Design Patterns - MVC Pattern

设计模式-MVC模式

MVC Pattern stands for Model-View-Controller Pattern. This pattern is used to separate application's concerns.

MVC设计模式 是Model-View-Controller 模式的**代表**（stand for）。该设计模式主要是用来**分离**（separate）应用的**关注点**（concerns）

注：

Model：模型 View：视图 Controller：控制器

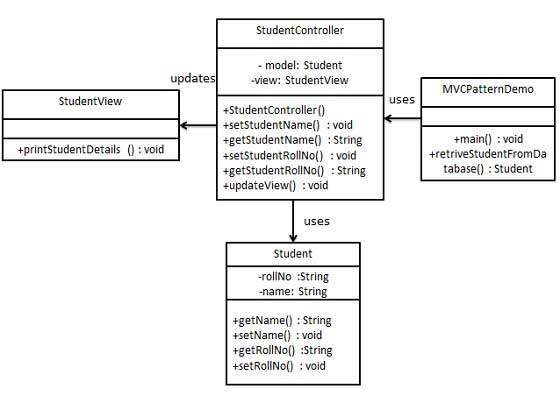
* **Model** - Model represents an object or JAVA POJO carrying data. It can also have logic to update controller if its data changes.
* **View** - View represents the visualization of the data that model contains.
* **Controller** - Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.
* **Model** - Model 代表一个对象（object）或者java普通对象（POJO）装载的数据，如果它的数据发生变化，它也可以有逻辑到update controller。
* **View** – View代表可视化模型所包含的数据。
* **Controller** - Controller 作用于model和view。它控制数据流向model对象，并且在数据发生改变的时候更新视图，它保持者view和model的分离。

We are going to create a *Student* object acting as a model.*StudentView* will be a view class which can print student details on console and *StudentController* is the controller class responsible to store data in *Student* object and update view*StudentView* accordingly.

首先我们会创建一个Student对象来扮演model，StudentView将作为一个view类，它能在控制台输出学生的详细信息，StudentController 作为一个controller 负责把数据存储到student对象并且相应的更新view StudentView

*MVCPatternDemo*, our demo class, will use *StudentController* to demonstrate use of MVC pattern.

*MVCPatternDemo，*我们的demo类，将使用StudentController来展示如何使用MVC模式



Step 1

第一步

Create Model.

创建 Model

*Student.java*

public class Student {

private String rollNo;

private String name;

public String getRollNo() {

return rollNo;

}

public void setRollNo(String rollNo) {

this.rollNo = rollNo;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

Step 2

第二步

Create View.

创建view

*StudentView.java*

public class StudentView {

public void printStudentDetails(String studentName, String studentRollNo){

System.out.println("Student: ");

System.out.println("Name: " + studentName);

System.out.println("Roll No: " + studentRollNo);

}

}

Step 3

第三步

Create Controller.

创建Controller

*StudentController.java*

public class StudentController {

private Student model;

private StudentView view;

public StudentController(Student model, StudentView view){

this.model = model;

this.view = view;

}

public void setStudentName(String name){

model.setName(name);

}

public String getStudentName(){

return model.getName();

}

public void setStudentRollNo(String rollNo){

model.setRollNo(rollNo);

}

public String getStudentRollNo(){

return model.getRollNo();

}

public void updateView(){

view.printStudentDetails(model.getName(), model.getRollNo());

}

}

Step 4

第四步

Use the *StudentController* methods to demonstrate MVC design pattern usage.

使用StudentController的方法展示MVC设计模式的的使用

*MVCPatternDemo.java*

public class MVCPatternDemo {

public static void main(String[] args) {

//fetch student record based on his roll no from the database

Student model = retriveStudentFromDatabase();

//Create a view : to write student details on console

StudentView view = new StudentView();

StudentController controller = new StudentController(model, view);

controller.updateView();

//update model data

controller.setStudentName("John");

controller.updateView();

}

private static Student retriveStudentFromDatabase(){

Student student = new Student();

student.setName("Robert");

student.setRollNo("10");

return student;

}

}

Step 5

第五步

Verify the output.

校验输出

Student:

Name: Robert

Roll No: 10

Student:

Name: John

Roll No: 10