Step 5: API Development for CRUD Operations

In this step, we'll create the RESTful APIs that allow you to perform CRUD (Create, Read, Update, Delete) operations on high scores.

Steps for API Development

- **Create New Packages**
  - Create two new packages under **src/main/java/com/example/highscoreapi**: **controller** and **repository**.
- **Create Repository Interface**
  - In the **repository** package, create an interface named **HighScoreRepository**.
  - Extend it from **JpaRepository<HighScore, Long>**.

java

Copy code

```
package com.example.highscore.repository;

import com.example.highscore.model.HighScore;
import org.springframework.data.jpa.repository.JpaRepository;

public interface HighScoreRepository extends JpaRepository<HighScore, Long> {
}
```

- **Create Controller Class**
  - In the **controller** package, create a class named **HighScoreController**.
  - Annotate this class with **@RestController** and **@RequestMapping("/api/highscores")**.
- **Implement CRUD Operations**
  - Autowire the **HighScoreRepository** in **HighScoreController**.
  - Implement methods for **GET**, **POST**, **PUT**, and **DELETE** operations.

Here's a skeleton for your controller class:

java

Copy code

```
package com.example.highscore.controller;
```

```java
import com.example.highscore.model.HighScore;
import com.example.highscore.repository.HighScoreRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api/highscores")
public class HighScoreController {

    @Autowired
    private HighScoreRepository highScoreRepository;

    // Get all high scores
    @GetMapping
    public List<HighScore> getAllHighScores() {
        return highScoreRepository.findAll();
    }

    // Add a new high score
    @PostMapping
    public HighScore addHighScore(@RequestBody HighScore highScore) {
        return highScoreRepository.save(highScore);
    }

    // Update an existing high score
    @PutMapping("/{id}")
    public HighScore updateHighScore(@PathVariable Long id,
@RequestBody HighScore highScore) {
        // Additional logic to check if id exists
        return highScoreRepository.save(highScore);
    }

    // Delete a high score
    @DeleteMapping("/{id}")
    public void deleteHighScore(@PathVariable Long id) {
        // Additional logic to check if id exists
        highScoreRepository.deleteById(id);
    }
}
```

Documentation

Update your **README.md**:

markdown

Copy code

```
## Step 5: API Development for CRUD Operations

We created two new packages: `controller` and `repository`. Within these, we implemented the HighScoreRepository interface and HighScoreController class. The controller class is annotated with `@RestController` and `@RequestMapping("/api/highscores")`, and contains methods for CRUD operations.
```

Commit Changes

- Stage your changes: **git add .**
- Commit: **git commit -m "Implemented CRUD operations"**

Conclusion

You've now successfully implemented RESTful APIs for CRUD operations. These APIs interact with the H2 database through the JPA repository.