Way2Million Webservice Anleitung

Way2Million Webservice Anleitung

Diese Anleitung führt durch die Erstellung des "Way2Million" Webservices mit Spring Boot. Der Service

beinhaltet eine Zinseszinsberechnung und ein einfaches Logging, um die Aufrufanzahl des Services

nachvollziehbar zu machen.

1. Projekt aufsetzen

- Gehe zu https://start.spring.io.

- Wähle die folgenden Optionen:

- Project: Maven

- Language: Java

- Spring Boot: Neueste Version

- Group: com.example

- Artifact: Way2Million

- Name: Way2Million

- Packaging: Jar

- Java Version: Java 11 oder höher

- Füge folgende Abhängigkeiten hinzu:

- Spring Web (für REST-Endpunkte)

- Spring Boot Actuator (optional für Monitoring)

- Spring Boot DevTools (optional für Entwicklung)

- Klicke auf Generate und entpacke das heruntergeladene Projekt.

2. Hauptklasse erstellen

Die generierte Hauptklasse Way2MillionApplication wird automatisch angelegt und startet die Spring-Boot-Anwendung:

```java

package com.example.Way2Million;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class Way2MillionApplication {

public static void main(String[] args) {

SpringApplication.run(Way2MillionApplication.class, args);

}

}

```

3. Controller für den Zinseszinsrechner erstellen

Erstelle einen Controller, der einen Endpunkt zur Berechnung des Zinseszinses bietet.

Erstelle eine neue Datei InterestCalculatorController.java im Paket com.example.Way2Million.controller.

```java

package com.example.Way2Million.controller;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class InterestCalculatorController {

@GetMapping("/calculate")

public String calculate(

@RequestParam double principal,

@RequestParam double rate,

@RequestParam int years) {

double amount = principal \* Math.pow((1 + rate / 100), years);

return String.format("Nach %d Jahren beträgt das Guthaben: %.2f", years, amount);

}

}

```

4. Logging für Aufrufzählung hinzufügen

Um zu verfolgen, wie oft der Webservice aufgerufen wurde, fügen wir einen Interceptor und einen Logger hinzu.

RequestCountingInterceptor:

Erstelle eine neue Datei RequestCountingInterceptor.java im Paket com.example.Way2Million.interceptor.

```java

package com.example.Way2Million.interceptor;

import org.springframework.stereotype.Component;

import org.springframework.web.servlet.HandlerInterceptor;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@Component

public class RequestCountingInterceptor implements HandlerInterceptor {

private int requestCount = 0;

@Override

public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) {

requestCount++;

System.out.println("Webservice aufgerufen: " + requestCount + " Mal");

return true;

}

public int getRequestCount() {

return requestCount;

}

}

```

WebConfig.java:

```java

package com.example.Way2Million.config;

import com.example.Way2Million.interceptor.RequestCountingInterceptor;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.annotation.Configuration;

import org.springframework.web.servlet.config.annotation.InterceptorRegistry;

import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration

public class WebConfig implements WebMvcConfigurer {

@Autowired

private RequestCountingInterceptor requestCountingInterceptor;

@Override

public void addInterceptors(InterceptorRegistry registry) {

registry.addInterceptor(requestCountingInterceptor);

}

}

```

5. Testklasse für den Interceptor erstellen

Erstelle eine Testklasse RequestCountingInterceptorTest.java unter src/test/java/com/example/Way2Million/interceptor.

```java

package com.example.Way2Million.interceptor;

import static org.junit.jupiter.api.Assertions.assertEquals;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

import org.springframework.mock.web.MockHttpServletRequest;

import org.springframework.mock.web.MockHttpServletResponse;

public class RequestCountingInterceptorTest {

private RequestCountingInterceptor requestCountingInterceptor;

@BeforeEach

public void setUp() {

requestCountingInterceptor = new RequestCountingInterceptor();

}

@Test

public void testRequestCountIncreasesOnEachRequest() throws Exception {

MockHttpServletRequest request = new MockHttpServletRequest();

MockHttpServletResponse response = new MockHttpServletResponse();

Object handler = new Object();

requestCountingInterceptor.preHandle(request, response, handler);

assertEquals(1, requestCountingInterceptor.getRequestCount());

requestCountingInterceptor.preHandle(request, response, handler);

assertEquals(2, requestCountingInterceptor.getRequestCount());

}

}

```

6. Anwendung starten

Starte die Anwendung über deine IDE oder das Terminal:

```bash

mvn spring-boot:run

```

Aufruf des Webservices:

Rufe die API im Browser oder einem REST-Client wie Postman auf:

http://localhost:8080/calculate?principal=10000&rate=5&years=10

Diese URL berechnet den Endbetrag bei einem Startkapital von 10.000, einem Zinssatz von 5% und einer Laufzeit von 10 Jahren.