mvn spring-boot:run ng serve --open

1. Setup Spring Boot Backend (Maven)

Step 1: Create a Spring Boot Project

You can create a Spring Boot project using **Spring Initializr**:

- Visit Spring Initializr
- Select:
  - **Project**: Maven
  - **Language**: Java
  - **Spring Boot Version**: Latest stable version
  - **Dependencies**:
    - **Spring Web** (for REST API)
    - **Spring Boot DevTools** (for development ease)
    - **Spring Data JPA** (for database interaction)
    - **H2 Database / MySQL** (for persistence)
    - **Lombok** (to reduce boilerplate code)
    - **Spring Security** (optional, for authentication)
- Click **Generate**, extract the zip, and open it in **IntelliJ IDEA / VS Code / Eclipse**.

Step 2: Configure `application.properties`

Modify the `src/main/resources/application.properties`:

For **H2 Database (In-memory)**:

```
properties
CopyEdit
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.h2.console.enabled=true
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
```

For **MySQL**:

```
properties
CopyEdit
spring.datasource.url=jdbc:mysql://localhost:3306/your_database
spring.datasource.username=root
spring.datasource.password=your_password
spring.jpa.hibernate.ddl-auto=update
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
```

---

mvn spring-boot:run ng serve --open

Step 3: Create Model Class

Create `User.java` inside `com.example.demo.model`:

```
java
CopyEdit
package com.example.demo.model;

import jakarta.persistence.*;
import lombok.*;

@Entity
@Getter @Setter @NoArgsConstructor @AllArgsConstructor
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private String email;
}
```

Step 4: Create Repository

Create `UserRepository.java` inside `com.example.demo.repository`:

```
java
CopyEdit
package com.example.demo.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import com.example.demo.model.User;

public interface UserRepository extends JpaRepository<User, Long> {
}
```

Step 5: Create Service Layer

Create `UserService.java` inside `com.example.demo.service`:

```
java
CopyEdit
package com.example.demo.service;

import com.example.demo.model.User;
import com.example.demo.repository.UserRepository;
import org.springframework.stereotype.Service;
import java.util.List;

@Service
public class UserService {
    private final UserRepository userRepository;
```

```
    public UserService(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    public List<User> getAllUsers() {
        return userRepository.findAll();
    }

    public User createUser(User user) {
        return userRepository.save(user);
    }
}
```

Step 6: Create REST Controller

Create `UserController.java` inside `com.example.demo.controller`:

```
java
CopyEdit
package com.example.demo.controller;

import com.example.demo.model.User;
import com.example.demo.service.UserService;
import org.springframework.web.bind.annotation.*;
import java.util.List;

@RestController
@RequestMapping("/users")
@CrossOrigin(origins = "http://localhost:4200") // Allow Angular to access
API
public class UserController {
    private final UserService userService;

    public UserController(UserService userService) {
        this.userService = userService;
    }

    @GetMapping
    public List<User> getUsers() {
        return userService.getAllUsers();
    }

    @PostMapping
    public User createUser(@RequestBody User user) {
        return userService.createUser(user);
    }
}
```

Step 7: Run Spring Boot Application

Run the application using:

mvn spring-boot:run ng serve --open

```
sh
CopyEdit
mvn spring-boot:run
```

Verify API in **Postman**:

- `GET http://localhost:8080/users`
- `POST http://localhost:8080/users` (with JSON body)

---

2. Setup Angular Frontend

Step 8: Create Angular Project

Run the following command:

```
sh
CopyEdit
ng new frontend
cd frontend
ng serve --open
```

Step 9: Install Bootstrap & HttpClient

Run:

```
sh
CopyEdit
npm install bootstrap --save
```

Add Bootstrap in `angular.json`:

```
json
CopyEdit
"styles": [
  "node_modules/bootstrap/dist/css/bootstrap.min.css",
  "src/styles.css"
]
```

Install **HttpClient** for API calls:

```
sh
CopyEdit
ng generate service services/user
```

Step 10: Setup `user.service.ts` for API Calls

Modify `src/app/services/user.service.ts`:

```
typescript
CopyEdit
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
```

```
})
export class UserService {
  private apiUrl = 'http://localhost:8080/users';

  constructor(private http: HttpClient) {}

  getUsers(): Observable<any[]> {
    return this.http.get<any[]>(this.apiUrl);
  }

  createUser(user: any): Observable<any> {
    return this.http.post<any>(this.apiUrl, user);
  }
}
```

Step 11: Create User Component

Run:

```
sh
CopyEdit
ng generate component components/user
```

Modify `src/app/components/user/user.component.ts`:

```
typescript
CopyEdit
import { Component, OnInit } from '@angular/core';
import { UserService } from 'src/app/services/user.service';

@Component({
  selector: 'app-user',
  templateUrl: './user.component.html',
  styleUrls: ['./user.component.css']
})
export class UserComponent implements OnInit {
  users: any[] = [];
  userName: string = '';
  userEmail: string = '';

  constructor(private userService: UserService) {}

  ngOnInit() {
    this.loadUsers();
  }

  loadUsers() {
    this.userService.getUsers().subscribe(data => {
      this.users = data;
    });
  }

  addUser() {
```

```
    const newUser = { name: this.userName, email: this.userEmail };
    this.userService.createUser(newUser).subscribe(() => {
      this.loadUsers();
      this.userName = '';
      this.userEmail = '';
    });
  }
}
```

Step 12: Create UI for Users

Modify `src/app/components/user/user.component.html`:

```
html
CopyEdit
<div class="container mt-4">
  <h2>User List</h2>
  <ul class="list-group">
    <li class="list-group-item" *ngFor="let user of users">
      {{ user.name }} - {{ user.email }}
    </li>
  </ul>

  <h3 class="mt-4">Add User</h3>
  <div>
    <input [(ngModel)]="userName" class="form-control my-2"
placeholder="Name" />
    <input [(ngModel)]="userEmail" class="form-control my-2"
placeholder="Email" />
    <button (click)="addUser()" class="btn btn-primary">Add User</button>
  </div>
</div>
```

Ensure **FormsModule** is imported in `app.module.ts`:

```
typescript
CopyEdit
import { FormsModule } from '@angular/forms';
@NgModule({
  imports: [FormsModule]
})
```

Step 13: Add Routing

Modify `src/app/app-routing.module.ts`:

```
typescript
CopyEdit
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { UserComponent } from './components/user/user.component';
```

```
const routes: Routes = [{ path: 'users', component: UserComponent }];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule {}
```

Step 14: Run Angular

Run:

```
sh
CopyEdit
ng serve
```

Visit `http://localhost:4200/users`

Conclusion

✅ Spring Boot serves data through REST API.
✅ Angular consumes the API and displays data in a user-friendly UI.
✅ You now have a **full-stack CRUD Angular + Spring Boot project!** 🚀