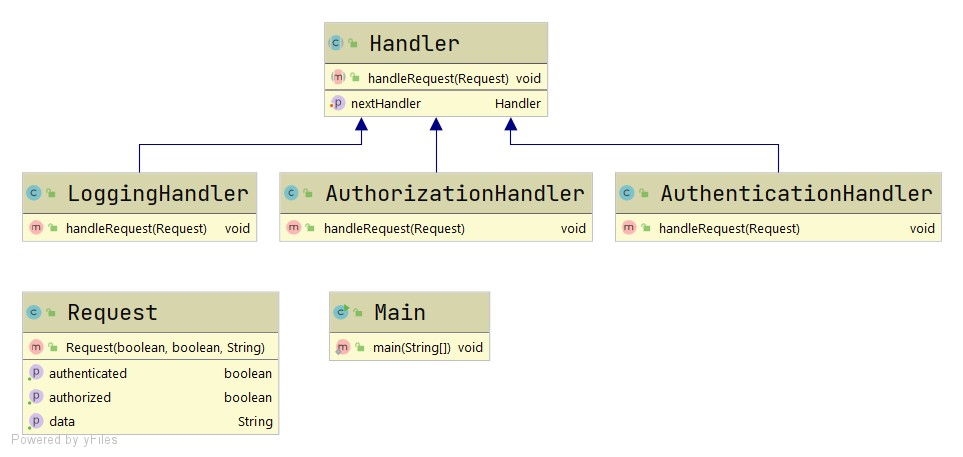
**Mẫu Chain of Responsibility:**

**Giới thiệu:**

Một chuỗi các đối tượng nhận yêu cầu và xử lý hoặc chuyển tiếp nó. Điều này giúp phân tán trách nhiệm xử lý yêu cầu và tạo sự linh hoạt trong việc thêm/bỏ các xử lý.

**Sau đây là ví dụ về mẫu Chain of Responsibility:**



Trong ví dụ này, AuthenticationHandler kiểm tra xem người dùng có được xác thực hay không và nếu có thì nó sẽ chuyển yêu cầu đến AuthorizationHandler. AuthorizationHandler kiểm tra xem người dùng có các quyền cần thiết để truy cập tài nguyên được yêu cầu hay không. Nếu cả hai lần kiểm tra đều thành công thì yêu cầu sẽ được chuyển đến LoggingHandler, nó chỉ ghi lại thông tin yêu cầu.

**Cấu trúc:**

1. Interface Handler:

Giao diện này xác định cấu trúc cơ bản cho tất cả các trình xử lý trong chuỗi. Mỗi trình xử lý phải triển khai phương thức handRequest(), phương thức này lấy đối tượng Yêu cầu làm đầu vào.

public interface Handler {

void handleRequest(Request request);

}

1. Class AuthenticationHandler:

Lớp này đại diện cho AuthenticationHandler trong chuỗi. Nó kiểm tra xem người dùng có được xác thực hay không và chuyển yêu cầu đến bộ xử lý tiếp theo nếu xác thực thành công.

public class AuthenticationHandler implements Handler {

private Handler nextHandler;

public AuthenticationHandler(Handler nextHandler) {

this.nextHandler = nextHandler;

}

*@Override*

public void handleRequest(Request request) {

if (authenticate(request)) {

System.***out***.println("User authenticated");

if (nextHandler != null) {

nextHandler.handleRequest(request);

}

} else {

System.***out***.println("Authentication failed");

}

}

private boolean authenticate(Request request) {

// Implement authentication logic here

return true; // Replace with actual authentication logic

}

}

1. Class AuthorizationHandler:

Lớp này đại diện cho AuthorizationHandler trong chuỗi. Nó kiểm tra xem người dùng có các quyền cần thiết để truy cập tài nguyên được yêu cầu hay không và chuyển yêu cầu đến trình xử lý tiếp theo nếu ủy quyền thành công.

public class AuthorizationHandler implements Handler {

private Handler nextHandler;

public AuthorizationHandler(Handler nextHandler) {

this.nextHandler = nextHandler;

}

*@Override*

public void handleRequest(Request request) {

if (!request.isAuthorized()) {

// Handle authorization failure

return;

}

nextHandler.handleRequest(request);

}

}

1. Class LoggingHandler:

Lớp này đại diện cho LoggingHandler trong chuỗi. Nó chỉ đơn giản là ghi lại thông tin yêu cầu.

public class LoggingHandler implements Handler {

private Handler nextHandler;

public LoggingHandler(Handler nextHandler) {

this.nextHandler = nextHandler;

}

*@Override*

public void handleRequest(Request request) {

// Log the request

System.***out***.println("Logging request: " + request);

nextHandler.handleRequest(request);

}

}

1. Class Request:

Lớp này đại diện cho một đối tượng yêu cầu được chuyển qua chuỗi xử lý. Nó chứa thông tin về người dùng, tài nguyên được yêu cầu và các dữ liệu liên quan khác.

public class Request {

private boolean authenticated;

private boolean authorized;

private String data;

public Request(boolean authenticated, boolean authorized, String data) {

this.authenticated = authenticated;

this.authorized = authorized;

this.data = data;

}

public boolean isAuthenticated() {

return authenticated;

}

public boolean isAuthorized() {

return authorized;

}

public String getData() {

return data;

}

}

1. Class Main

Lớp này thể hiện cách sử dụng các trình xử lý bằng cách tạo ra một chuỗi trách nhiệm và gửi yêu cầu thông qua nó.

public class Main {

public static void main(String[] args) {

Handler authenticationHandler = new AuthenticationHandler(null);

Handler authorizationHandler = new AuthorizationHandler(authenticationHandler);

Handler loggingHandler = new LoggingHandler(authorizationHandler);

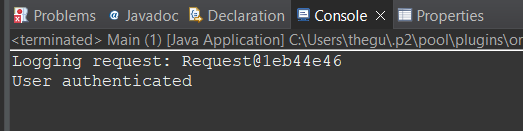
Request request = new Request(true, true, "Some data");

loggingHandler.handleRequest(request);

}

}

**Kết quả và thử nghiệm:**



Nếu xác thực hoặc ủy quyền không thành công, chuỗi sẽ bị hỏng và các trình xử lý tiếp theo sẽ không được gọi. Điều này cho phép chấm dứt sớm quá trình xử lý yêu cầu nếu có bất kỳ vấn đề nào với xác thực hoặc ủy quyền của người dùng.

Trong ví dụ này:

AuthenticationHandler kiểm tra xem người dùng có được xác thực không.

AuthorizationHandler kiểm tra xem người dùng có quyền truy cập không.

LoggingHandler ghi lại mọi hành động.

Mỗi handler xử lý một phần của yêu cầu và có thể chuyển yêu cầu tiếp theo trong chuỗi nếu yêu cầu đó đáp ứng các điều kiện cần thiết. Nếu một handler phát hiện yêu cầu không hợp lệ, nó có thể dừng chuỗi xử lý ngay lập tức.

**Lợi ích:**

Mẫu Chuỗi trách nhiệm rất hữu ích trong các trường hợp bạn cần xử lý một chuỗi các yêu cầu liên quan và mỗi yêu cầu có thể được xử lý bởi nhiều trình xử lý. Mẫu này cho phép bạn xâu chuỗi các trình xử lý lại với nhau và yêu cầu mỗi trình xử lý thực hiện một kiểm tra hoặc hành động cụ thể theo yêu cầu. Nếu trình xử lý không xử lý được yêu cầu, trình xử lý tiếp theo trong chuỗi có thể tiếp quản.