MVC

What is MVC

- It’s a way to structure your applications
- The architecture used today for web applications uses MVC.
- The Model-View-Controller (MVC) separates an application into three main logical components:
  - The model
  - The view
  - The controller

MVC needs all these components to be separated as different objects.

The Component: Model

model layer -- data layer.

- The model represents data and the rules that govern access to and updates of data.
- In enterprise software, a model often serves as a software approximation of a real-world process.
- The data is “modelled” in a way it’s easy to store, retrieve, and edit.
- It defines rules to for data the way it is represents a real-world object.
- Any request from user would lead to the appropriate model which returns a data representation.
- This model remains same, only the view is different for various request.
- A model also define rules for updating a model’s data.

Model contains Java classes, View is used to display the data and controller contains servelets.

The Component: Controller

- The controller translates the user's interactions with the view into actions that the model will perform.
- User interactions could be button clicks or menu selections, whereas in an enterprise web application, they appear as GET and POST HTTP requests.
- Depending on the context, a controller may also select a new view - for example, a web page of results -- to present back to the user.
- The controller is the like the housekeeper of the application – it performs coordination between model and view to entertain a user request.

Controller(2)

- The primary function of a controller is to call and coordinate with the model to fetch any necessary resources required to act.
- Usually, on receiving a user request, the controller calls the appropriate model for the task at hand.
- There may be a primary controller that is responsible for receiving all the requests and calling the specific controller for specific actions.

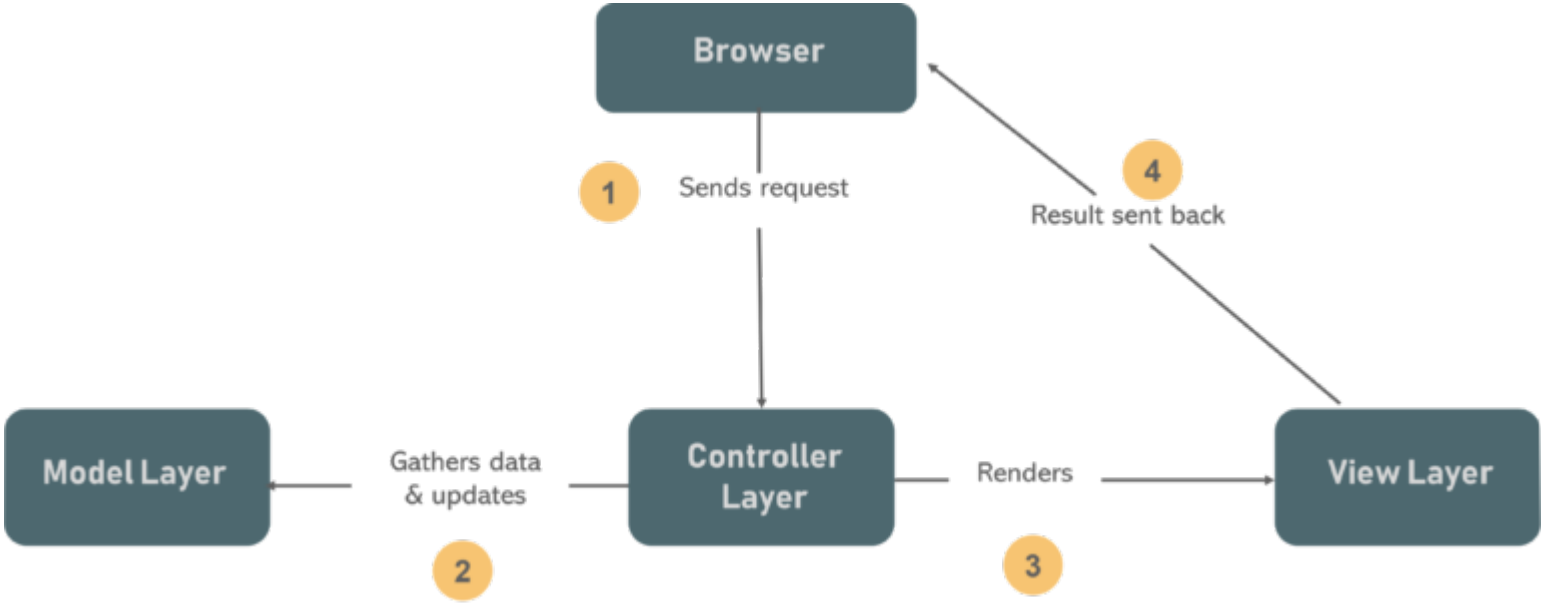
The Component: View

- View is responsible for rendering the data received from the model.
- It specifies exactly how the model data should be presented.
- If the model data changes, the view updates its presentation by using a push model or pull model.
- Presents the data to the user whenever asked for.
- There may be
  - Pre-designed templates
  - Multiple views per model based on the requirements.

advantages--
1. components are easy to maintain bcz of less dependency.
2. application becomes more understandable.
3. each layer is maintained separately therefore no need to deal with massive code.
4. extending and testing of application is easier.

Example : Online stationery shop

- Consider a particular case : Pens
- The controller responsible for pen related queries : “pens_controller.php” (say)
- A model that will store the data regarding the pens : “pens_model.php” (say)
- Views to show pen: a list of pens, a table displaying pens, a page per pen with specifications, etc.



First, the “pens_controller.php” handles the user request as a GET or POST request.

The controller then examines the request and the parameters and calls the required model – “pens_model.php”.

The controller asks the model to return the list of available pens.

Now, the model searches the database for the necessary information, applies logics if necessary, and returns the data to the controller.

The controller then picks an appropriate view and presents the data to the user.

MVC using Java : Example 1

- Object of Student Class: The model.
- StudentView class: The view class to print details on the console.
- StudentController class : A controller that stores data in studentObject and updates StudentView accordingly

model

```
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;
   }
}
```

view

```
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);
   }
}
```

controller

```
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());
   }
}
```

Java main class

```
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;
   }
}
```

MVC using Java : Example 2

- Objec of Course Class : The model
- CourseView Class : the view for Course model
- CourseContoller Class : The controller

The model

```
package MyPackage;

public class Course {
	private String CourseName;
	private String CourseId;
	private String CourseCategory;

	public String getId() {
	   return CourseId;
	}

	public void setId(String id) {
	   this.CourseId = id;
	}

	public String getName() {
	   return CourseName;
	}

	public void setName(String name) {
	   this.CourseName = name;
	}

	public String getCategory() {
		return CourseCategory;
	    }

	public void setCategory(String category) {
		this.CourseCategory = category;
	    }
  }
```

The view

```
package MyPackage;

public class CourseView {
      public void printCourseDetails(String CourseName, String CourseId, String CourseCategory)
{
         System.out.println("Course Details: ");
         System.out.println("Name: " + CourseName);
         System.out.println("Course ID: " + CourseId);
         System.out.println("Course Category: " + CourseCategory);
      }
   }
```

The controller

```
package MyPackage;

public class CourseController {
      private Course model;
      private CourseView view;

      public CourseController(Course model, CourseView view){
         this.model = model;
         this.view = view;
      }

      public void setCourseName(String name){
         model.setName(name);
      }

      public String getCourseName(){
         return model.getName();
      }

       public void setCourseId(String id){
         model.setId(id);
      }
```

The controller(2)

```
    public String getCourseId(){
       return model.getId();
    }

    public void setCourseCategory(String category){
           model.setCategory(category);
    }

        public String getCourseCategory(){
           return model.getCategory();
    }
    public void updateView(){
       view.printCourseDetails(model.getName(), model.getId(), model.getCategory());
    }
}
```

The java main class

```
package MyPackage;

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

         Course model  = retriveCourseFromDatabase();

         CourseView view = new CourseView();

         CourseController controller = new CourseController(model, view);

         controller.updateView();

         //update model data
         controller.setCourseName("Python");
         System.out.println("After updating, Course Details are as follows");

         controller.updateView();
      }

      private static Course retriveCourseFromDatabase(){
         Course course = new Course();
         course.setName("Java");
         course.setId("01");
         course.setCategory("Programming");
         return course;
      }
   }
```