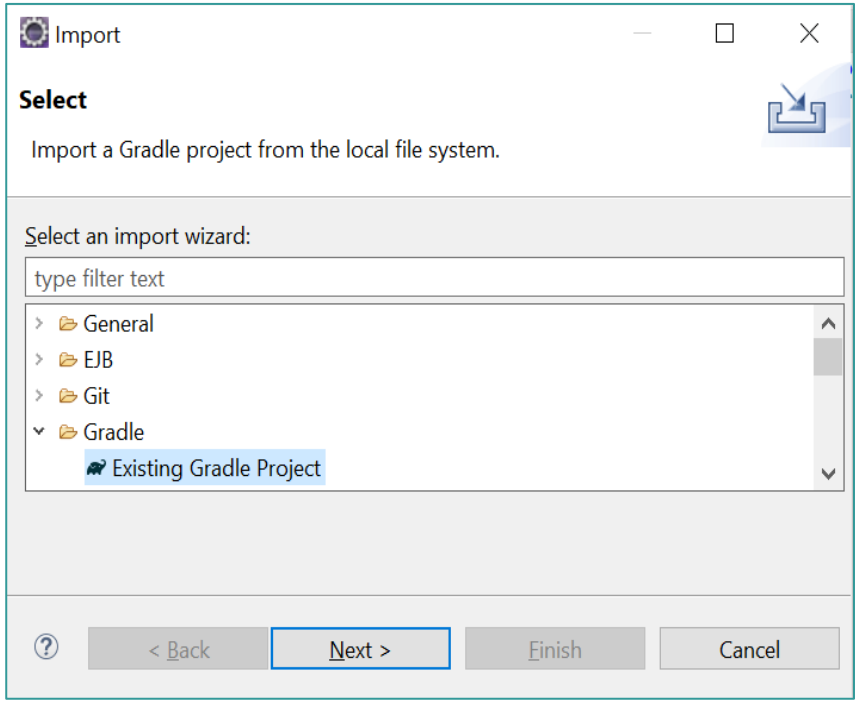
- Κάνουμε τις ρυθμίσεις και πατάμε Generate. Όπως βλέπετε χρησιμοποιούμε **Gradle** ενώ εισάγουμε το **spring-boot-starter-web και το Thymeleaf**. Μπορούμε να εισάγουμε και αργότερα κατευθείαν στο build.gradle

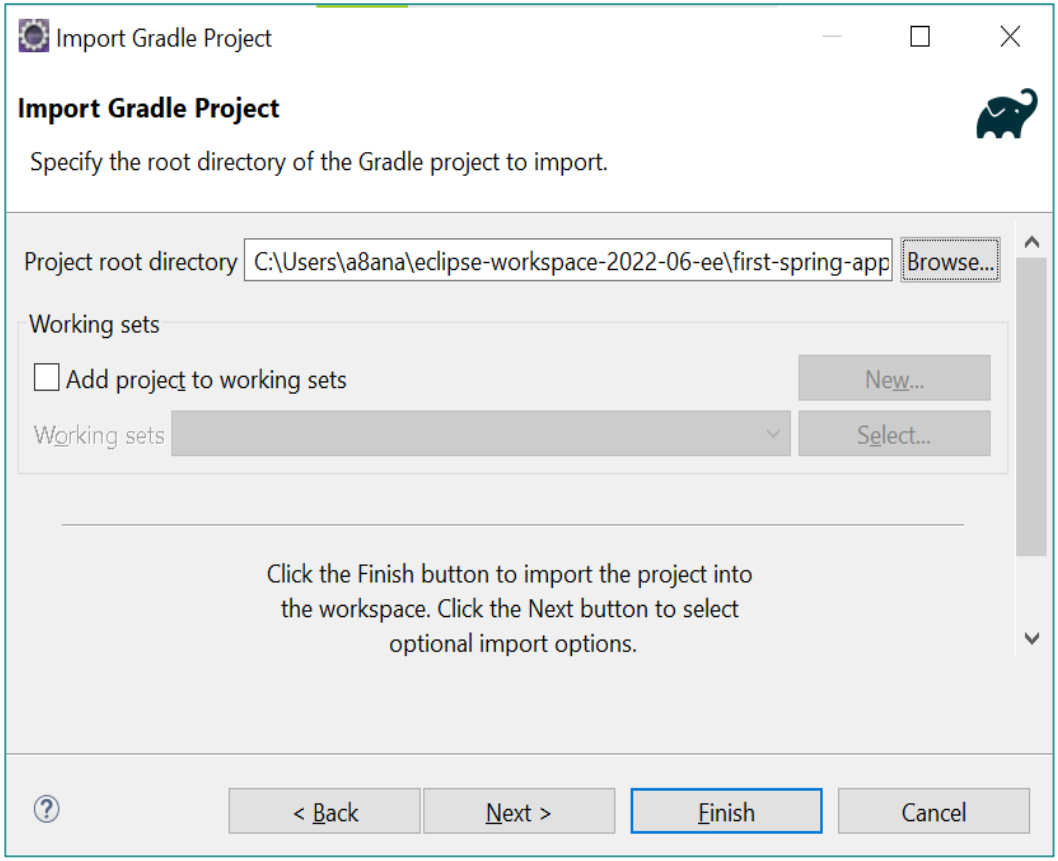
Extract



- Κάνουμε extract το φάκελο first-spring-app μέσα στο workspace του Eclipse

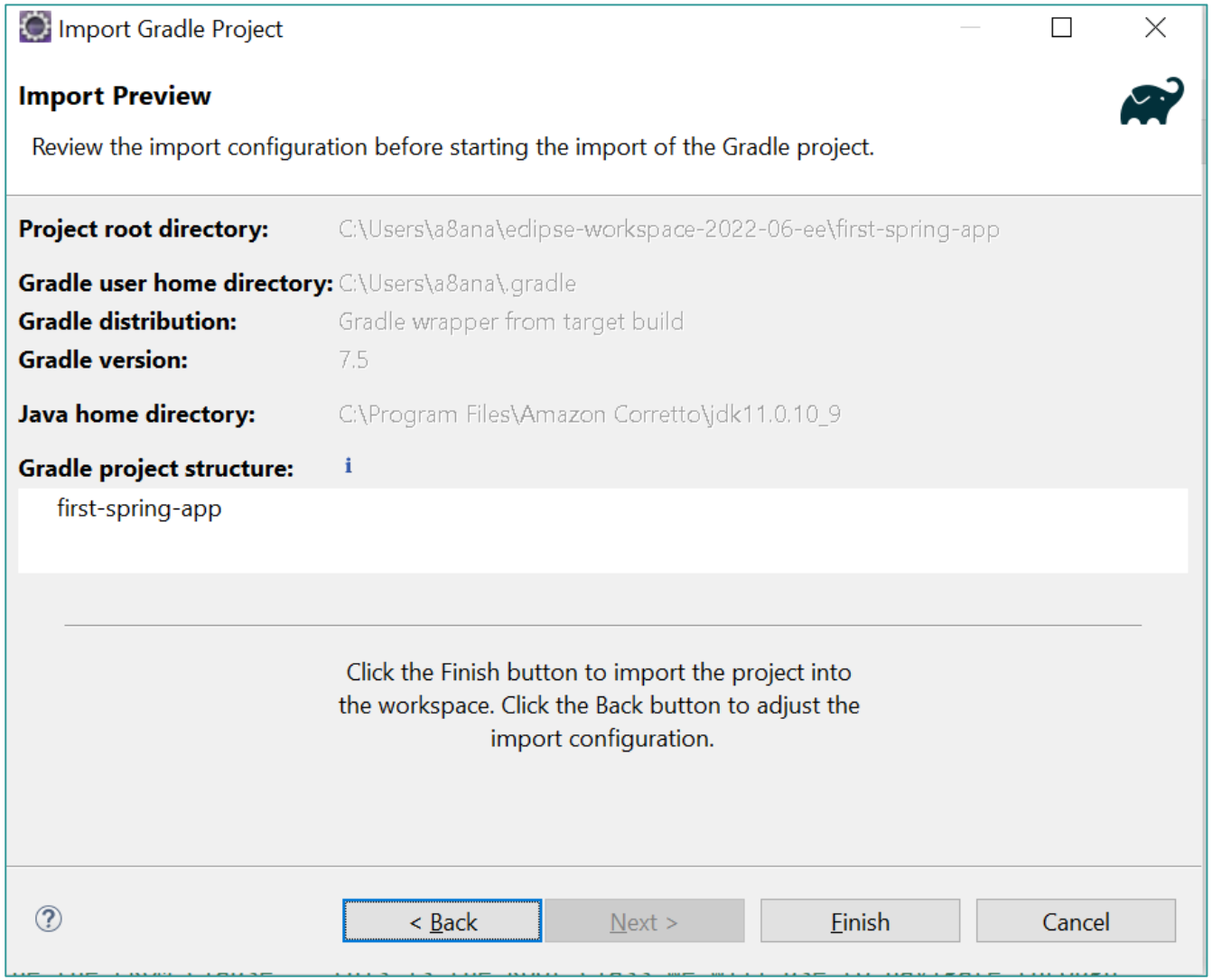
Δημιουργία νέου Project με import





- Αφού **(1)** έχουμε κάνει extract από το zip τον folder, first-spring-app, μέσα στο eclipse-workspace, στη συνέχεια **(2)** κάνουμε import ένα Existing Gradle Project και **(3)** επιλέγουμε ως project root directory, το φάκελο που έχουμε κάνει extract

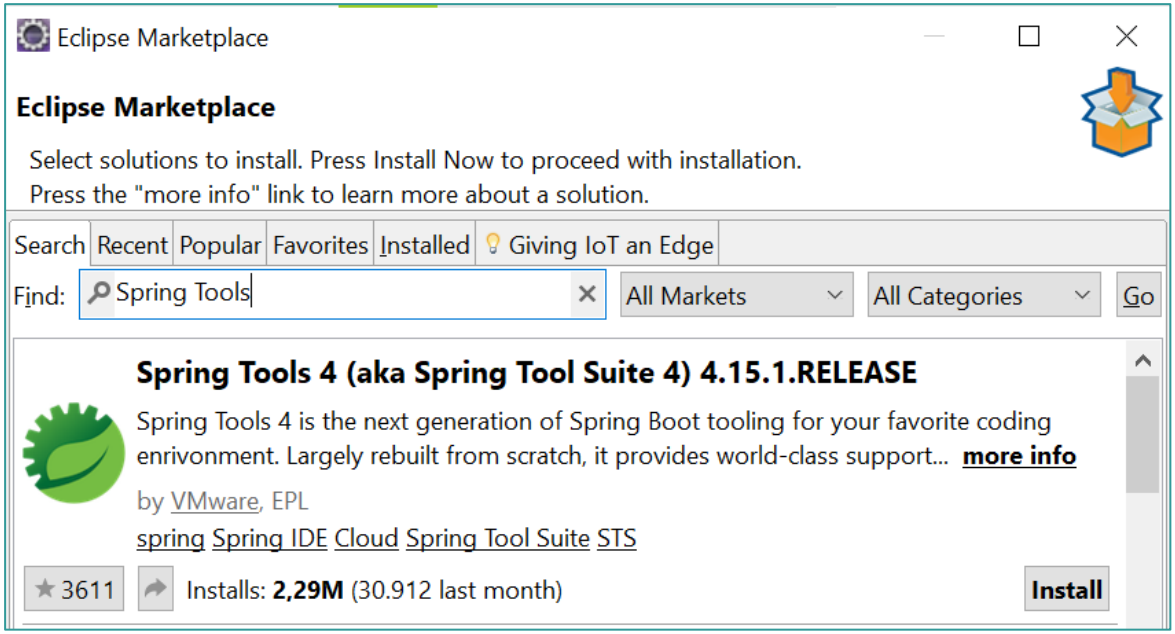
Import Wizard



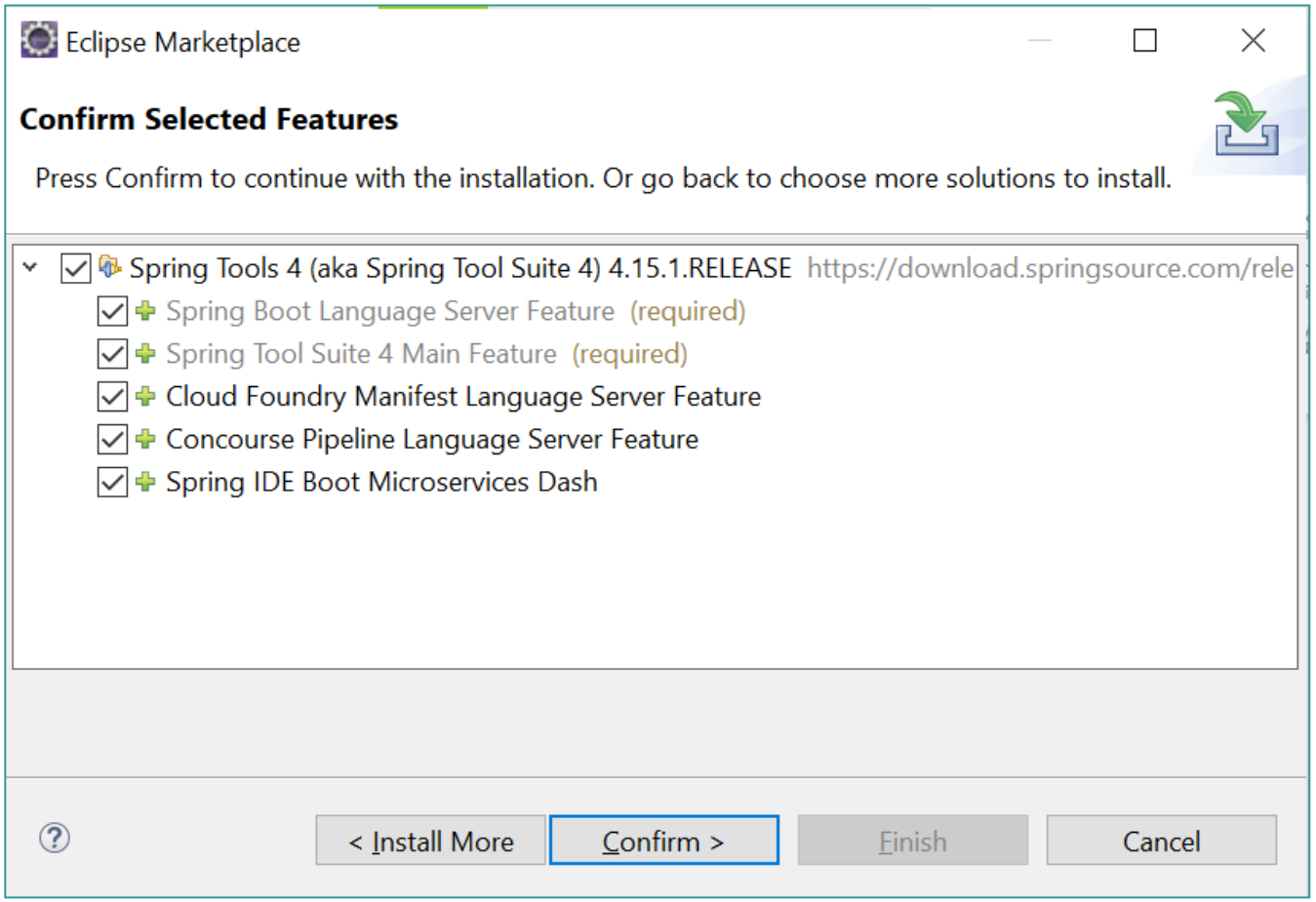
- Ο Wizard βρίσκει αυτόματα το Gradle και το Java Home

Eclipse Marketplace

- Εναλλακτικά, Από το *Eclipse* αφού εγκαταστήσουμε μέσω του Eclipse Marketplace τα Spring Tools

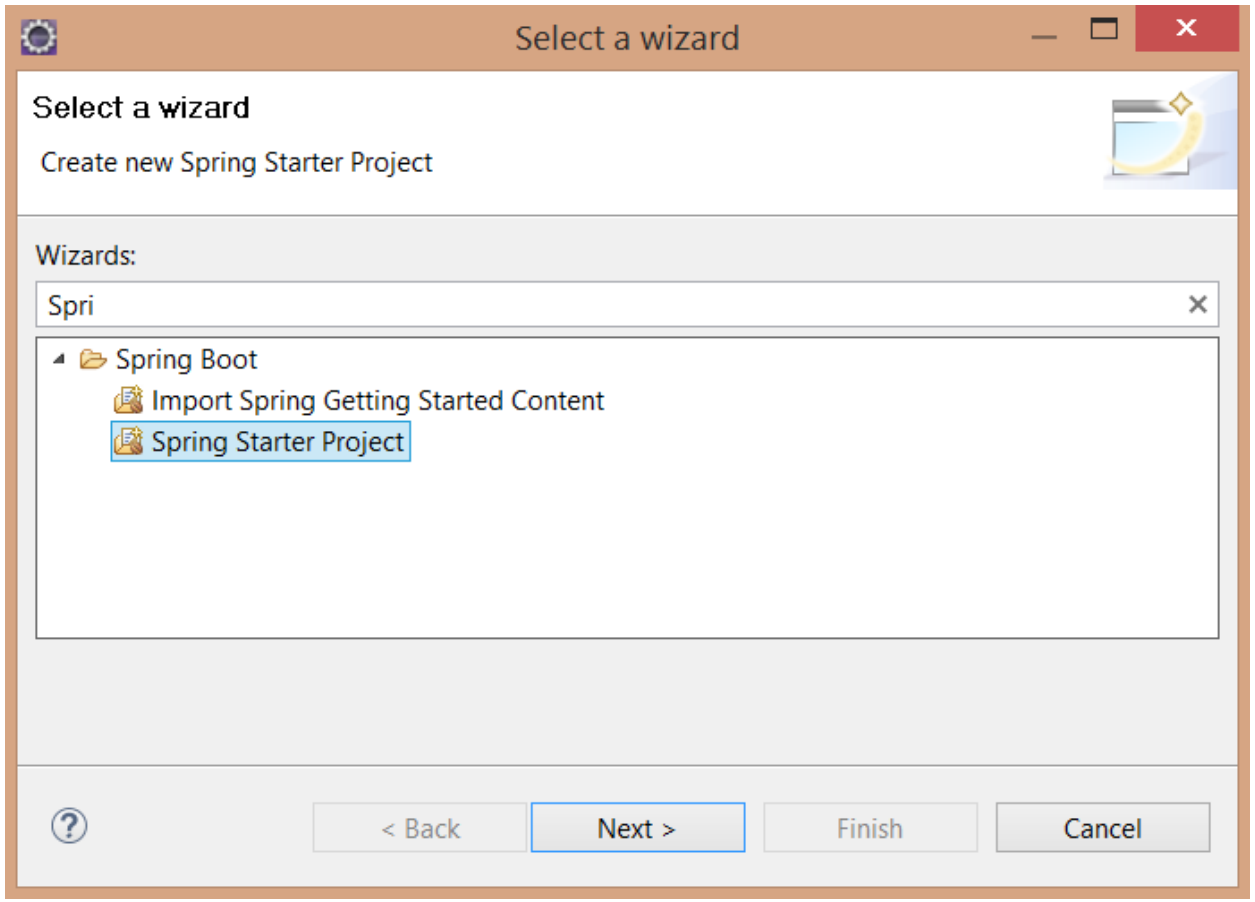


Eclipse Marketplace Spring Tools



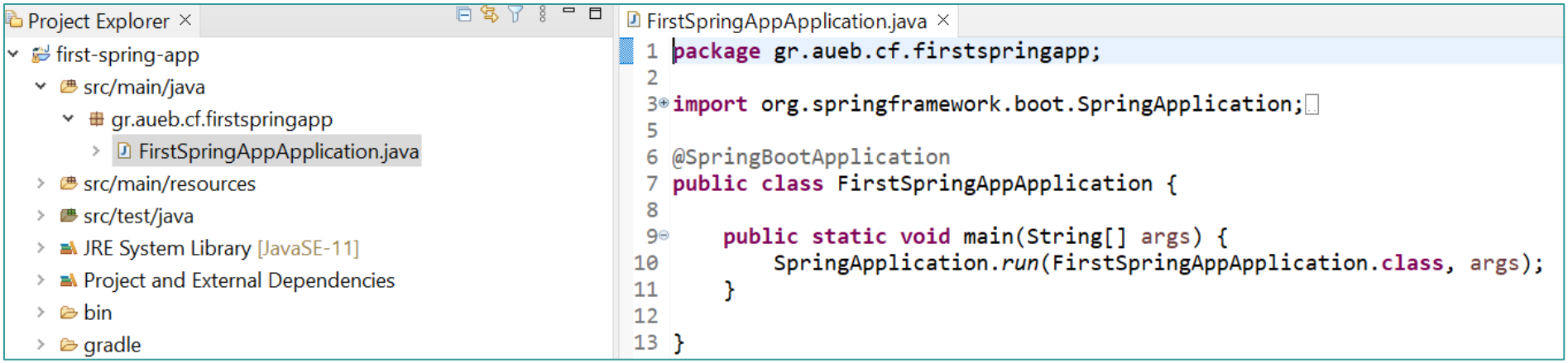
- Confirm / Accept & Finish

New Spring Starter Project



- File / New Project και αναζητούμε για Spring

FirstSpringAppApplication



- Το **FirstSpringAppApplication** που περιέχει τη *main()* δημιουργήθηκε από το Spring Boot. Περιέχει μία μόνο γραμμή κώδικα με τη **static μέθοδο run** της **SpringApplication** class (δείτε επόμενη διαφάνεια)
- Τρέχει με δεξί κλικ στο Project και **Run As Java Application**
- Το Spring Boot 3.2.* by default περιέχει και χρησιμοποιεί Tomcat 10.1

War & Embedded Web Server


```
plugins {
    id 'org.springframework.boot' version '3.2.1'
    id 'io.spring.dependency-management' version '1.0.12.RELEASE'
    id 'java'
    id 'war'
    id 'eclipse-wtp'
}

group = 'gr.aueb.cf'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '11'

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    providedRuntime 'org.springframework.boot:spring-boot-starter-tomcat'
}
```

- 1. id ‘war’ στα plugins
- 2. Ο Tomcat περιλαμβάνεται στο Spring Boot με το starter-web. Στα dependencies δίνουμε το tomcat για runtime (providedRuntime) ώστε θα τρέχει με το Gradle αλλά θα μπορεί να γίνει deploy το war και σε installed web server (Servlet Container) χωρίς interfere με το Servlet Container που θα γίνει deploy το war

Servlet Initializer


```
1  package gr.aueb.cf;
2
3  import org.springframework.boot.builder.SpringApplicationBuilder;
4  import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;
5
6  public class ServletInitializer extends SpringBootServletInitializer {
7
8      @Override
9      protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {
10         return application.sources(FirstSpringAppApplication.class);
11     }
12
13 }
```

- Αυτή η κλάση χρειάζεται όταν χρησιμοποιούμε war και η εφαρμογή μας τρέχει σε servlet container

Run war

Windows PowerShell

```
PS C:\Users\a8ana\eclipse-workspace-2022-06-ee\first-spring-app\build\libs> java -jar .\first-spring-app-0.0.1-SNAPSHOT.war

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.7.2)

2022-07-27 22:23:45.062  INFO 18228 --- [           main] gr.aueb.cf.FirstSpringAppApplication     : Starting FirstSpringAppAppl
ication using Java 11.0.10 on thanassis-pc with PID 18228 (C:\Users\a8ana\eclipse-workspace-2022-06-ee\first-spring-app\build\li
bs\first-spring-app-0.0.1-SNAPSHOT.war started by a8ana in C:\Users\a8ana\eclipse-workspace-2022-06-ee\first-spring-app\build\li
bs)
2022-07-27 22:23:45.066  INFO 18228 --- [           main] gr.aueb.cf.FirstSpringAppApplication     : No active profile set, fall
ing back to 1 default profile: "default"
2022-07-27 22:23:46.064  INFO 18228 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with por
t(s): 8080 (http)
2022-07-27 22:23:46.078  INFO 18228 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2022-07-27 22:23:46.078  INFO 18228 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [A
pache Tomcat/9.0.65]
2022-07-27 22:23:46.497  INFO 18228 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedde
d WebApplicationContext
2022-07-27 22:23:46.497  INFO 18228 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext:
 initialization completed in 1365 ms
2022-07-27 22:23:46.711  INFO 18228 --- [           main] o.s.b.a.w.s.WelcomePageHandlerMapping    : Adding welcome page templat
e: index
2022-07-27 22:23:46.860  INFO 18228 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s):
8080 (http) with context path ''
2022-07-27 22:23:46.869  INFO 18228 --- [           main] gr.aueb.cf.FirstSpringAppApplication     : Started FirstSpringAppAppli
cation in 2.222 seconds (JVM running for 2.626)
2022-07-27 22:23:53.678  INFO 18228 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring Dispatc
herServlet 'dispatcherServlet'
2022-07-27 22:23:53.679  INFO 18228 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispa
tcherServlet'
2022-07-27 22:23:53.683  INFO 18228 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in
 1 ms
```

Servlet Container

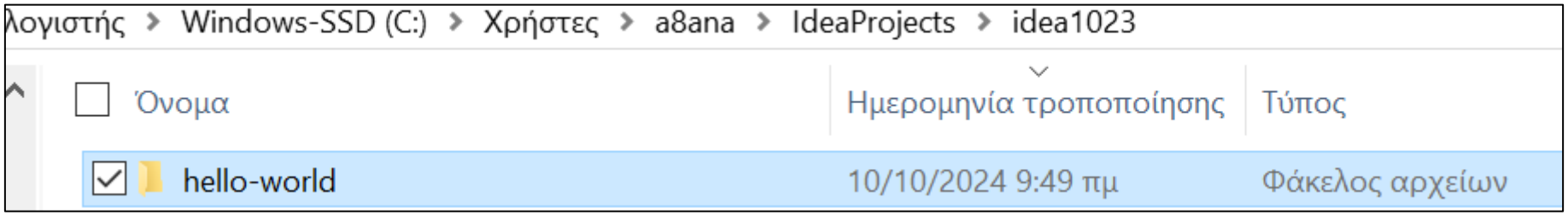
- Αν έχουμε εγκατεστημένο Apache Tomcat μπορούμε να κάνουμε απλά copy/paste το war (έστω *hello.war*) αρχείο μας στον φάκελο webapps του Tomcat
- Ο Apache αυτόματα θα κάνει deploy (δημιουργήσει) τον φάκελο hello και η εφαρμογή μας θα τρέχει στο localhost:8080/hello
- Επειδή ο Apache δημιουργεί φακέλους θα πρέπει να έχει και τα αντίστοιχα δικαιώματα και αν δεν τα έχει θα πρέπει να δοθούν (αλλιώς δεν μπορεί να δημιουργεί φακέλους)

Run jar


```
PS C:\Users\a8ana\eclipse-workspace-2022-06-ee\first-spring-app\build\libs> java -jar .\first-spring-app-0.0.1-SNAPSHOT.jar

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.7.2)

2022-07-27 22:37:31.102  INFO 8852 --- [           main] gr.aueb.cf.FirstSpringAppApplication     : Starting FirstSpringAppAppli
cation using Java 11.0.10 on thanassis-pc with PID 8852 (C:\Users\a8ana\eclipse-workspace-2022-06-ee\first-spring-app\build\libs
\first-spring-app-0.0.1-SNAPSHOT.jar started by a8ana in C:\Users\a8ana\eclipse-workspace-2022-06-ee\first-spring-app\build\libs
)
2022-07-27 22:37:31.105  INFO 8852 --- [           main] gr.aueb.cf.FirstSpringAppApplication     : No active profile set, falli
ng back to 1 default profile: "default"
2022-07-27 22:37:32.064  INFO 8852 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port
(s): 8080 (http)
2022-07-27 22:37:32.081  INFO 8852 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2022-07-27 22:37:32.082  INFO 8852 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Ap
ache Tomcat/9.0.65]
2022-07-27 22:37:32.162  INFO 8852 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded
 WebApplicationContext
2022-07-27 22:37:32.162  INFO 8852 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext:
initialization completed in 993 ms
2022-07-27 22:37:32.378  INFO 8852 --- [           main] o.s.b.a.w.s.WelcomePageHandlerMapping    : Adding welcome page template
: index
2022-07-27 22:37:32.528  INFO 8852 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8
080 (http) with context path ''
2022-07-27 22:37:32.539  INFO 8852 --- [           main] gr.aueb.cf.FirstSpringAppApplication     : Started FirstSpringAppApplic
ation in 1.864 seconds (JVM running for 2.258)
```

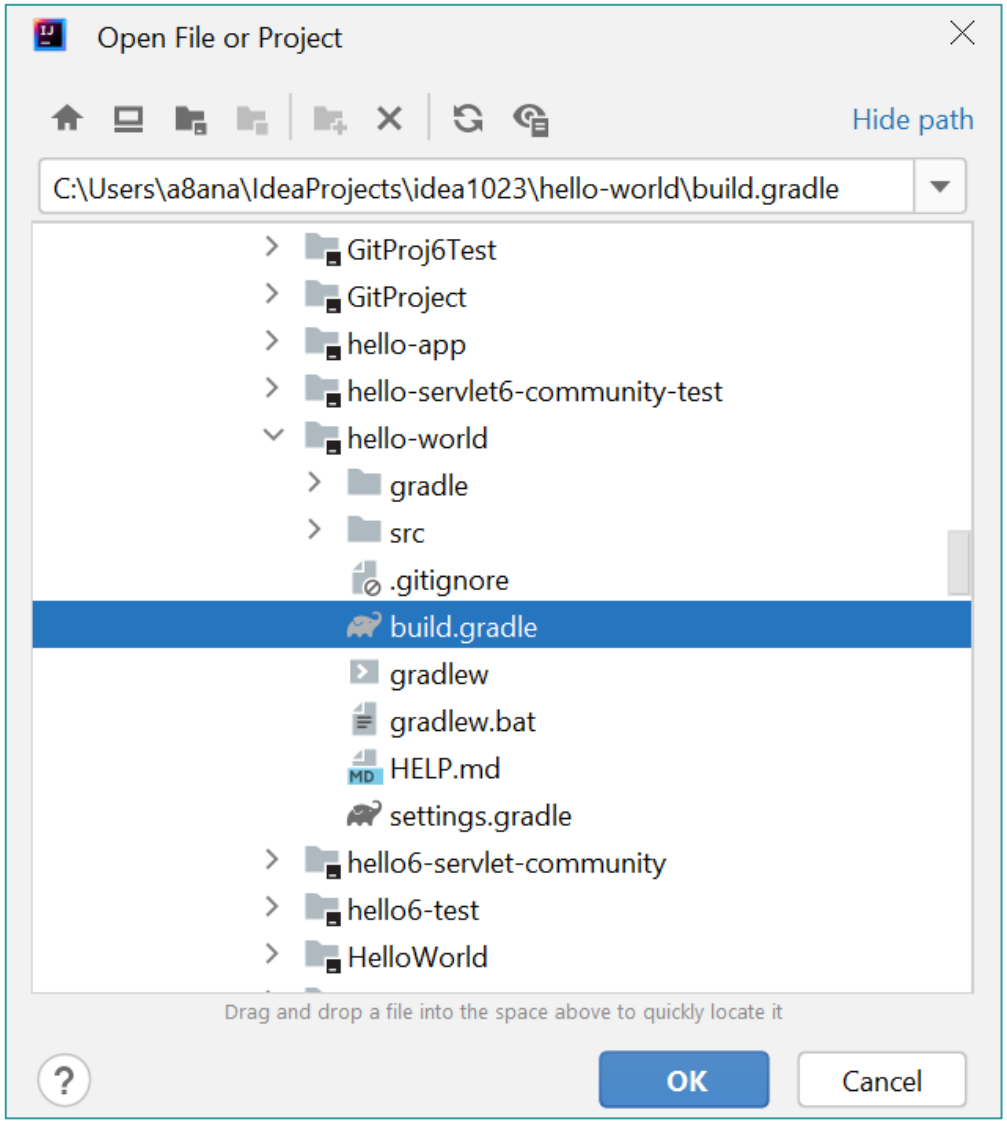
- Αν δεν συμπεριλάβουμε τα Servlet Initializer, id ‘war’ και providedRuntime για τον Tomcat, by default το Gradle δίνει εκτελέσιμο jar που πάλι μπορούμε να τρέξουμε

IntelliJ Community Edition



- Για το IntelliJ community edition πάλι δημιουργούμε το project στο start.spring.io και μετά extract σε ένα φάκελο όπως παραπάνω

File / Open



- Επιλέγουμε **File / Open και μετά το build.gradle και OK**
- Το project γίνεται import
- Σε περίπτωση προβλήματος πάμε στο **File / Project Structure και Project Settings / Project** και αλλάζουμε το SDK σε Java 17

Static index.html

hello-world 〉 src 〉 main 〉 resources 〉 static 〉 index.html

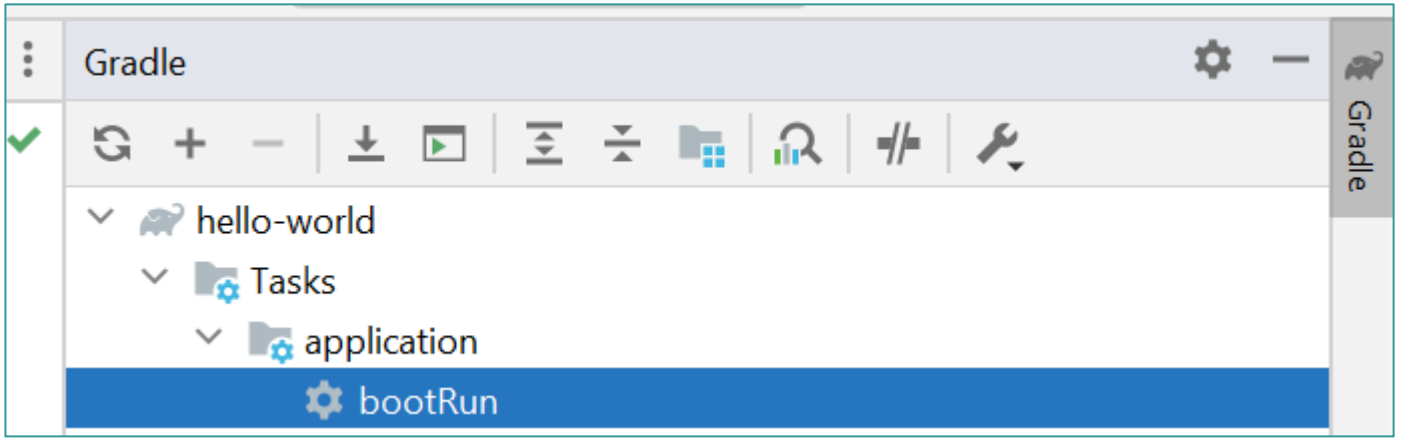
Project

- .gradle
- .idea
- build
- gradle
- src
  - main
    - java
      - gr
        - aueb
          - cf
            - hello_world
              - HelloWorldApplication
    - resources
      - static
        - index.html
      - templates
      - application.properties

HelloWorldApplication.java | index.html

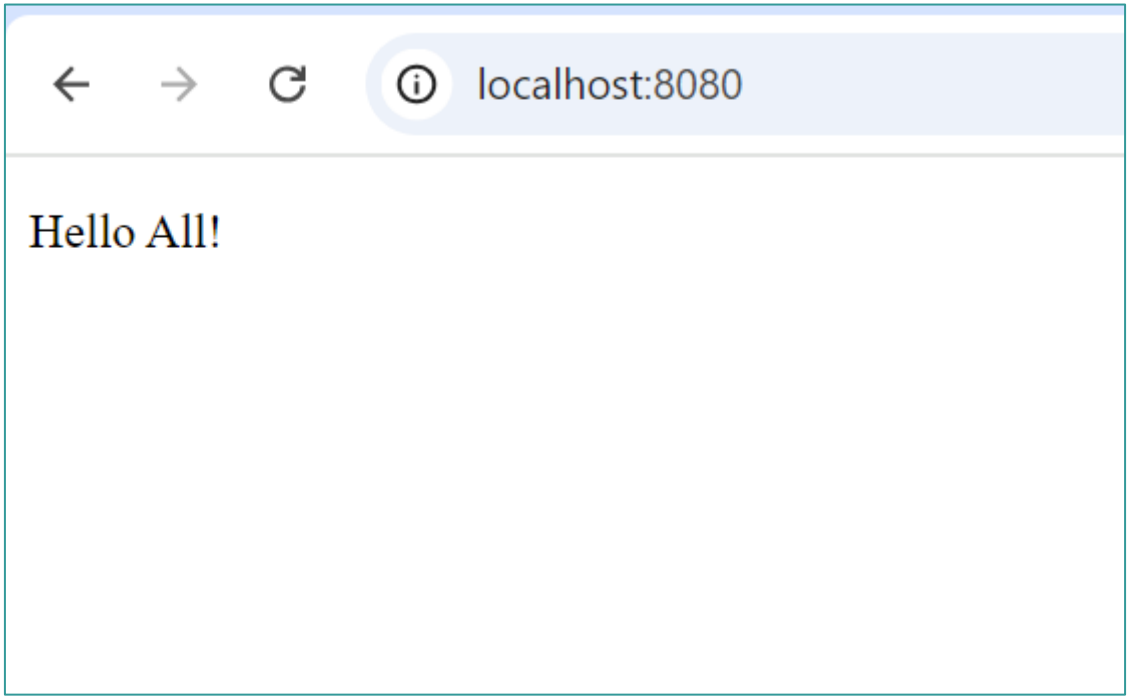
```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <title>Title</title>
6   </head>
7   <body>
8       <p>Hello All!</p>
9   </body>
10  </html>
```

Deployment

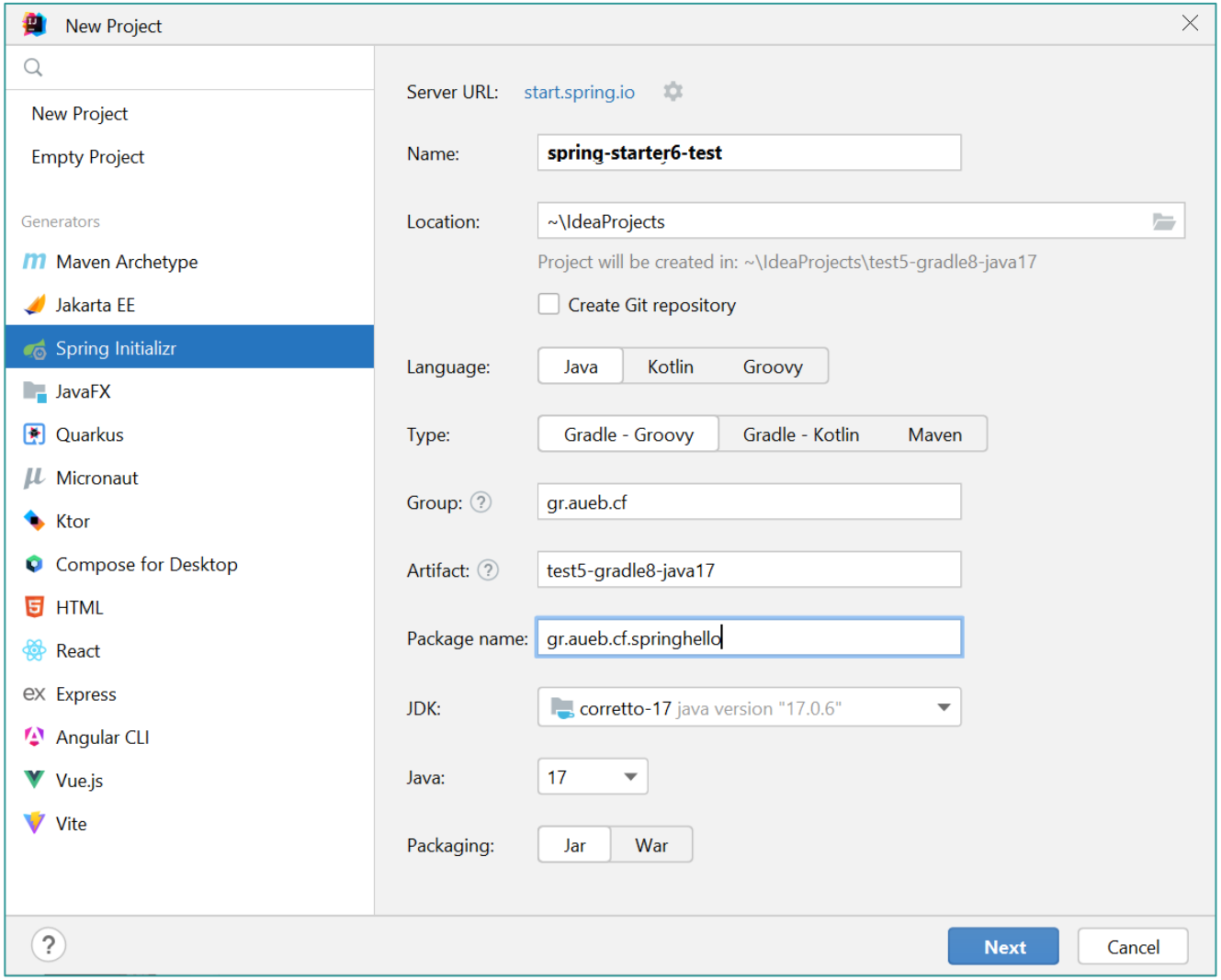


- Εκτελούμε από το gradle plugin του IntelliJ ή και κυρίως από command line με ./gradlew bootRun

Αποτέλεσμα

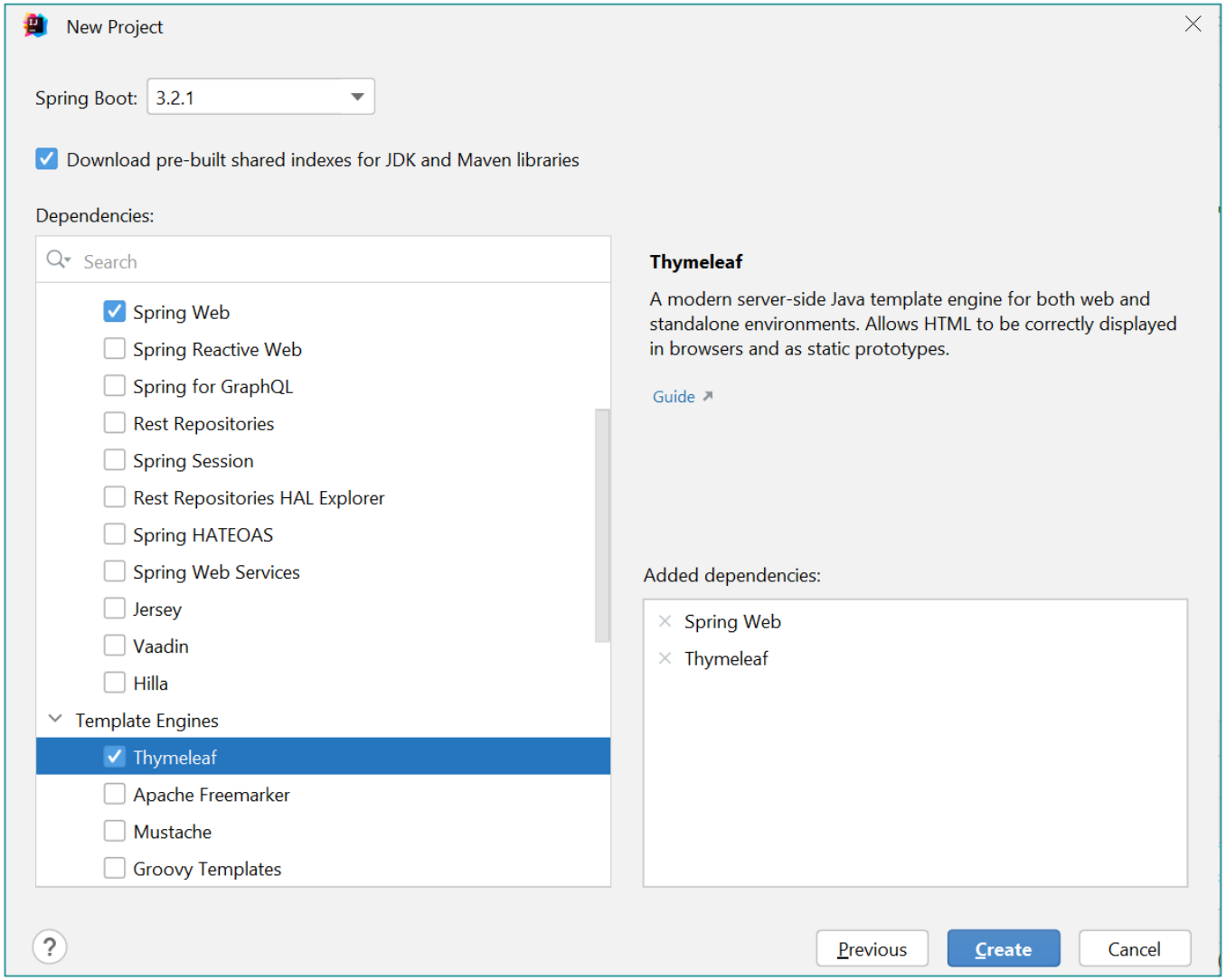


IntelliJ Ultimate Edition



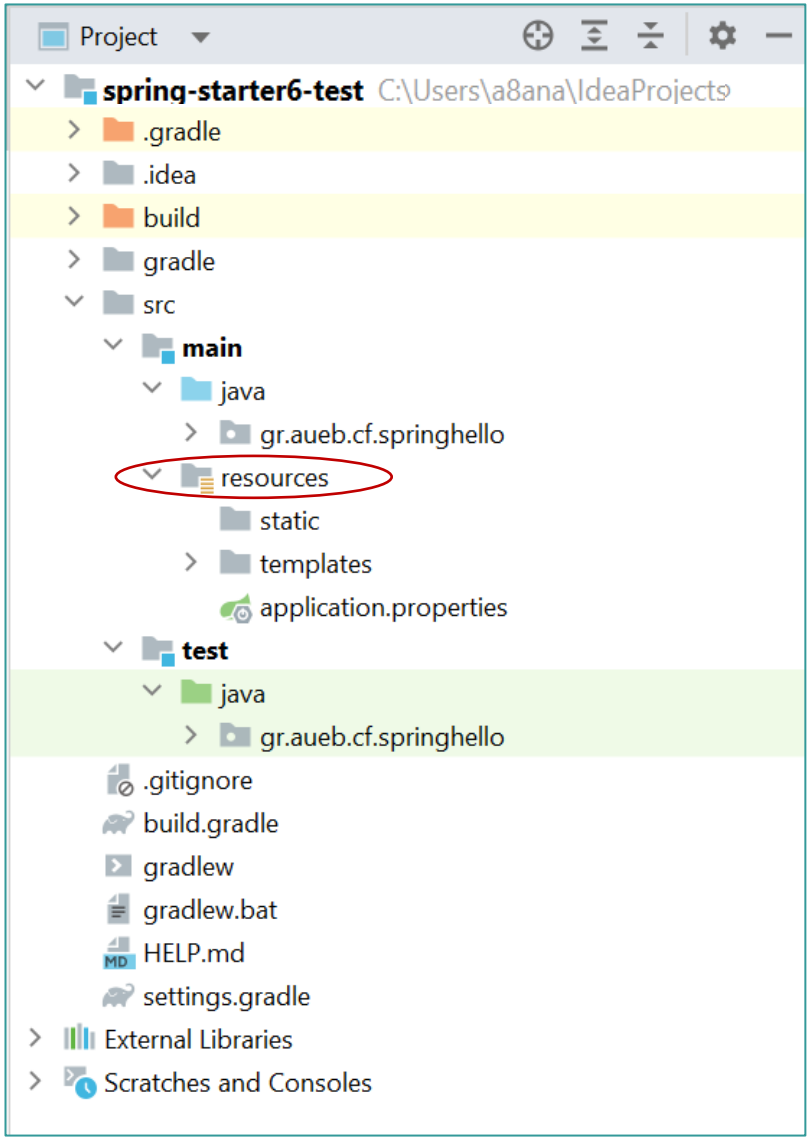
- Ξεκινάμε ένα Spring Initializr project
- Με Gradle, Java 17 και Jar packaging

Setup Dependencies



- Το Spring Boot 3.0 υποστηρίζει τουλάχιστον Java 17
- Εισάγουμε starter-web και starter-Thymeleaf

Αρχική Δομή Project



- Τυπική Maven ή Gradle δομή: source code **src/main/java**, test code στο **src/test/java**, non-Java resources στο **src/main/resources**
- **Στο φάκελο resources:**
  - ***application.properties***: Αρχικά είναι empty, αλλά χρησιμοποιείται για configuration properties
  - ***static***: folder για static content (images, stylesheets, JavaScript κ.λπ.)
  - ***templates***: folder για .html template files, όπως Thymeleaf templates

Build.gradle legacy


```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.2.1'
    id 'io.spring.dependency-management' version '1.1.4'
}

group = 'gr.aueb.cf'
version = '0.0.1-SNAPSHOT'

java {
    sourceCompatibility = '17'
}

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
    mavenCentral()
}

dependencies { Edit Starters...
    implementation 'org.springframework.boot:spring-boot-starter-web'
    compileOnly 'org.projectlombok:lombok'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
}

tasks.named('test', Test) { Test it ->
    useJUnitPlatform()
}
```

- Το io.spring.dependency-management οργανώνει αυτόματα τα versions των dependencies (δεν χρειάζεται να δίνουμε version numbers)
- Το spring boot 3.2.1 ορίζει ότι η version του Spring που χρησιμοποιείται είναι η 6.1.2
- Το 'test' είναι όνομα του task και το 'Test' είναι config κλάση ώστε μαζί με το *useJUnitPlatform* να κάνουν config το testing process με Junit 5

Gradle > 6.7 - toolchain


```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.3.4'
    id 'io.spring.dependency-management' version '1.1.6'
}

group = 'gr.aueb.cf'
version = '0.0.1-SNAPSHOT'

java {
    toolchain { JavaToolchainSpec it ->
        languageVersion = JavaLanguageVersion.of(17)
        vendor = JvmVendorSpec.AMAZON
    }
}

repositories {
    mavenCentral()
}
```

- Το build.gradle στο Gradle είναι όπως το POM.xml στον Maven

Build.gradle


```
17  repositories {
18      mavenCentral()
19  }
20
21  dependencies {  Edit Starters...
22      implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'
23      implementation 'org.springframework.boot:spring-boot-starter-web'
24      testImplementation 'org.springframework.boot:spring-boot-starter-test'
25      testRuntimeOnly 'org.junit.platform:junit-platform-launcher'
26      compileOnly 'org.projectlombok:lombok'
27      annotationProcessor 'org.projectlombok:lombok'
28  }
29
30  tasks.named('test', Test) {  Test it ->
31      useJUnitPlatform()
32  }
```

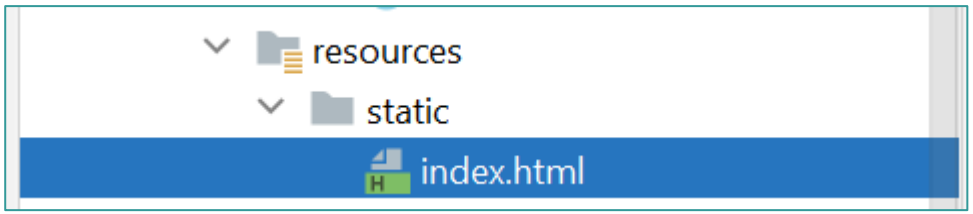
SpringBootApplication


```java
package gr.aueb.cf.springstarter;

import ...

@SpringBootApplication
public class SpringStarterApplication {

    public static void main(String[] args) {
        SpringApplication.run(SpringStarterApplication.class, args);
    }

}
```

- Entry point. Το default path είναι το /

Index.html


```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Hello World</title>
6  </head>
7  <body>
8      <p>Hello Coding!</p>
9  </body>
10 </html>
```



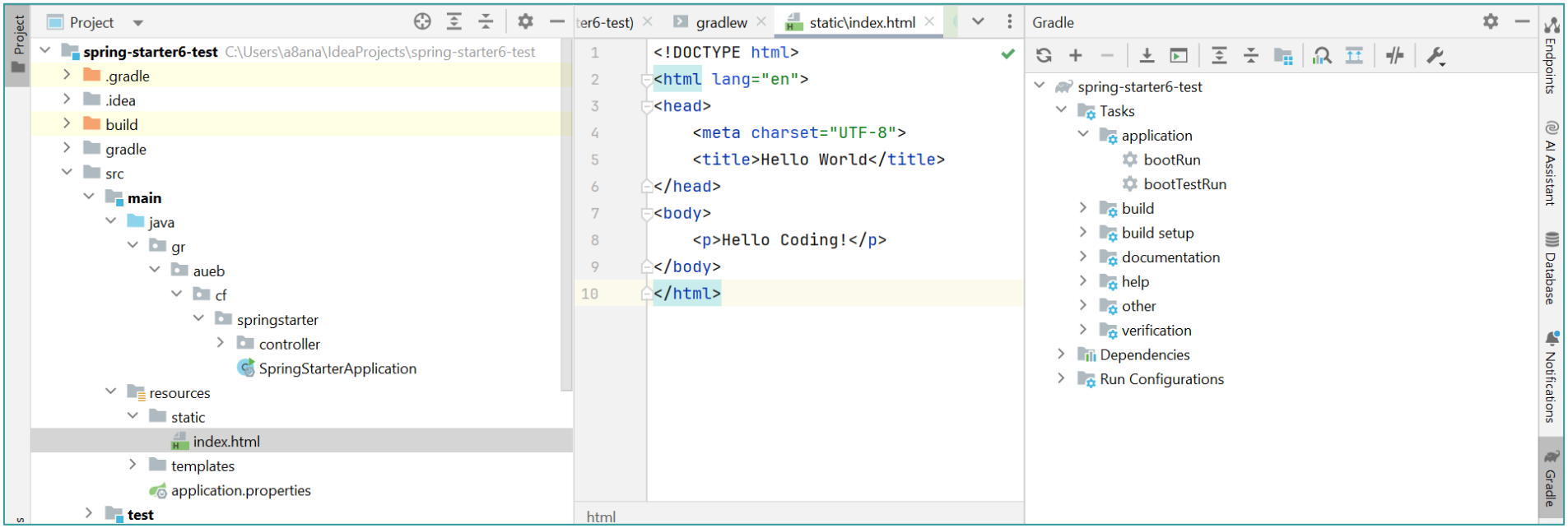
- Στο φάκελο **src/main/resources/static** δημιουργούμε ένα νέο HTML file (index.html)

Main / Run


```
Terminal:  Local ×  Git Bash ×  +  ⌄

a8ana@thanassis-pc MINGW64 ~/IdeaProjects/spring-starter6-test
$ ./gradlew bootRun

> Task :bootRun


  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.3.4)
```

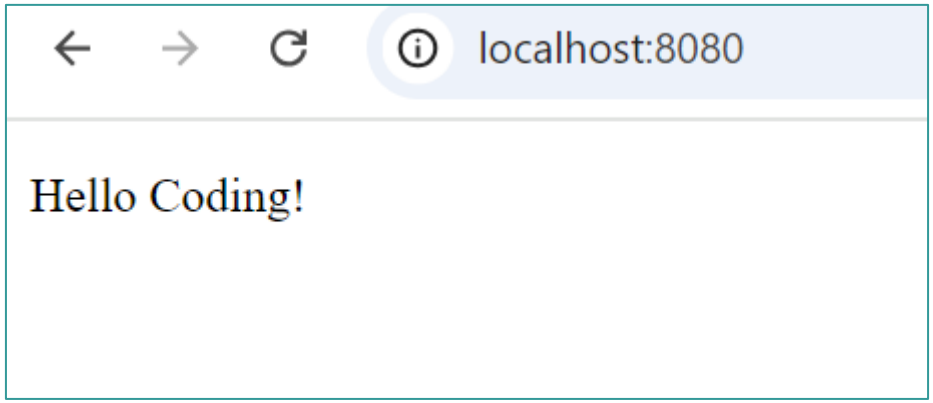
- Με ./gradlew από το terminal εκκινούμε το project

Gradle Tasks



- Εναλλακτικά όπως και στο Community Edition έτσι κι εδώ επιλέγουμε Run Gradle Tasks από το Gradle στο μενού δεξιά

Αποτέλεσμα

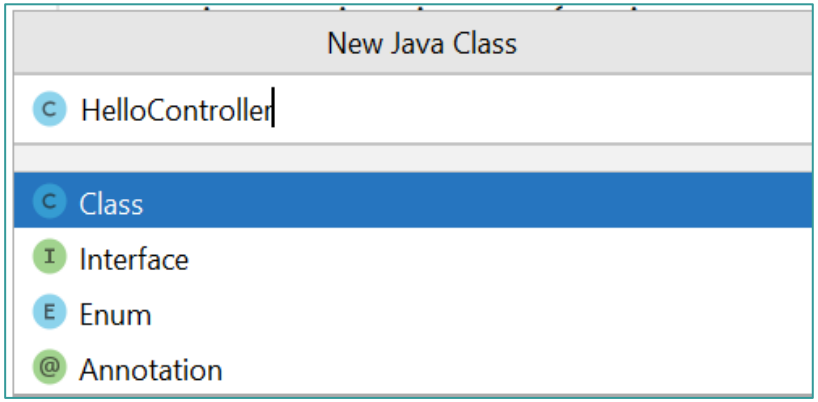


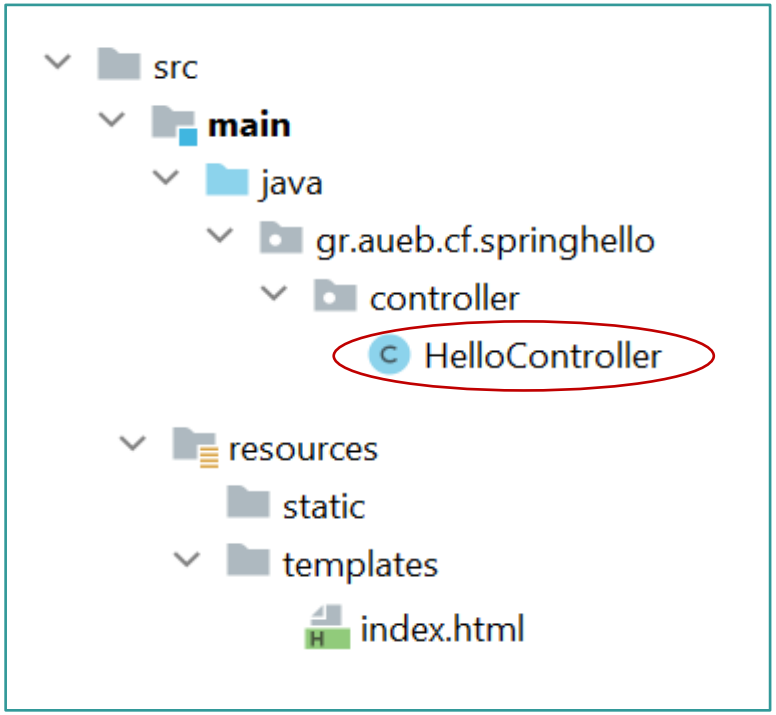
- Εμφανίζεται το index.html είτε ως *localohost:8080/index.html* είτε ως *localhost:8080/*
- Αυτό το content είναι static. Αν θέλουμε να εμφανίσουμε δυναμικό περιεχόμενο με σε σελίδα που έχει code-behind (controller) τότε χρησιμοποιούμε το φάκελο *templates*

Controller / RestController

- Η βασική δομή στο Spring MVC είναι δύο τύποι: 1) Controller ή 2) RestController, που είναι κλάσεις που δέχονται requests και απαντάνε με πληροφορίες κάποιου τύπου
- Στην περίπτωση των browser-facing applications ο **Controller** απαντάει επιστρέφοντας ένα html αρχείο
- Στα RESTful web services ο **RestController** απαντάει επιστρέφοντας ένα Response με status και data

Controller (για Web Page)





- Θα γράψουμε ένα Controller που θα δέχεται requests σε ένα path (π.χ. **/hello/coding-factory**) και θα επιστρέφει ένα index page
- Conceptually, μπορούμε να θεωρούμε αυτό το path, ως τον controller

Controller (1)


```java
package gr.aueb.cf.springstarter.controller;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
@RequestMapping("/hello")
public class HelloController {

```

- Η κλάση HelloController είναι annotated ως **@Controller** που σημαίνει ότι επιστρέφει web σελίδες
- Το **@RequestMapping** ορίζει το root path του Controller

Controller (2)

- Παρατηρούμε στην κλάση *HelloController* το annotation **@Controller** της κλάσης **που σηματοδοτεί ότι επιστρέφεται σελίδα** και όχι data καθώς και το annotation **@RequestMapping** που ορίζει το path του Controller (μπορεί να μην υπάρχει path, οπότε εννοείται το /), ενώ στη συνέχεια ορίζουμε την μέθοδο **getHello** με παραμέτρους το *path* και το *method* που απλά επιστρέφει τη σελίδα
- Οι σελίδες αναζητούνται by default στο φάκελο **templates** δηλαδή src/main/resources/templates Κάνοντας return "index" ο ViewResolver του Spring αναζητά την index.html

@GetMapping (1)


```java
package gr.aueb.cf.springstarter.controller;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
@RequestMapping("/hello")
public class HelloController {

    @GetMapping("/coding")
//    @RequestMapping(path = "/coding", method = RequestMethod.GET)
    public String sayHello(Model model) {
        model.addAttribute("message", "Hello Coding!");
        return "index";
    }
}
```

- Το **@GetMapping ορίζει το path** στο **/coding** που περνάει ως παράμετρος στο @GetMapping. Εναλλακτικά, θα μπορούσαμε να έχουμε **@RequestMapping** (με path και method)
- Επιστρέφουμε την index.html με **return "index"** όπου το Spring περιμένει να βρει μία σελίδα index.html στο φάκελο templates (convention over configuration)

@GetMapping (2)


```
9   @Controller
10  @RequestMapping("/hello")
11  public class HelloController {
12
13      @GetMapping("/coding")
14  @  public String sayHello(Model model) {
15          model.addAttribute("message", "Hello Coding!");
16          return "index";
17      }
18  }
```

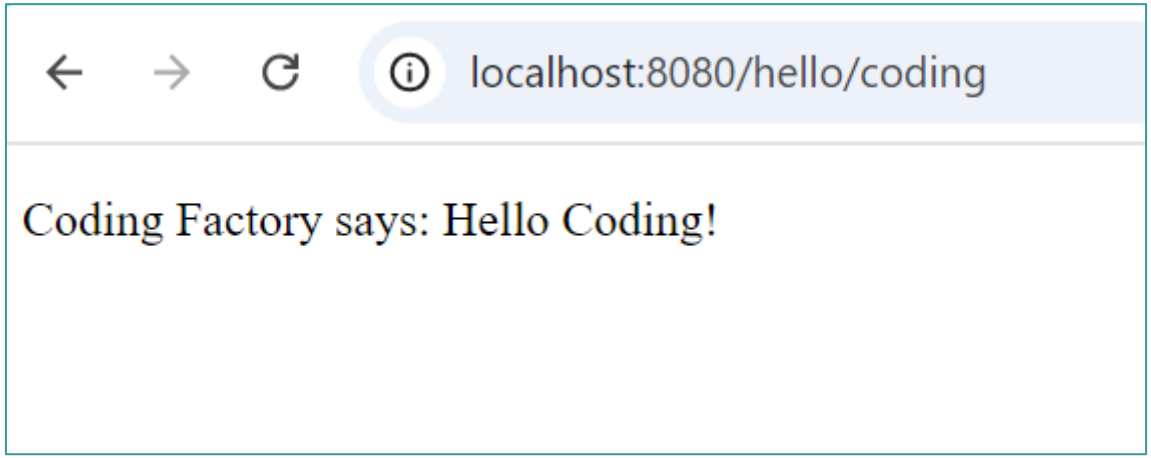
- Το Model είναι ένα interface του Spring που λειτουργεί ως Wrapper από model attributes. Περνάει ως παράμετρος στη μέθοδο που το χρησιμοποιεί. Ο Controller θέτει με **model.addAttribute()** τα presentational data
- Είναι προσβάσιμο στο view μέσω του SpEL - Spring Expression Language με το ${}

Index.html / thymeleaf


```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <title>Title</title>
6   </head>
7   <body>
8       <p>Coding Factory says: <span th:text="${message}"></span></p>
9   </body>
10  </html>
```

- Δημιουργούμε το View με Thymeleaf (**th:**). Στο HTML Template έχουμε εισάγει το ${cf}

Αποτέλεσμα



- Πρόσβαση στη σελίδα μπορούμε να έχουμε απευθείας με μέσω του Controller **localhost:8080/hello/coding**

Thymeleaf (1)


```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Title</title>
6  </head>
7  <body>
8      <p th:text="${dateStr}"></p>
9  </body>
10 </html>
```

- Δημιουργούμε νέα σελίδα date.html. Το Thymeleaf namespace στο <html> δεν είναι απαραίτητο
- Έχουμε εισάγει το ${dateStr}

Thymeleaf (2)


```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Title</title>
6  </head>
7  <body>
8      <p th:text="${dateStr}"></p>
9  </body>
10 </html>
```

Στο <h1> έχουμε την ιδιότητα **th:text** που αντιστοιχεί σε περιεχόμενο, και την τιμή "${dateStr}" που είναι Thymeleaf/Spring Expression Language και αντιστοιχεί σε πεδία του interface *model* που θα ορίσουμε στον **Controller** (βλ. επόμενη διαφάνεια). Επομένως με αυτόν τον τρόπο μέσω του interface **Model** επιτυγχάνουμε μεταφορά δεδομένων από τον Controller στο View

Interface Model


```
23      @GetMapping("/date")
24  @   public String getDate(Model model) {
25          LocalDateTime now = LocalDateTime.now();
26          DateTimeFormatter dateTimeFormatter = DateTimeFormatter.ofPattern("EEEE, dd MMMM, yyyy HH:mm");
27          String formattedDate = now.format(dateTimeFormatter);
28          model.addAttribute("dateStr", formattedDate);
29          return "date";
30      }
31  }
```

- Το Model είναι ένα interface του Spring που λειτουργεί ως Wrapper από model attributes. Περνάει ως παράμετρος στη μέθοδο που το χρησιμοποιεί
- Ο Controller θέτει με **model.addAttribute()** τα presentational data
- Είναι προσβάσιμο στο view μέσω του SpEL (Spring Expression Language ${})
- Το EEEE στο pattern είναι η ημέρα, το dd η ημερομηνία, το MMMM ο μήνας ολογράφως.

@RequestParam

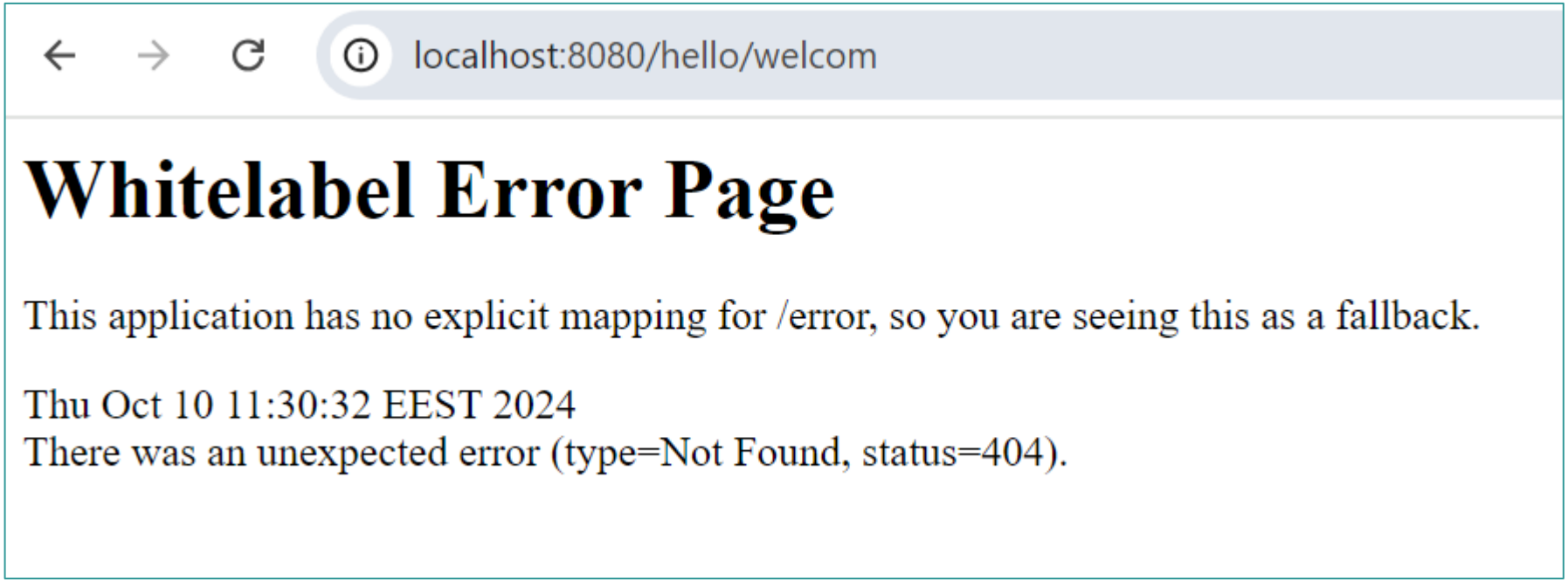

```
33      @GetMapping("/welcome")
34  @   public String sayHello(@RequestParam(value = "name", defaultValue = "Quest") String name,
35                             Model model) {
36          model.addAttribute("name", name);
37          return "welcome";
38      }
39  }
```

- Με @RequestParam κάνουμε bind query strings
- Επίσης, ορίζουμε μία error.html page για error handling

message.html


```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <title>Title</title>
6   </head>
7   <body>
8       <p>Hello <span th:text></span></p>
9   </body>
10  </html>
```

Whitelabel error



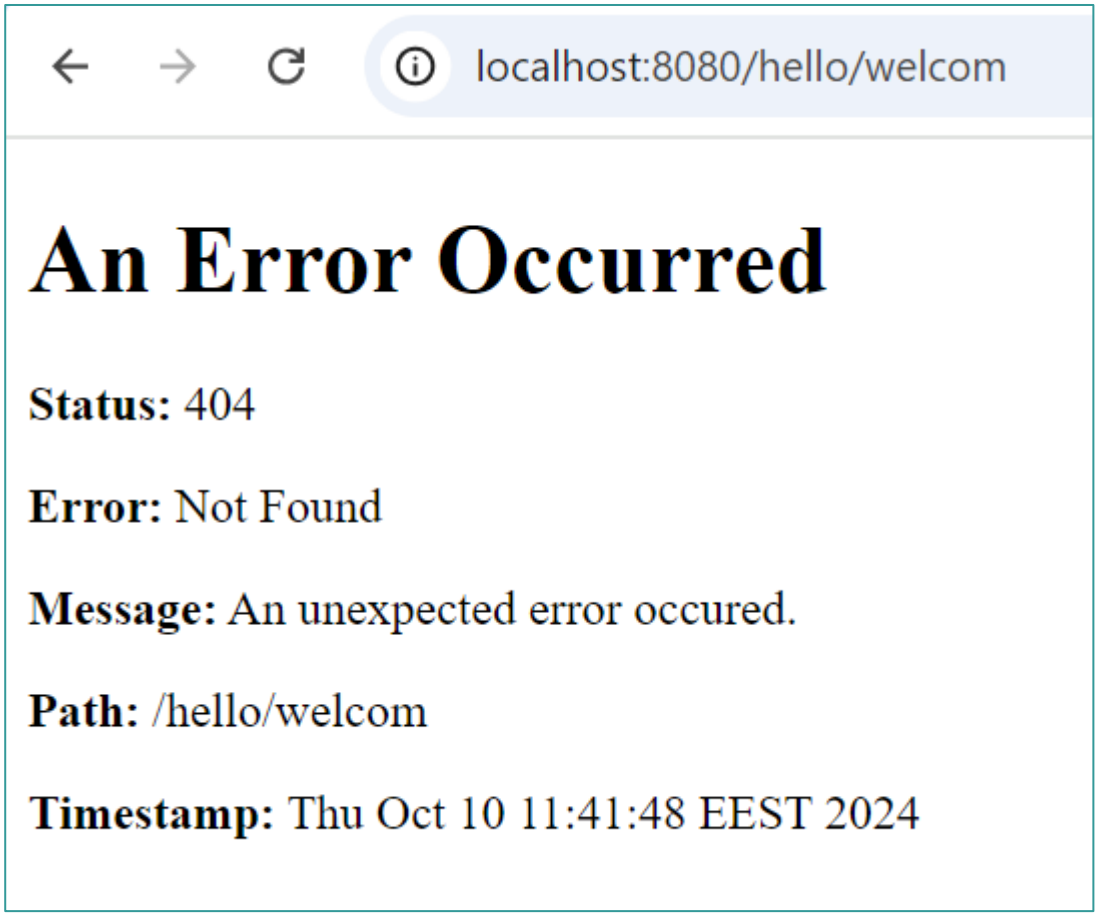
- Το Spring παρέχει ένα default /error endpoint που επιστρέφει μία error.html ή οποία να δεν υπάρχει επιστρέφεται ένα Whitelabel μήνυμα
- Θα ορίσουμε μία error.html

Error.html (1)


```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Title</title>
6  </head>
7  <body>
8      <h1>An Error Occurred</h1>
9      <p><strong>Status:</strong> <span th:text="${status}"></span></p>
10     <p><strong>Error:</strong> <span th:text="${error}"></span></p>
11     <p><strong>Message:</strong> <span th:text="${message} ?: 'An unexpected error occured.' "></span></p>
12     <p><strong>Path:</strong> <span th:text="${path}"></span></p>
13     <p><strong>Timestamp:</strong> <span th:text="${timestamp}"></span></p>
14 </body>
15 </html>
```

- Τα status, error, message, path, timestamp παρέχονται ως attributes από το Spring και το /error endpoint
- Το ?: είναι ο Elvis operator στο Thymeleaf ή αλλιώς σε άλλες γλώσσες είναι ο null-coalescing operator που παρέχει μία default τιμή αν το expression είναι null or evaluates σε τιμή false

Error.html (2)



- Το αποτέλεσμα είναι το αναμενόμενο

Messages.properties UTF-8

Settings

Editor › File Encodings

Reset

Global Encoding: UTF-8

Project Encoding: <System Default: windows-1253>

Path | Encoding

Encodings are not configured

To change encoding IntelliJ IDEA uses for a file, a directory, or the entire project, add its path if necessary and then select encoding from the encoding list. Built-in file encoding (e.g. JSP, HTML or XML) overrides encoding you specify here. If not specified, files and directories inherit encoding settings from the parent directory or from the Project Encoding.

Default encoding for properties files: UTF-8

Transparent native-to-ascii conversion

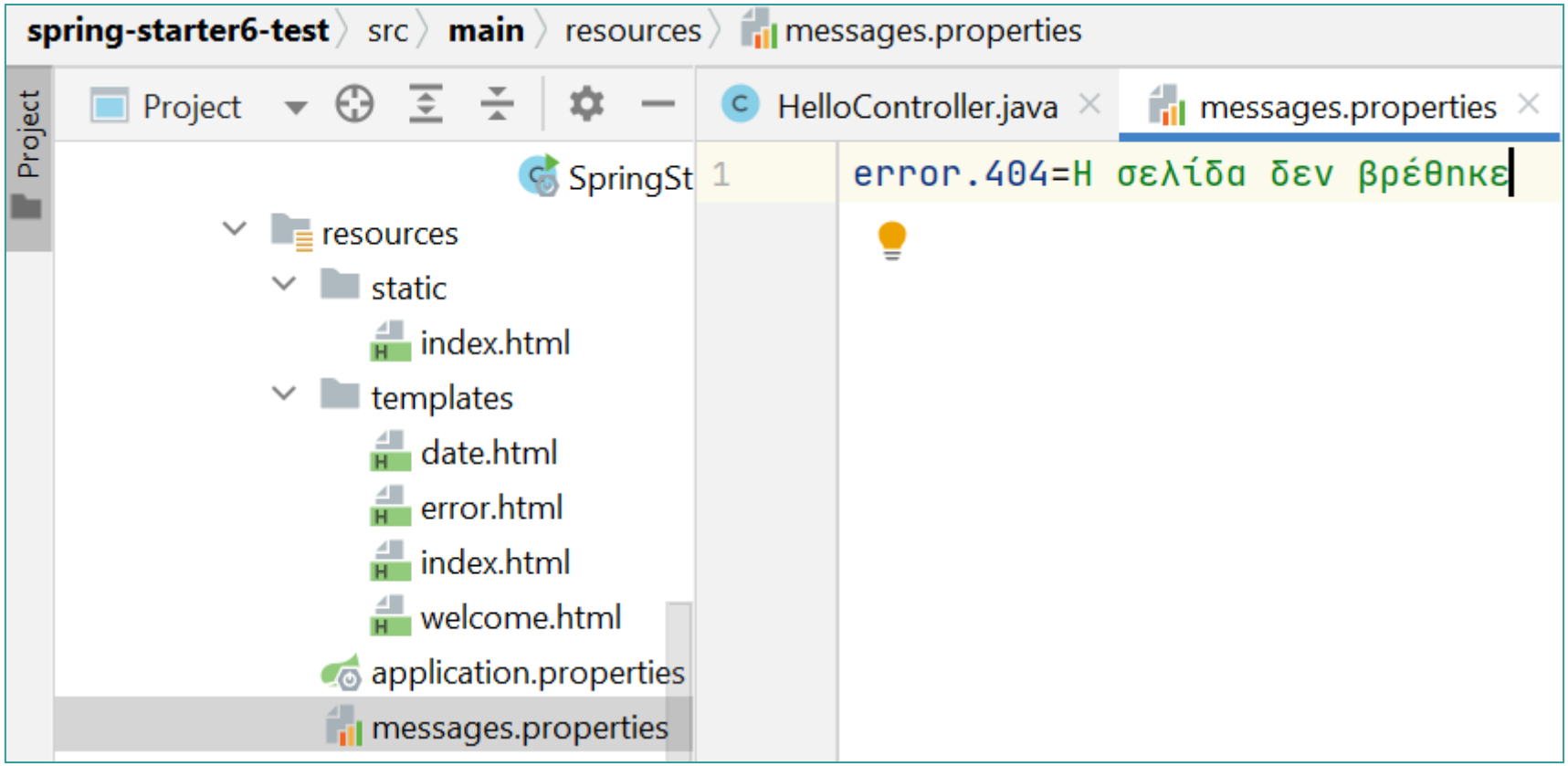
Create UTF-8 files: with NO BOM

IDEA will NOT add UTF-8 BOM to every created file in UTF-8 encoding

Appearance & Behavior
Data Editor and Viewer
Quick Lists
Path Variables
Presentation Assistant
Keymap
Editor
General
Code Editing
Font
Color Scheme
Code Style
Inspections
File and Code Templates
File Encodings
Live Templates
File Types
Copyright
Inlay Hints
Duplicates
Emmet
GUI Designer
Intentions
Language Injections

OK Cancel Apply

Messages.properties



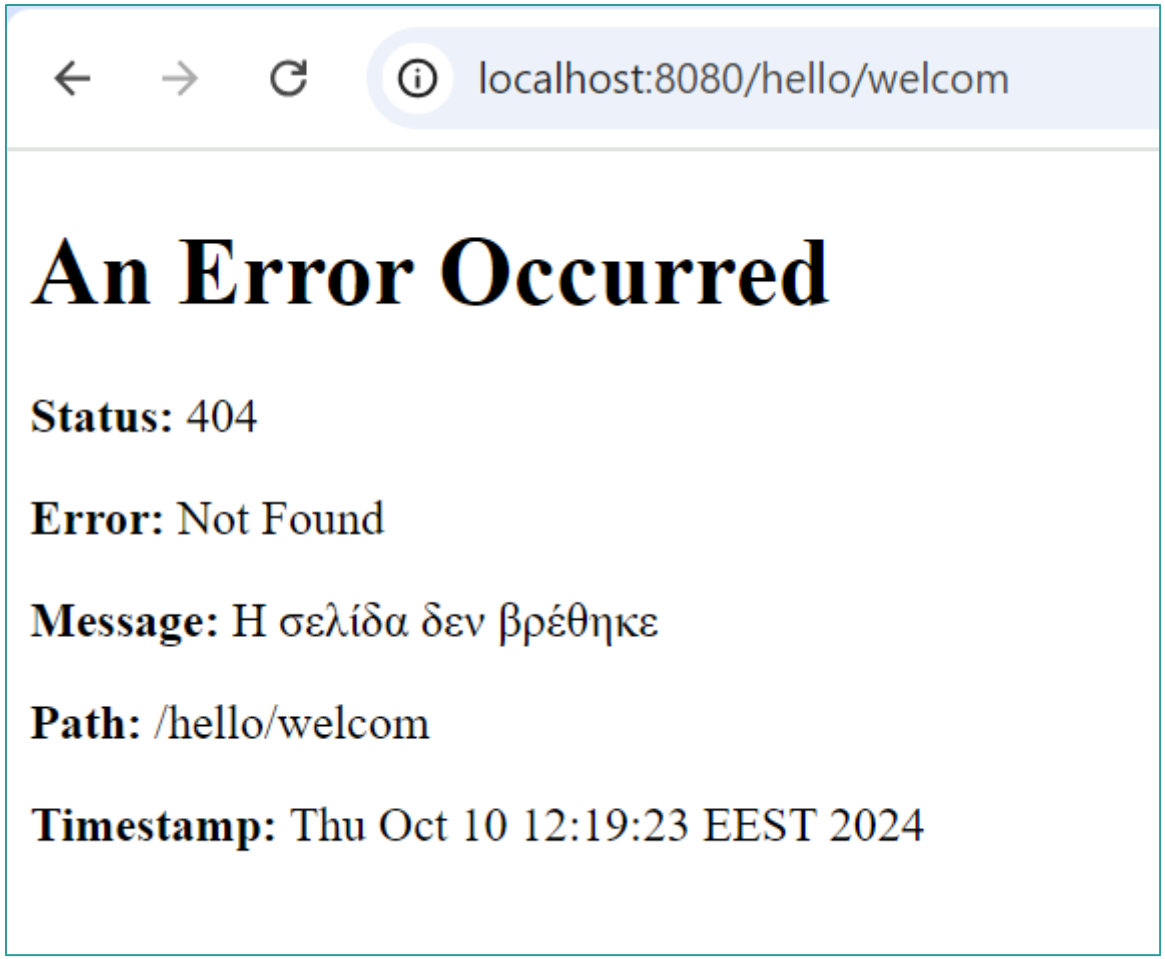
- Στο αρχείο messages.properties μπορούμε ένα έχουμε ένα κεντρικό σημείο των error μηνυμάτων σε μορφή key value

Error.html


```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <h1>An Error Occurred</h1>
    <p><strong>Status:</strong> <span th:text="${status}"></span></p>
    <p><strong>Error:</strong> <span th:text="${error}"></span></p>
    <p><strong>Message:</strong> <span th:text="${message} ?: #{error.404} "></span></p>
    <p><strong>Path:</strong> <span th:text="${path}"></span></p>
    <p><strong>Timestamp:</strong> <span th:text="${timestamp}"></span></p>
</body>
</html>
```

- Με #{error.html} έχουμε πρόσβαση στα messages.properties

Αποτέλεσμα



- Το αποτέλεσμα είναι το αναμενόμενο

Teacher insert DTO


```java
package gr.aueb.cf.springstarter.dto;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@NoArgsConstructor
@AllArgsConstructor
@Getter
@Setter
public class TeacherInsertDTO {
    private String firstname;
    private String lastname;
}

```

- Ορίζουμε ένα dto στο οποίο θα κάνουμε bind δεδομένα από φόρμα

TeacherReadOnlyDTO


```
14  public class TeacherRestController {
15
16      @GetMapping("/teachers/{id}")
17      public ResponseEntity<TeacherReadOnlyDTO> getOneTeacher(@PathVariable() Long id) {
18          TeacherReadOnlyDTO readOnlyDTO = new TeacherReadOnlyDTO(1L, "Αθανάσιος", "Ανδρούτσος");
19
20          if (Objects.equals(id, 1L)) {
21              return new ResponseEntity<>(readOnlyDTO, HttpStatus.OK);
22          } else {
23              return new ResponseEntity<>(HttpStatus.NOT_FOUND);
24          }
25      }
```

- Δημιουργούμε ένα TeacherReadOnlyDTO

Insert student


```java
@Controller
@RequestMapping("/teachers")
public class TeacherController {

    @GetMapping("/insert")
    public String getTeacherForm(Model model) {
        model.addAttribute("teacherInsertDTO", new TeacherInsertDTO());
        return "teachers/insert";
    }

    @PostMapping("/insert")
    public String insertTeacher(@ModelAttribute("teacherInsertDTO") TeacherInsertDTO teacherInsertDTO,
                                Model model) {
        // insert teacher
        // TeacherReadOnlyDTO readOnlyDTO = Mapper..
        TeacherReadOnlyDTO readOnlyDTO = new TeacherReadOnlyDTO(1L, "Athanassios", "Androutsos");
        model.addAttribute("dto", readOnlyDTO);
        return "teachers/success";
    }
}
```

- Κάνουμε get τη φόρμα (teachers/insert) περνώντας ένα instance του TeacherInsertDTO ως teacherInsertDto. Με POST παίρνουμε πίσω το teacherInsertDto και το κάνουμε bind με @ModelAttribute. Στη συνέχεια καλούμε τη success.html

Insert.html


```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Εγγραφή Καθηγητή</title>
</head>
<body>
    <div th:object="${teacherInsertDTO}">

        <form action="" method="POST">

            <div>
                <span>Εγγραφή Καθηγητή</span>
            </div>
            <div>
                <input type="text" name="firstname" placeholder="Όνομα" th:field="*{firstname}">
            </div>
            <div>
                <input type="text" name="lastname" placeholder="Επώνυμο" th:field="*{lastname}">
            </div>
              <div>
                  <input type="submit" value="Εισαγωγή">
              </div>
          </form>
      </div>
</body>
</html>
```

- Το ${teacherInsertDto} το εισάγουμε στο html με th:object. Στα πεδία του dto κάνουμε bind με th:field και *{ }
- Το th:field λειτουργεί όπως το value. Κάνει bind αλλά και εμφανίζει τις τιμές των αντίστοιχων πεδίων του teachrInsertDto

Success.html


```
1   <!DOCTYPE html>
2   <html xmlns:th="http://www.thymeleaf.org" lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <title>Title</title>
6   </head>
7   <body>
8       <p>Student Inserted</p>
9       <p th:text="${dto.firstname}"></p>
10      <p th:text="${dto.lastname}"></p>
11  </body>
12  </html>
```

Data Binding

- Με το **Thymeleaf και το Spring μπορούμε να κάνουμε data-binding τα input πεδία της φόρμας με DTO objects στην πλευρά του Server** με το Annotation **@ModelAttribute**
- Ουσιαστικά **η σύνδεση γίνεται μέσω ενός κοινού αντικειμένου** που κάνουμε αρχικά add στο Model με model.addAttribute("teacherInsertDto", …) και στη συνέχεια κάνουμε reference στην HTML φόρμα (με th:object = "${teacherInsertDto}")

Data Binding Thymeleaf

- **HTML Φόρμα:** Το Thymeleaf μας παρέχει ειδικές ιδιότητες (HTML attributes) για data binding και form handling:
  - **th:object** κάνει reference το backing bean, τυπικά ένα DTO object που περιέχει πεδία ίδια με τα πεδία της φόρμας. Μέσα σε μια φόρμα μπορεί να υπάρχει ένα μόνο **th:object**
  - **th:field** που κάνει bind form input fields με τα πεδία του form-backing bean

REST Controller (1)


```java
package gr.aueb.cf.springstarter.rest;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/api")
public class HelloRestController {

    @GetMapping("/hello")
    public String sayHello() {
        return "Hello World!";
    }
}
```

- Ο HelloRestController είναι @**RestController**. Δεν επιστρέφει σελίδα αλλά Data. Στο παράδειγμα η sayHello() επιστρέφει απλά *String*. Αν όμως επιστρέφαμε **Object ή Collection** θα επιστρεφόταν αυτόματα JSON

REST Controller (2)


```java
package gr.aueb.cf.springstarter.rest;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/api")
public class HelloRestController {

    @GetMapping("/hello")
    public String sayHello() {
        return "Hello World!";
    }
}
```

- Η μέθοδος say*Hello()* κάνει **@GetMapping ("/hello")** και απαντάει σε **GET** Requests.

Postman

http://localhost:8080/api/hello

Save

GET http://localhost:8080/api/hello Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies Headers (5) Test Results 200 OK 120 ms 174 B Save as Example

Pretty Raw Preview Visualize Text

```
1  Hello Rest
```

- Με Postman ελέγχουμε

Student REST Controller


```java
public class TeacherRestController {

    @GetMapping("/teachers/{id}")
    public ResponseEntity<TeacherReadOnlyDTO> getOneTeacher(@PathVariable() Long id) {
        TeacherReadOnlyDTO readOnlyDTO = new TeacherReadOnlyDTO(1L, "Αθανάσιος", "Ανδρούτσος");

        if (Objects.equals(id, 1L)) {
            return new ResponseEntity<>(readOnlyDTO, HttpStatus.OK);
        } else {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
    }
```

- Το ResponseEntity<> είναι παρόμοιο με το Response της JEE
- Ορίζεται με Generics και παρέχεται σε συντακτικά σύντομη μορφή με μία ή δυο παραμέτρους΄, όπου η 1η μπορεί να είναι το dto αν υπάρχει και η 2η το HTTP status code ή απλά με μία παράμετρο το HTTP status code

Path variable


```
14  @RestController
15  @RequestMapping("/api")
16  public class TeacherRestController {
17
18      @GetMapping("/teachers/{id}")
19      public ResponseEntity<TeacherReadOnlyDTO> getOneTeacher(@PathVariable("id") Long id) {
```

- Στις παραμέτρους της μεθόδου getOneStudent έχουμε μία παράμετρο που είναι annotated ως **@PathVariable** με το όνομα **id** (όπως και το όνομα της path variable στο URI) ενώ επίσης και το όνομα της παραμέτρου (Long id) είναι το ίδιο αν και αυτό δεν είναι απαραίτητο
- Έτσι κάνουμε binding μεταξύ της path variable και της τυπικής παραμέτρου. Εφόσον μάλιστα το όνομα του path variable είναι ίδιο με το όνομα της παραμέτρου όπως εδώ, το id στο Path Variable θα μπορούσε να παραλειφθεί

ResponseEntity (1)


```
16      @GetMapping("/teachers/{id}")
17      public ResponseEntity<TeacherReadOnlyDTO> getOneTeacher(@PathVariable() Long id) {
18          TeacherReadOnlyDTO readOnlyDTO = new TeacherReadOnlyDTO(1L, "Αθανάσιος", "Ανδρούτσος");
19
20          if (Objects.equals(id, 1L)) {
21              return new ResponseEntity<>(readOnlyDTO, HttpStatus.OK);
22          } else {
23              return new ResponseEntity<>(HttpStatus.NOT_FOUND);
24          }
25      }
```

- Επιστρέφουμε ένα instance της κλάσης **ResponseEntity<T>.** Η 1η παράμετρος στο return new ResponseEntity<> –αν υπάρχει- είναι το entity όπως το dto και η 2η παράμετρος είναι το HttpStatus ως **HttpStatus.OK.** Μπορεί να υπάρχει και μία μόνο παράμετρος, όπως παραπάνω το HttpStatus ως **NOT_FOUND**.

Postman

http://localhost:8080/api/teachers/1

Save | Share

GET http://localhost:8080/api/teachers/1 Send

Params | Authorization | Headers (7) | Body | Scripts | Tests | Settings | Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body | Cookies | Headers (5) | Test Results — 200 OK • 382 ms • 232 B

Pretty | Raw | Preview | Visualize | JSON

```
1  {
2      "firstname": "Αθανάσιος",
3      "lastname": "Ανδρούτσος"
4  }
```

- Με Postman ελέγχουμε το endpoint

Add Student - JSON Binding


```
27      @PostMapping("")
28      public ResponseEntity<TeacherReadOnlyDTO> addTeacher(@RequestBody TeacherInsertDTO teacherInsertDTO) {
29          // Add the teacher and get back a TeacherReadOnlyDTO from Service Layer
30          TeacherReadOnlyDTO readOnlyDTO = new TeacherReadOnlyDTO(1L, "Αθανάσιος", "Ανδρούτσος");
31          return new ResponseEntity<>(new TeacherReadOnlyDTO(), HttpStatus.CREATED);
32      }
33  }
```

- Με **@RequestBody** αντιστοιχούμε row data (JSON) σε DTO
- Επιστρέφουμε Created

Postman

POST http://localhost:8080/api/teachers Send

Params Authorization Headers (9) Body Scripts Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
{
    "firstname": "Athanassios",
    "lastname": "Androutsos"
}
```

Body Cookies Headers (5) Test Results 201 Created 193 ms 244 B

Pretty Raw Preview Visualize JSON

```
{
    "id": 1,
    "firstname": "Αθανάσιος",
    "lastname": "Ανδρούτσος"
}
```

- Έλεγχος με Postman