

الاعتماديات وإعدادات الـ Spring Boot

يتم تضمين إعدادات Spring Boot افتراضياً في ملف `pom.xml` عند إنشاء مشروع Spring Boot. يمكنك تعديل هذه الإعدادات في ملف `pom.xml` لإضافة أو تغيير الإعدادات. قاعده مع الـ `spring-boot-starter` لإدارة الـ `spring` بشكل مرن.

1. الاعتماديات الإضافية

لإضافة اعتماديات إضافية، نحتاج إلى تعديل ملف `pom.xml` لإضافة الـ `spring-boot-starter-data-jpa` و `spring-boot-starter-web` و `spring-h2-console` إلى سترتاج الـ `spring`.

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
  <groupId>com.h2database</groupId>
  <artifactId>h2</artifactId>
  <scope>runtime</scope>
</dependency>
```

2. إعدادات الـ Spring Boot

يتم تخزين إعدادات الـ Spring Boot في ملف `application.properties`. يمكنك تعديل هذه الإعدادات في ملف `application.properties` لإضافة أو تغيير الإعدادات. قاعده مع الـ `spring` بشكل مرن.

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

3. إنشاء الـ Spring Boot

لإنشاء الـ Spring Boot، نحتاج إلى تعديل ملف `pom.xml` لإضافة الـ `spring-boot-starter` إلى سترتاج الـ `spring`.

```

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private String email;

    // Getters and setters
}

```

4. مستودع واجهة إنشاء

User: لكي ان `userRepository` عمليات الواجهة هذه ستوفر `JpaRepository` من تمتد مستودع واجهة بانشاء ق

```

import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User, Long> {

    //
}

```

5. خدمة في المستودع استخد

البيانات: قاعدة عمليات لإجراء الخدمة في المستودع بإدخال ق

```

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;

@Service
public class UserService {

    @Autowired
    private UserRepository userRepository;

    public List<User> getAllUsers() {

```

```

        return userRepository.findAll();
    }

    public User getUserById(Long id) {
        return userRepository.findById(id).orElse(null);
    }

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

    public void deleteUser(Long id) {
        userRepository.deleteById(id);
    }
}

```

6. إنشاء المستخدمين

User الكيان مع لل تفاعل النهاية نقاط لتقديم المستخدمين بانشاء م

```

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/users")
public class UserController {

    @Autowired
    private UserService userService;

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

    @GetMapping("/{id}")
    public User getUserById(@PathVariable Long id) {
        return userService.getUserById(id);
    }
}

```

```

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

@DeleteMapping("/{id}")
public void deleteUser(@PathVariable Long id) {
    userService.deleteUser(id);
}
}

```

7. الـتطبيقات تشغيل

User الكيان على `UserController` عمليات تنفيذ على الـآن قادرًا تكون أن يجب بـك. الخاص `UserController` تطبيقات بتشغيل قم `UserController` النهاية نقاط باستخدام `UserController`.

أكبر بشكل توسيعه يمكنك `UserController` مشروع في بيئات كمصدر مستودع لاستخدام أساسية بنى الإعداد هذا يوفر تعقيدًا. أكثر أعمال منطق وتطبيقات الاستثناءات، معالجة مخصصة، أسئلة إضافة خلال من