Day15_Assignmnet

Case Study 2: User Profile Service with Caching ⍰ The Scenario: A social media platform, "Connectly," has a User Profile Service that fetches user data from a database. Another service, the Timeline Service, constantly calls the User Profile Service to fetch data for displaying posts. The database is a single point of failure and can sometimes become slow or unreachable. If the database goes down, the User Profile Service will fail, causing the Timeline Service to fail, and so on.

The Solution with a Circuit Breaker and Fallback Cache: Connectly implements a circuit breaker to protect against database failures.

• Closed State: The circuit breaker is closed, and the User Profile Service fetches data directly from the database.

• Threshold Trigger: The database experiences a period of high load, causing requests to the User Profile Service to timeout. The circuit breaker detects this based on its configured threshold (e.g., 90% of calls failing).

• Open State: The circuit breaker opens, preventing any more requests from hitting the database.

• Fallback with a Cache: In the fallbackMethod, the User Profile Service does not return a generic error. Instead, it attempts to retrieve the requested user data from a local cache (e.g., Redis or an in-memory cache).

◦ If the data is found in the cache, it's returned to the Timeline Service. This allows the Timeline Service to continue displaying timelines with slightly outdated but still usable data.

◦ If the data is not in the cache, it returns a generic fallback response.

• Half-Open State: After a set time, the circuit breaker transitions to the half-open state, allowing a few requests to test the database connection.

• Recovery: If the database has recovered, the test requests succeed, the circuit breaker closes, and the User Profile Service returns to fetching fresh data from the database.

Program:

Pom.xml:

```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>3.5.5-SNAPSHOT</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>com.example</groupId>
	<artifactId>User-profile-service</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>User-profile-service</name>
	<description>Demo project for Spring Boot</description>
	<url/>
	<licenses>
		<license/>
	</licenses>
	<developers>
		<developer/>
	</developers>
	<scm>
		<connection/>
		<developerConnection/>
		<tag/>
		<url/>
	</scm>
	<properties>
```

```
		<java.version>17</java.version>
		<spring-cloud.version>2025.0.0</spring-cloud.version>
	</properties>
	<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-actuator</artifactId>
		</dependency>
		<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>org.springframework.cloud</groupId>
			<artifactId>spring-cloud-starter-circuitbreaker-resilience4j</artifactId>
		</dependency>
<dependency>
   <groupId>org.springdoc</groupId>
   <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
   <version>2.6.0</version>
</dependency>
		<dependency>
			<groupId>com.h2database</groupId>
			<artifactId>h2</artifactId>
```

```xml
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>
    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>${spring-cloud.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
```

```
	<repositories>
		<repository>
			<id>spring-snapshots</id>
			<name>Spring Snapshots</name>
			<url>https://repo.spring.io/snapshot</url>
			<releases>
				<enabled>false</enabled>
			</releases>
		</repository>
	</repositories>
	<pluginRepositories>
		<pluginRepository>
			<id>spring-snapshots</id>
			<name>Spring Snapshots</name>
			<url>https://repo.spring.io/snapshot</url>
			<releases>
				<enabled>false</enabled>
			</releases>
		</pluginRepository>
	</pluginRepositories>

</project>
```

Entity:

User.java:

```
package com.example.user.profile.entity;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;


@Entity
@Table(name = "users")
public class User {
   @Id
   private String username;
   private String fullName;
   private String bio;


   public User() {}
   public User(String username, String fullName, String bio) {
      this.username = username;
      this.fullName = fullName;
      this.bio = bio;
   }
          public String getUsername() {
                    return username;
          }
          public void setUsername(String username) {
                    this.username = username;
          }
          public String getFullName() {
                    return fullName;
          }
          public void setFullName(String fullName) {
                    this.fullName = fullName;
          }
```

```
        public String getBio() {
                return bio;
        }
        public void setBio(String bio) {
                this.bio = bio;
        }

}
```

Controller:

UserController:

```
package com.example.user.profile.controller;
import org.springframework.web.bind.annotation.*;

import com.example.user.profile.entity.User;
import com.example.user.profile.service.UserProfileService;

@RestController
@RequestMapping("/api/users")
public class UserController {
  private final UserProfileService userProfileService;
  public UserController(UserProfileService userProfileService) {
    this.userProfileService = userProfileService;
  }

  @GetMapping("/{username}")
  public User getUser(@PathVariable String username) {
    return userProfileService.getUserProfile(username);
    }
}
```

Repository:

UserRepository:

```
package com.example.user.profile.repository;
import org.springframework.data.jpa.repository.JpaRepository;

import com.example.user.profile.entity.User;
public interface UserRepository extends JpaRepository<User, String> {
    User findByUsername(String username);
}
```

Service:

UserProfileService:

```
package com.example.user.profile.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import com.example.user.profile.entity.User;
public interface UserRepository extends JpaRepository<User, String> {
    User findByUsername(String username);

}
```