**Java Reflections – Loading Classes, Constructors and Methods at Run Time:**

Java reflections package is used if we want to load any class, it’s constructor or methods at run time. In this one we only see what is needed for our selenium tutorials.

Here we created a package “actions” and a class under it called HomePage. We have created two static variables and a constructor to initialize those variables. We have also created four methods as well.

We have created another package “drivers” and class under it “ContactActions”. In this class we don’t create any object for HomePage class but we dynamically access this class and it’s constructor and methods at run time.

**HomePage.java of actions package:**

**package** actions;

**public** **class** HomePage {

**public** **static** **int** *globala*;

**public** **static** String *globalstr*;

**public** HomePage(**int** a, String str) {

*globala* = a;

*globalstr* = str;

}

**public** **int** getSumofTwoWithGlobal(**int** a) {

**return**(*globala*+a);

}

**public** **int** getSumofThreeWithGlobal(**int** a, **int** b) {

**return**(*globala*+a+b);

}

**public** String concatTwoWithGlobal(String str1) {

**return**(*globalstr*+str1);

}

**public** String concatThreeWithGlobal(String str1, String str2) {

**return**(*globalstr*+str1+str2);

}

}

For accessing a class at run time, we use Class.forName() method. We have to pass the package name followed by “.” followed by class name. It returns a Class object. By writing Class<?>, you're declaring a Class object which can be of any type (? is a wildcard). The Class type is a type that contains metainformation about a class.

Yu can get all the methods of the class by using getDeclaredMethods() which returns an array of public methods the class has. So we save all these in Method[].

Now you can print all the methods using Arrays.toString() method or you can loop through the Method[] array and print one by one.

To invoke any method first we have to load the method. This we do it with

Method mc = cls.getDeclaredMethod("getSumofTwoWithGlobal",**int**.**class**);

For loading a method we use getDeclaredMethod() and we have to pass the name of the method and type of variables the methoduse. If we use integer parameters in the method we use int.class and if we use string parameters it is string.class.

Next before invoking the method, we need to load the constructor. We use getConstructor() method and the variable types the constructor use.

Constructor<?> constructor = cls.getConstructor(**int**.**class**, String.**class**);

Next, we invoke the method using invoke() method. We first pass the constructor.newInstance() as parameter and then the actual parameters the method use. Since this method returns an integer, we capture that in variable sum.

**int** sum = (**int**) mc.invoke(constructor.newInstance(1, "Selenium"), 2);

**ContactActions.java of drivers package:**

**package** drivers;

**import** java.lang.reflect.Constructor;

**import** java.lang.reflect.InvocationTargetException;

**import** java.lang.reflect.Method;

**import** java.util.Arrays;

**public** **class** ContactActions {

**public** **static** **void** main(String[] args) **throws** ClassNotFoundException, NoSuchMethodException, SecurityException, IllegalAccessException, IllegalArgumentException, InvocationTargetException, InstantiationException {

Class<?> cls = Class.*forName*("actions.HomePage");

//Print all public methods

Method[] publicMethods = cls.getDeclaredMethods();

System.***out***.println(Arrays.*toString*(publicMethods));

**for**(**int** i=0; i<publicMethods.length; i++) {

System.***out***.println(publicMethods[i]);

}

//Invoke method getSumofTwoWithGlobal

Method mc = cls.getDeclaredMethod("getSumofTwoWithGlobal",**int**.**class**);

Constructor<?> constructor = cls.getConstructor(**int**.**class**, String.**class**);

**int** sum = (**int**) mc.invoke(constructor.newInstance(1, "Selenium"), 2);

System.***out***.println("Sum for getSumofTwoWithGlobal is "+sum);

//Invoke method getSumofThreeWithGlobal

mc = cls.getDeclaredMethod("getSumofThreeWithGlobal",**int**.**class**,**int**.**class**);

constructor = cls.getConstructor(**int**.**class**, String.**class**);

sum = (**int**) mc.invoke(constructor.newInstance(1, "Selenium"), 2, 3);

System.***out***.println("Sum for getSumofThreeWithGlobal is "+sum);

//Invoke method concatTwoWithGlobal

mc = cls.getDeclaredMethod("concatTwoWithGlobal", String.**class**);

constructor = cls.getConstructor(**int**.**class**, String.**class**);

String cnt = (String) mc.invoke(constructor.newInstance(1, "Selenium"), " Automation");

System.***out***.println(cnt);

//Invoke method concatThreeWithGlobal

mc = cls.getDeclaredMethod("concatThreeWithGlobal", String.**class**, String.**class**);

constructor = cls.getConstructor(**int**.**class**, String.**class**);

cnt = (String) mc.invoke(constructor.newInstance(1, "Selenium"), " Automation", " Testing");

System.***out***.println(cnt);

}

}