Università di Catania

Uni ct MATEMATICA E INFORMATICA

REST Service

Alessandro Midolo, PhD

Dipartimento di Matematica e Informatica

Università di Catania

alessandro.midolo@unict.it

https://www.dmi.unict.it/amidolo/

A.A. 2024/2025

Obiettivo

- Implementare un servizio che accetta richieste HTTP GET all'url http://localhost:8080/greeting
- Il servizio risponde con un file JSON che rappresenta un semplice saluto
- E' possibile personalizzare il saluto con l'inserito di un parametro opzionale *name* nella stringa di query
- Il file Json avrà questa struttura:

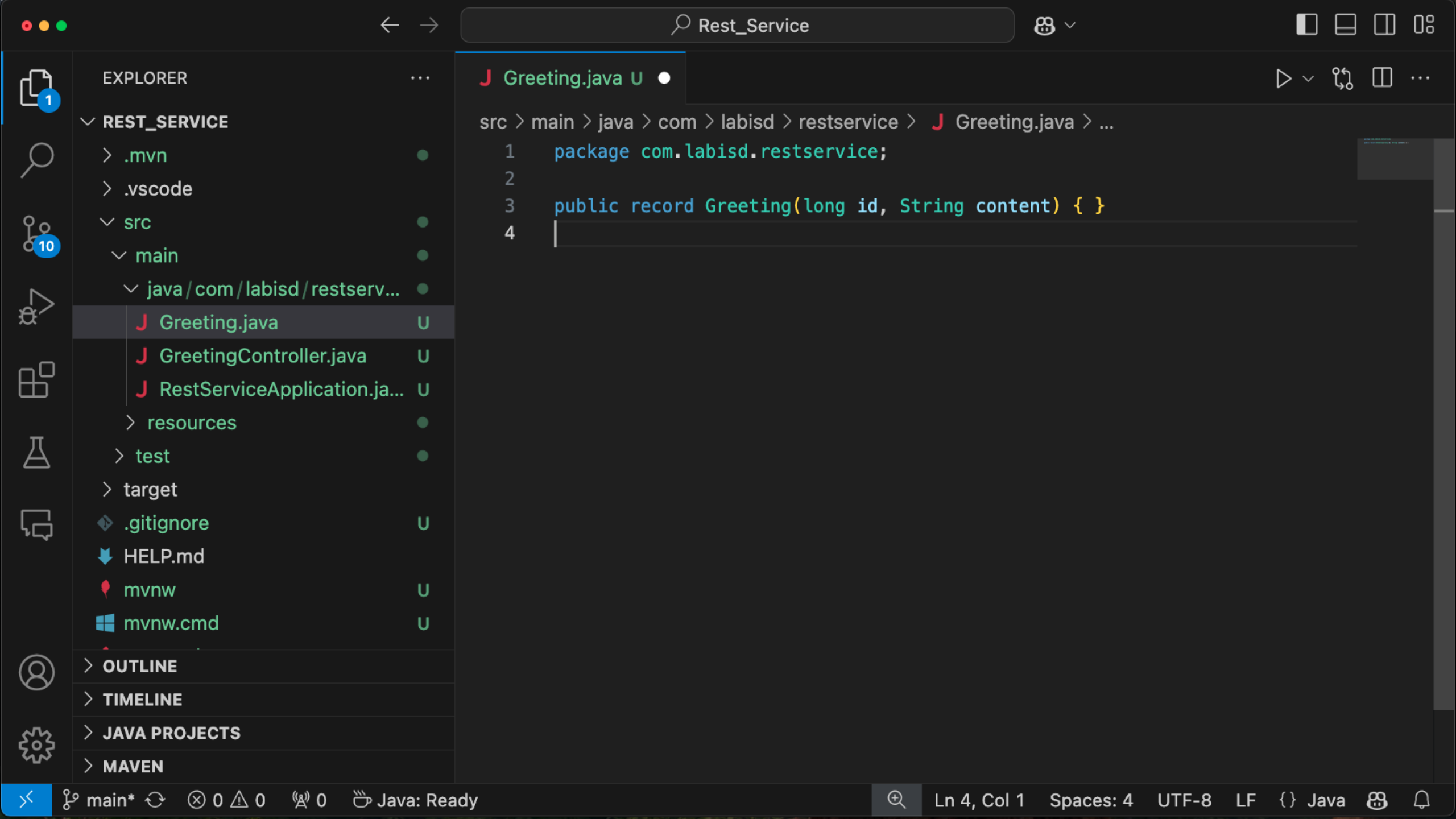
```
{"id":1,"content":"Hello, User!"}
```

Inizializziamo il Progetto

- Navigare all'url https://start.spring.io per inizializzare il progetto
- Scegliere Maven e java come linguaggio
- Selezionare la dipendenza *Spring Web*
- Generare il progetto
- Scaricare il file zip ed estrarlo all'interno del proprio IDE

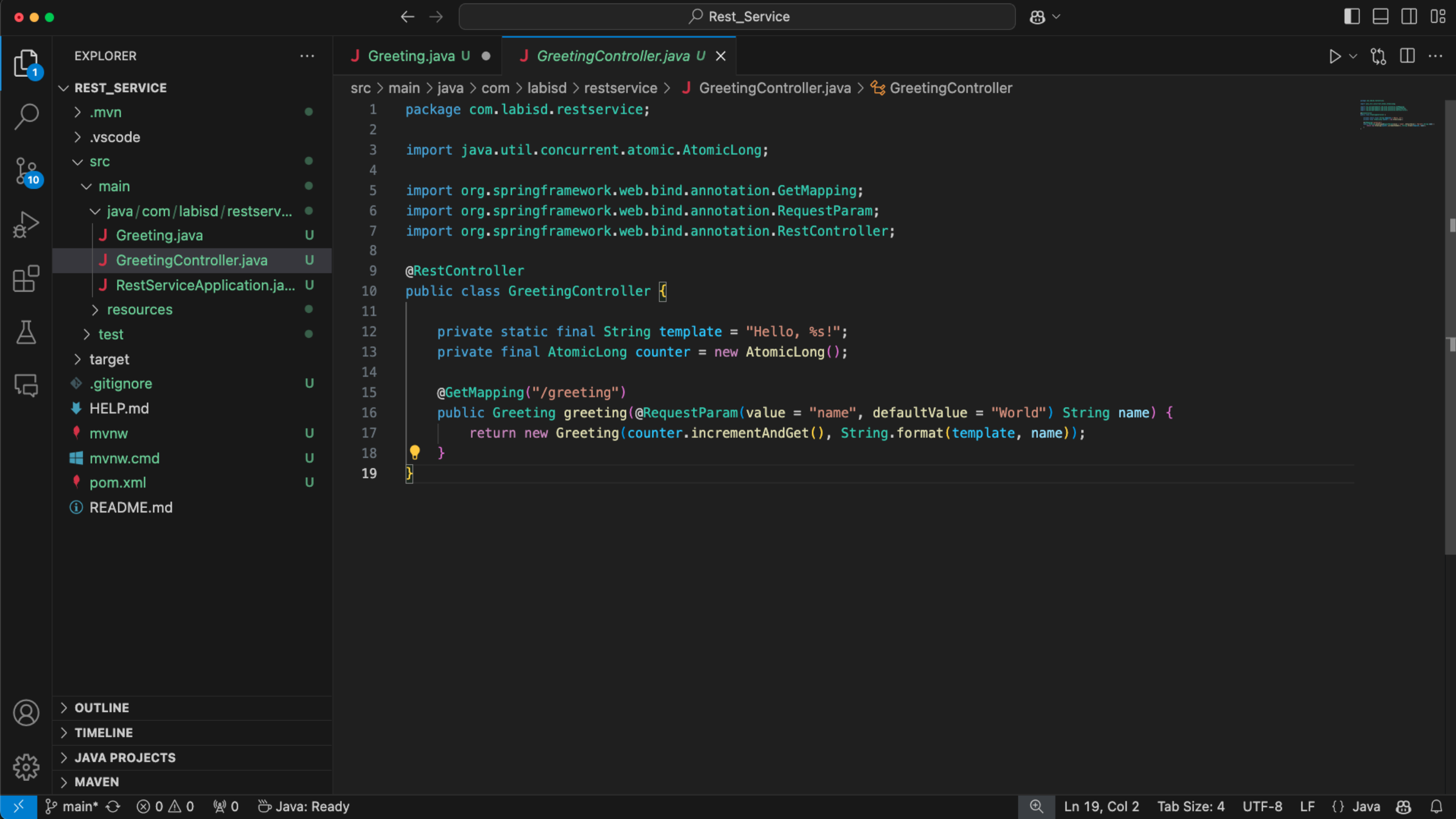
Resource Representation Class

- Il servizio gestirà richieste GET per la pagina */greeting*, con un parametro opzionale *name* nella stringa di query
- La richiesta GET dovrebbe ritornare una risposta 200 OK con un json nel corpo della risposta, la quale rappresentarà il saluto
- La risposta avrà due campi:
  - Il campo *id* è un identificatore unico per il saluto
  - Il campo *content* è la rappresentazione testuale del saluto
- Creiamo quindi una *resource representation class* che rappresenterà i dati in questione


Rest_Service
EXPLORER
REST_SERVICE
.mvn
.vscode
src
main
java/com/labisd/restserv...
Greeting.java U
GreetingController.java U
RestServiceApplication.ja... U
resources
test
target
.gitignore U
HELP.md
mvnw U
mvnw.cmd U
OUTLINE
TIMELINE
JAVA PROJECTS
MAVEN
Greeting.java U
src > main > java > com > labisd > restservice > Greeting.java > ...
1 package com.labisd.restservice;
2
3 public record Greeting(long id, String content) { }
4
main* 0 0 0 Java: Ready
Ln 4, Col 1 Spaces: 4 UTF-8 LF Java

Resource Controller

- Le richieste HTTP sono gestite da un controller
- Queste componenti sono identificate dall'annotazione ***@RestController***
- Andiamo quindi a creare un nostro controller personalizzato per la gestione delle chiamate GET alla pagina */greeting* ritornando una nuova istanza della classe *Greeting* che abbiamo appena creato


Rest_Service
EXPLORER
REST_SERVICE
.mvn
.vscode
src
main
java/com/labisd/restserv...
Greeting.java U
GreetingController.java U
RestServiceApplication.ja... U
resources
test
target
.gitignore U
HELP.md
mvnw U
mvnw.cmd U
pom.xml U
README.md
Greeting.java U
GreetingController.java U
src > main > java > com > labisd > restservice > GreetingController.java > GreetingController
1 package com.labisd.restservice;
2
3 import java.util.concurrent.atomic.AtomicLong;
4
5 import org.springframework.web.bind.annotation.GetMapping;
6 import org.springframework.web.bind.annotation.RequestParam;
7 import org.springframework.web.bind.annotation.RestController;
8
9 @RestController
10 public class GreetingController {
11
12 private static final String template = "Hello, %s!";
13 private final AtomicLong counter = new AtomicLong();
14
15 @GetMapping("/greeting")
16 public Greeting greeting(@RequestParam(value = "name", defaultValue = "World") String name) {
17 return new Greeting(counter.incrementAndGet(), String.format(template, name));
18 }
19 }
OUTLINE
TIMELINE
JAVA PROJECTS
MAVEN
main*
0 0
0
Java: Ready
Ln 19, Col 2
Tab Size: 4
UTF-8
LF
Java

- ***@GetMapping*** assicura che le chiamate HTTP GET ricevute alla pagine */greeting* vengano mappate all metodo *greeting()* (per le chiamate POST si usa ***@PostMapping***)
- ***@RequestParam*** associa il valore del parametro *name* nella stringa della query al parametro *name* del metodo *greeting()*. Se il valore è assente nella query, verrà usato il valore di default
- L'implementazione del corpo del metodo crea e ritorna un oggetto di tipo *Greetings* con gli attributi *id* e *content* basati sul successivo valore di counter e formatta la stringa *name* usando il template di greeting
- Una differenza fondamentale tra un controller MVC tradizionale e il controller del servizio web RESTful mostrato in precedenza è il modo in cui viene creato il corpo della risposta HTTP
  - Invece di affidarsi a una tecnologia di visualizzazione per eseguire il rendering lato server dei dati di greeing in HTML, questo controller del servizio web RESTful popola e restituisce un oggetto di *Greeting*. I dati dell'oggetto verranno scritti direttamente nella risposta HTTP come JSON
- L'oggetto *Greeting* deve essere convertito in JSON. Grazie al supporto del convertitore di messaggi HTTP di Spring, non è necessario effettuare questa conversione manualmente. Poiché Jackson 2 è sul classpath, *MappingJackson2HttpMessageConverter* di Spring viene automaticamente scelto per convertire l'istanza di *Greeting* in JSON

Eseguiamo il Progetto

- Si può eseguire l'applicazione dalla riga di comando con Gradle o Maven. Si può anche creare un singolo file JAR eseguibile che contiene tutte le dipendenze, le classi e le risorse necessarie ed eseguirlo
- Creare un file JAR eseguibile semplifica la distribuzione, la versione e la distribuzione del servizio come applicazione durante tutto il ciclo di vita dello sviluppo, in diversi ambienti etc.
- Gradle
  - **./gradlew bootRun** per eseguire l'applicazione
  - **./gradlew build** per generare il file Jar
- Maven
  - **./mvnw spring-boot:run** per eseguire l'applicazione
  - **./mvnw clean package** per generare il file jar

Università di Catania

Uni ct MATEMATICA E INFORMATICA

The End 

Alessandro Midolo, PhD

Dipartimento di Matematica e Informatica

Università di Catania

alessandro.midolo@unict.it

https://www.dmi.unict.it/amidolo/

A.A. 2024/2025