Java.lang - Class And Reflection

2016년 9월 25일 일요일 오전 1:00

Class 클래스

- Java는 클래스와 인터페이스의 메타데이터가 담긴 클래스
- 여기서의 메타데이터란 클래스의 이름, 멤버(필드와 메소드) 정보 등을 말한다.

---

getClass() 메소드

- Object에 정의된 메소드
- 인스턴스가 가지고 있는 Class 인스턴스 반환

```
Class<?> classInst = obj.getClass();
```

---

forName() 메소드

- Object에 정의된 static 메소드
- 클래스(패키지 포함)의 이름을 인자로 받아서 Class 객체 리턴

```
public class ClassExample {
     public static void main(String[] args){
          BMW i8 = new BMW("i8", "white", "hybrid");

          Class<?> classInst = i8.getClass();

          System.out.println(classInst.getName()); // 패키지를 포함한 클래스 풀 네임
          System.out.println(classInst.getSimpleName()); // 패키지를 제외한 클래스 네임
          System.out.println(classInst.getPackage().getName()); // 패키지의 네임

          try {
               Class<?> classInst2 = Class.forName("com.java.BMW");

               System.out.println(classInst.getName());
               System.out.println(classInst.getSimpleName());
               System.out.println(classInst.getPackage().getName());
          } catch (ClassNotFoundException e) {
               e.printStackTrace();
          }
     }
}
```

리플렉션 (Reflection)

- Class 객체를 이용하여 클래스의 생성자, 필드, 메소드 정보를 알아내는 것을 리플렉션이라 한다.
- Run Time(실행시점)에 동적으로 특정 클래스의 정보를 객체로부터 추출하는 기법
- Java.lang.reflect에 리플렉션 용 Constructor, Field, Method, Annotation 클래스가 선언되어 있어서 각각에 대한 메타데이터를 담을 수 있다.

```
public class BMW {
     private String name;
     private String color;
     private String oil;
     private String oil;


     public BMW(String name, String color, String oil, int price){
          this.name = name;
          this.color = color;
          this.oil = oil;
          this.price = price;
     }

     public String drive(String gear){ return gear; }
     public String parking(String gear){ return gear; }
     public void EngineStart(){ }
}

public class GetClass {
     public static void main(String[] args){
```

```
BMW i8 = new BMW("i8", "white", "hybrid", 2500);

Class<?> classInst = i8.getClass();

System.out.println("클래스 이름 : "+classInst.getName()+"\n");

System.out.println("Constructor Info");
// 생성자 정보
Constructor<?>[] constructors = classInst.getDeclaredConstructors();
for(Constructor<?> constructor : constructors){
    System.out.println(constructor.getName());

    Class<?>[] params = constructor.getParameterTypes();
    printParams(params);
}

System.out.println("Field Info");
// 변수 정보
Field[] fields = classInst.getDeclaredFields();
for(Field field : fields){
    System.out.println(field.getType().getSimpleName()
            +" "+field.getName());
}

System.out.println("Method Info");
// 메소드 정보
Method[] methods = classInst.getDeclaredMethods();
for(Method method : methods){
    System.out.println(method.getName());

    Class<?>[] params = method.getParameterTypes();
    printParams(params);
}

// 필드 수정하기
try {
    Field colorField = classInst.getDeclaredField("color");

    // private 멤버 접근 가능
    colorField.setAccessible(true);

    //(적용할 해당 Class 인스턴스의 클래스의 인스턴스, 값)
    colorField.set(i8, "black");

    System.out.println(colorField.get(i8));
    // get(적용할 해당 Class 인스턴스의 클래스의 인스턴스)
} catch (Exception e) { e.printStackTrace();}

// 메소드 호출
try {
    Class<?> classInst2 = Class.forName("com.java.BMW");
    Constructor<?> inst2Cons = classInst2.getConstructor(
            new Class[]{String.class, String.class, String.class, Integer.TYPE});


    Method drive =
        classInst2.getMethod("drive", new Class[]{String.class});
    // getMethod(메소드명, 파라미터 타입)

    System.out.println(
        drive.invoke(
            //해당 생성자를 통해 인스턴스 생성
            inst2Cons.newInstance(
```

```
            drive.invoke(
                // 해당 생성자를 통해 인스턴스 생성
                inst2Cons.newInstance(
                new Object[]{"520d", "white", "disel", new Integer(250)}),

                // 메소드에 전달할 인자
                new Object[]{new String("D")})
        );
    } catch (Exception e) { e.printStackTrace(); }
}

public static void printParams(Class<?>[] params){
    for(int i=0;i<params.length;i++){
        System.out.println(params[i].getName());
    }
}
}
```

---

newInstance() - 리플렉션을 이용한 동적 인스턴스 생성

- Class 객체를 이용하면 new 연산자 없이도 동적으로 인스턴스 생성이 가능하다.

```
try{

    Class<?> clazz = Class.forName("com.java.BMW");
    Object obj = clazz.newInstance();

    // 반환형이 Object이므로 해당 객체를 사용하려면 해당 객체 타입에 맞게 캐스팅 필요
    BMW i3 = (BMW)obj;
    i3.drive("D");

}catch(Exception e){
    e.printStackTrace();
}
```

- 기본 생성자가 없을 시에는 Constructor 객체를 추출 후 Constructor 객체를 대상으로 newInstance() 호출
