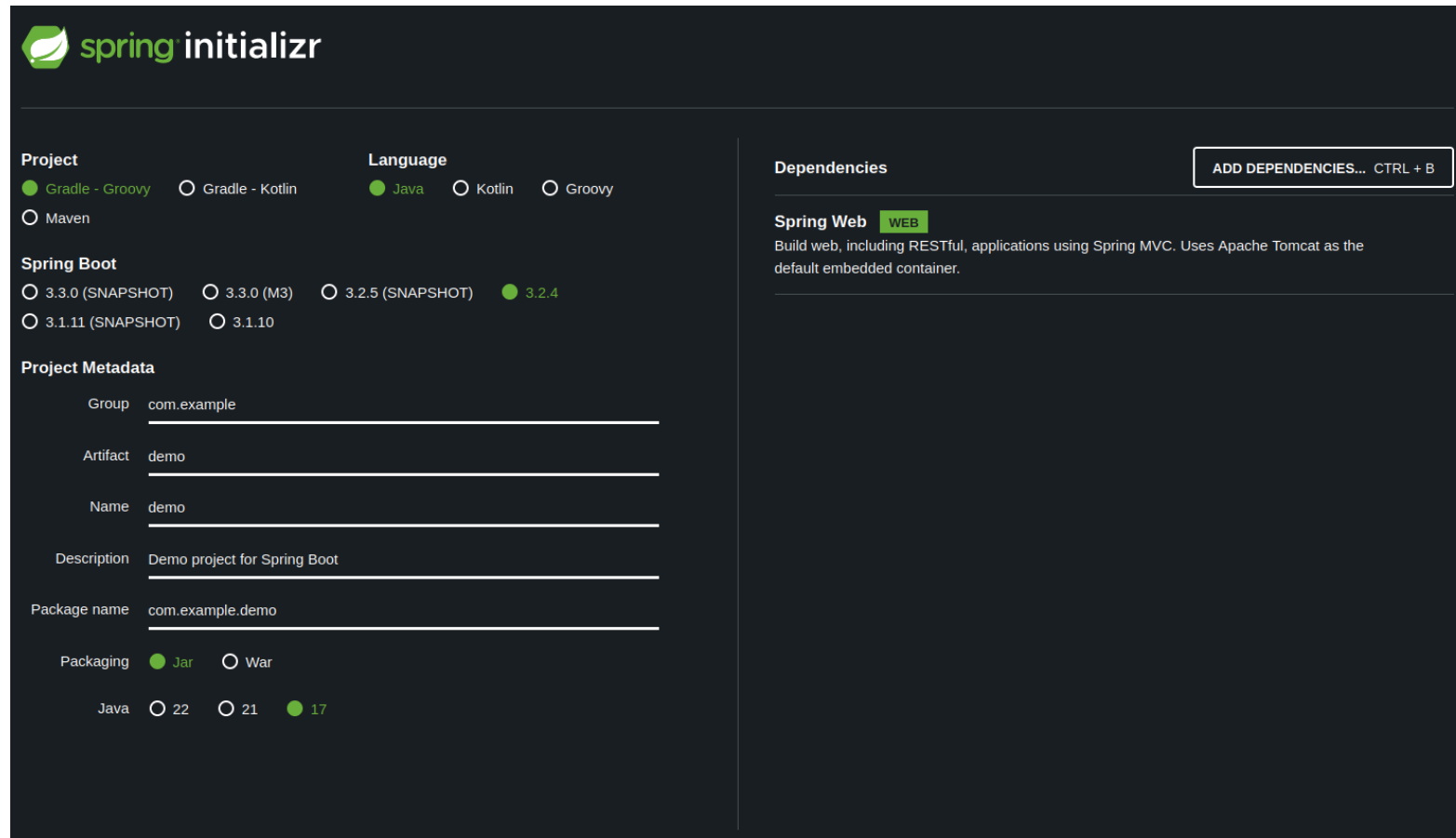


# REST API con Java Spring

# Java Spring

Per creare un REST API con Java Spring, è necessario utilizzare **spring initializr** [start.spring.io](https://start.spring.io).



The screenshot shows the Spring Initializr web application interface. The header features the Spring logo and the text "spring initializr". The main content area is divided into several sections:

- Project**: Includes radio buttons for "Gradle - Groovy" (selected), "Gradle - Kotlin", "Maven", "Kotlin", and "Groovy".
- Language**: Includes radio buttons for "Java" (selected), "Kotlin", and "Groovy".
- Spring Boot**: Includes radio buttons for "3.3.0 (SNAPSHOT)", "3.3.0 (M3)", "3.2.5 (SNAPSHOT)", "3.2.4" (selected), "3.1.11 (SNAPSHOT)", and "3.1.10".
- Project Metadata**: Includes input fields for "Group" (com.example), "Artifact" (demo), "Name" (demo), "Description" (Demo project for Spring Boot), and "Package name" (com.example.demo).
- Packaging**: Includes radio buttons for "Jar" (selected) and "War".
- Java**: Includes radio buttons for "22", "21", and "17" (selected).
- Dependencies**: Includes a button "ADD DEPENDENCIES... CTRL + B" and a section for "Spring Web" (selected) with a description: "Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container."

Si può scegliere il linguaggio, la versione di Spring Boot, le dipendenze e il progetto. Nel nostro caso possiamo lasciare tutto default e aggiungere solo la dipendenza **Spring Web**.

Avremo così un progetto con una struttura base.

# Struttura del progetto

```
├── src
│   ├── main
│   │   ├── java
│   │   │   ├── com
│   │   │   │   ├── example
│   │   │   │   │   ├── demo
│   │   │   │   │   │   ├── DemoApplication.java
│   │   │   │   │   │   ├── Greeting.java
│   │   │   │   │   │   └── HelloController.java
```

# Prima API

Per creare una API, è necessario creare una classe con un metodo che restituisca un oggetto.

```
@RestController
public class HelloController {

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

In questo caso, la classe `HelloController` è un **RestController** e il metodo `hello` restituisce una stringa sulla "route" `/hello`.

# Parametri

Per passare parametri, è possibile utilizzare l'annotazione `@RequestParam` .

```
@GetMapping("/hello")  
public String hello(@RequestParam(value = "name", defaultValue = "World") String name) {  
    return "Hello " + name + "!";  
}
```

In questo caso, il parametro `name` è opzionale e ha un valore di default `World` .

# Record

In java è possibile utilizzare i **record** per creare classi con pochi campi.

```
public record Greeting(long id, String content) {}
```

In questo caso, il record `Greeting` ha due campi: `id` e `content`.

```
private int counter = 0;

@GetMapping("/hello")
public Greeting hello(@RequestParam(value = "name", defaultValue = "World") String name) {
    return new Greeting(counter++, "Hello " + name + "!");
}
```

Il record può essere restituito come risposta, e verrà automaticamente convertito in JSON.