Case Study: User Profile Service

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>3.5.4</version>
                <relativePath/> <!-- lookup parent from repository -->
        </parent>
        <groupId>com.example</groupId>
        <artifactId>Circuit-Breaker-user-profile-service</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>Circuit-Breaker-user-profile-service</name>
        <description>Circuit Breaker Case Study 2: User Profile Service with
Caching</description>
        <url/>
        <licenses>
                <license/>
        </licenses>
        <developers>
                <developer/>
        </developers>
        <scm>
                <connection/>
                <developerConnection/>
                <tag/>
                <url/>
        </scm>
        <properties>
                <java.version>21</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>com.h2database</groupId>
                <artifactId>h2</artifactId>
                <scope>runtime</scope>
            </dependency>
            <dependency>
                <groupId>io.micrometer</groupId>
                <artifactId>micrometer-registry-prometheus</artifactId>
                <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>
</project>
```

application.properties

```
spring.application.name=Circuit-Breaker-user-profile-service
spring.datasource.url=jdbc:h2:mem:userdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.jpa.hibernate.ddl-auto=update
spring.h2.console.enabled=true
resilience4j.circuitbreaker.instances.userService.registeredHealthIndicator=true
resilience4j.circuitbreaker.instances.userService.slidingWindowType=COUNT_BASED
resilience4j.circuitbreaker.instances.userService.slidingWindowSize=10
resilience4j.circuitbreaker.instances.userService.failureRateThreshold=90
resilience4j.circuitbreaker.instances.userService.minimumNumberOfCalls=5
resilience4j.circuitbreaker.instances.userService.permittedNumberOfCallsInHalfOpenState=3
resilience4j.circuitbreaker.instances.userService.waitDurationInOpenState=5s
management.endpoints.web.exposure.include=*
management.endpoint.health.show-details=always
```

User.java

```
package com.example.userprofileservice.entity;
import jakarta.persistence.*;
@Entity
@Table(name = "users")
public class User {
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
          private Long id;

          private String name;
          private String email;
                public User() {
                        super();
                }
                public User(Long id, String name, String email) {
                        super();
                        this.id = id;
                        this.name = name;
                        this.email = email;
                }
                public Long getId() {
                        return id;
                }
                public void setId(Long id) {
                        this.id = id;
                }
```

```
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
}
```

UserRepository.java

```
package com.example.userprofileservice.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import com.example.userprofileservice.entity.User;
public interface UserRepository extends JpaRepository<User, Long>{
}
```

UserService.java

```
package com.example.userprofileservice.service;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.example.userprofileservice.entity.User;
import com.example.userprofileservice.repository.UserRepository;
import io.github.resilience4j.circuitbreaker.annotation.CircuitBreaker;
@Service
public class UserService {
	@Autowired
	  private UserRepository userRepository;

	@CircuitBreaker(name = "userService", fallbackMethod = "fallbackMethod")
	public User getUserById(Long id) {
	   User user = userRepository.findById(id).orElseThrow();
	   return user;
	}

	public String fallbackMethod(Throwable t) {
		return "Fallback: Service is currently unavailable. " + t.getMessage();
	}

	public User saveUser(User user) {
```

```
        User savedUser = userRepository.save(user);
        return savedUser;
    }
}
```

UserController.java

```
package com.example.userprofileservice.controller;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import com.example.userprofileservice.entity.User;
import com.example.userprofileservice.service.UserService;
@RestController
@RequestMapping("/users")
public class UserController {
    @Autowired
    private UserService userService;
    @GetMapping("/{id}")
    public ResponseEntity<User> getUser(@PathVariable Long id) {
        return ResponseEntity.ok(userService.getUserById(id));
    }

    @PostMapping
    public ResponseEntity<User> createUser(@RequestBody User user) {
        return ResponseEntity.ok(userService.saveUser(user));
    }
}
```