**Giới thiệu**

Java Relection là một core package trong thư viện chuẩn của Java. Mục đích của nó là cho phép chúng ta truy cập vào **gần như mọi thứ** bên trong đối tượng. "Dưới một góc độ khác"!

Chúng ta thường biết tới Java thông qua khái niệm hướng đối tượng như sau:

String str = "Hello Loda";

str.toUpperCase(); // Chúng ta gọi hàm toUpperCase() thông qua toán tử "."

// Mọi thứ trong đối tượng là khép kín, chúng ta phải gọi thông qua hàm public

Hoặc

public class Girl {

String name;

int age;

int atk;

int agi;

int def;

// ... Và 1000 thuộc tính khác

public static void main(String[] args) {

Girl girl = new Girl();

// Chúng ta thường phải nhớ tên thuộc tính để gọi nó ra

girl.name = "Ngoc Trinh";

// Giá sử class này có 100 thuộc tính là String.

// Bạn muốn set giá trị của tất cả trường String là "Ngoc Trinh"

// Bạn sẽ rất bối rối vs việc gọi từng thuộc tính bằng việc ".{tên thuộc tính}" như này.

// Có cách nào cho code duyệt tìm toàn bộ thuộc tính, cái nào là String thì đổi nó thành "Ngoc Trinh"?

}

}

Đúng vậy, khi chúng ta muốn gọi tên thuộc tính, mà lại không muốn gõ . và nhớ ra tên thuộc tính, thì làm như nào?

Bây giờ, chúng ta phải tiếp cận từ góc nhìn khác. Chúng ta sẽ ước mình có thể **duyệt hết tất cả** các thuộc tính của 1 class bằng vòng lặp. Rồi check xem thuộc tính có là String không? nếu có thì gán giá trị mới là "Ngoc Trinh"!

Để làm được điều này, chúng ta cần đào sâu vào Class và phá vỡ giới hạn của java truyền thống. Đây là lúc Java Reflection (Sự phản chiếu) vào trận.

**Java Reflection**

Java Reflecion cho phép bạn đánh giá, sửa đổi cấu trúc và hành vi của một đối tượng tại thời gian chạy (runtime) của chương trình. Đồng thời nó cho phép bạn truy cập vào các thành viên private (private member) tại mọi nơi trong ứng dụng, điều này không được phép với cách tiếp cận truyền thống.

**Lấy ra Thuộc tính (Field)**

Quay trở lại ví dụ trên, Chúng ta sẽ lấy ra toàn bộ thuộc tính của Girl. Tìm xem cái nào tên name và bổ sung giá trị mới cho nó.

public class Girl {

private String name;

public Girl() {

}

public Girl(String name) {

this.name = name;

}

public void setName(String name){

this.name = name;

}

@Override

public String toString() {

return "Girl{" +

"name='" + name + '\'' +

'}';

}

public static void main(String[] args) throws Exception {

Girl girl = new Girl(); // KHởi tạo đối tượng Girl

girl.setName("Ngoc trinh");

// Lay ra tat ca field cua object

// Chỉ để bạn xem ví dụ thôi, bỏ qua phần này nhé!

for(Field field : girl.getClass().getDeclaredFields()){

System.out.println();

System.out.println("Field: " +field.getName());

System.out.println("Type: " +field.getType());

}

// PHẦN CHÍNH

Field nameField = girl.getClass().getDeclaredField("name"); // Lấy ra field có tên "name" (nếu không tìm thấy, nó sẽ bắn NoSuchFieldException)

nameField.setAccessible(true); // Cho phép truy cập tạm thời. (Vì nó đang là Private mà)

// Bây giờ cái "nameField" đại diện cho thuộc tính "name" của mọi object có class Girl.

nameField.set(girl, "Bella"); // thay giá trị mới của `girl` bằng nameField.

System.out.println(girl);

}

}

// Output:

// Field: name

// Type: class java.lang.String

// Girl{name='Bella'}

**Lấy ra Hàm (Method)**

Vấn đề đặt ra, giống với field. Chúng ta cũng sẽ có nhu cầu duyệt tìm một method nào đó và sử dụng nó:

public static void main(String[] args) throws Exception {

Class<Girl> girlClass = Girl.class;

// Su dung getDeclaredMethods de lay ra nhung method cua class va cha no.

Method[] methods = girlClass.getDeclaredMethods();

for(Method method : methods){

System.out.println();

System.out.println("Method: " + method.getName());

System.out.println("Parameters: " + Arrays.toString(method.getParameters()));

}

// Lay ra method ten la setName va co 1 tham so truyen vao ->

// => chính là: setName(String name)

Method methodSetName = girlClass.getMethod("setName", String.class);

// Bây giờ methodSetName sẽ đại diện cho method setName(String name) của mọi object có class là Girl

Girl girl = new Girl(); // Tạo ra đối tượng Girl

// Thực hiện hàm setName() trên đối tượng girl, giá trị truyền vào là "Ngoc Trinh"

methodSetName.invoke(girl, "Ngoc Trinh");

System.out.println(girl);

}

**Lấy ra Constructor**

Lấy ra hàm khởi tạo của một class. Từ đó cho phép chúng ta cách tạo ra đối tượng từ theo một cách khác, thay vì new Class() như bình thường

public static void main(String[] args) {

Class<Girl> girlClass = Girl.class;

System.out.println("Class: " + girlClass.getSimpleName());

System.out.println("Constructors: " + Arrays.toString(girlClass.getConstructors())); // Lấy ra toàn bộ Constructor của class này

try {

// Tạo ra một object Girl từ class. (Khởi tạo không tham số)

Girl girl1 = girlClass.newInstance();

System.out.println("Girl1: " + girl1);

// Lấy ra hàm constructor với tham số là 1 string

// Chính là -> public Girl(String name) {}

Constructor<Girl> girlConstructor = girlClass.getConstructor(String.class);

Girl girl2 = girlConstructor.newInstance("Hello");

System.out.println("Girl2: " + girl2);

} catch (Exception e) {

// Exception xay ra khi constructor khong ton tai hoac tham so truyen vao khong dung

e.printStackTrace();

}

}

**Lấy ra Annotation trên Field, Method, Class**

Đúng vậy, đây cũng chính là một trong những phần quan trọng bậc nhất của Java Reflection. Cho phép chúng ta kiểm tra Class hiện tại đang được chú thích bởi những Annotation nào.

@SuppressWarnings("deprecation")

@Deprecated

public class Girl {

private String name;

public Girl() {

}

public Girl(String name) {

this.name = name;

}

@Nullable

public void setName(String name){

this.name = name;

}

@Override

public String toString() {

return "Girl{" +

"name='" + name + '\'' +

'}';

}

public static void main(String[] args) {

Class<Girl> girlClass = Girl.class;

System.out.println("Class: "+girlClass.getSimpleName()); // Lấy ra tên Class

for(Annotation annotation : girlClass.getDeclaredAnnotations()){

System.out.println("Annotation: " + annotation.annotationType()); // Lấy ra tên các Annatation trên class này

}

for(Method method: girlClass.getDeclaredMethods()){ // Lấy ra các method của class

System.out.println("\nMethod: " + method.getName()); //Tên method

for(Annotation annotation : method.getAnnotations()){

System.out.println("Annotation: " + annotation.annotationType()); // Lấy ra tên các Annatation trên method này

}

}

}

}