

The Model View Controller Pattern



Team presentation



Vlad-Marian Lupu

vlad.lupu.19@ull.edu.es



Gabriel Rodriguez Hernandez

gabriel.rodriguez.30@ull.edu.es

Index

- 📌 1. Introduction
- 📌 2. Components
- 📌 3. Advantages & disadvantages
- 📌 4. Where can we use it?

Important

- It can be considered outdated
- Other patterns are used, such as:
 - Model-view-presenter
 - Multi-tier Architecture
- You still have to learn how to use it

1. Introduction

A bit of history and what
exactly is the MVC?



The MVC Pattern beginnings

- 💡 It was used for the first time in 1979 in Smalltalk-79.
- 💡 Initially it has 4 components: Model, View, Thing and Editor.
- 💡 The use of MVC grew after the introduction of NeXT's WebObjects in 1996.
- 💡 A note about MVC from the creator.

What is the MVC Pattern?

- 🔍 Architecture pattern.
- 🔍 Create scalable and modular projects.
- 🔍 Has three components
 - 🏷 Model
 - 🏷 View
 - 🏷 Controller
- 🔍 The applications created are easy to understand for others programmers.
- 🔍 Allows greater reuse of components.
- 🔍 Faster development process

2. Components

What are the elements that
make up the pattern



2.1

The Model

Contains only application data



What is The Model

The **central** component of the pattern. It is the application's dynamic data structure, independent of the user interface. It manages the **data**, **logic** and **rules** of the application.



Business logic

- Rules that determine how data is created.
- The subsequent delivery of results to the user through the interface

2.2

The View

Presents the model data to the user



What is The View

The view is the layer that interacts with the user, and gets information about him, this is called **events**.



What does the View With the data?

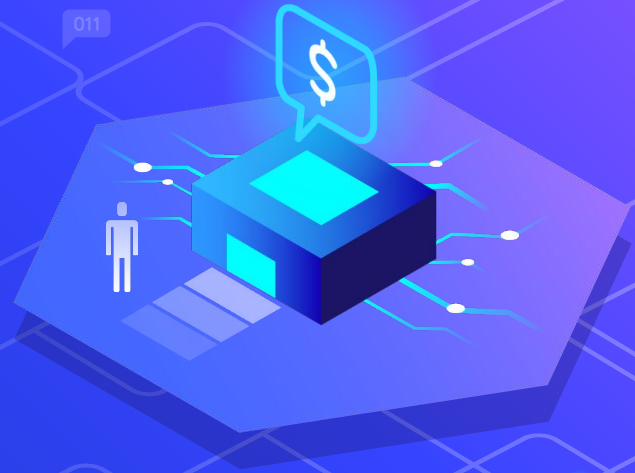
The View Knows how to the Model data, but doesn't know what is this information or what the user can do to manipulate it.

The View isn't intended to manage data. Is created to show the results and interact with the users.

2.3

The Controller

It maps user actions - model actions



What is The Controller

It serves as the link between the view and the model, receiving and responding to events, typically user actions and invoking changes on the model and, most likely, in the view.



2.4

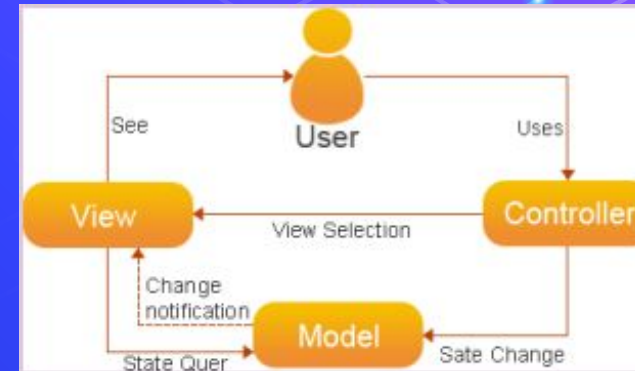
Functionality

How they relate



Are related...

- 📌 The view shows the Model to the user
- 📌 The user uploads events
- 📌 The view send the events to the controller
- 📌 The controller manipulates the module as indicated by the event
- 📌 The model performs operations to modify the information
- 📌 The controller shows a new view to the user

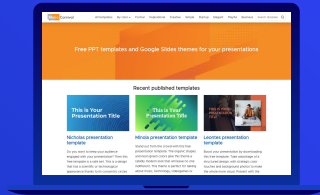


MVC flow

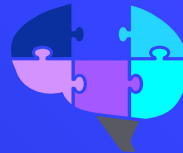
User



View



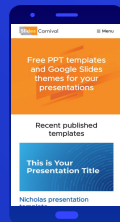
Controller



Model



View



Example



2.5

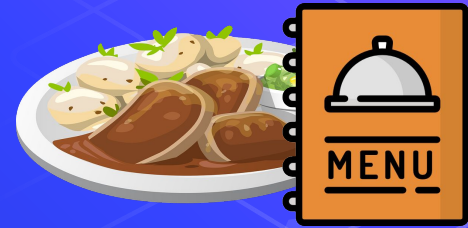
Real life example

Let's see if you understood



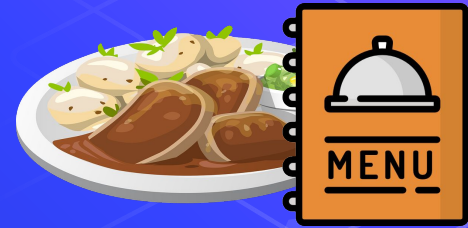
Restaurant example

We have:



Restaurant example

We have:



✓ Controller

Restaurant example

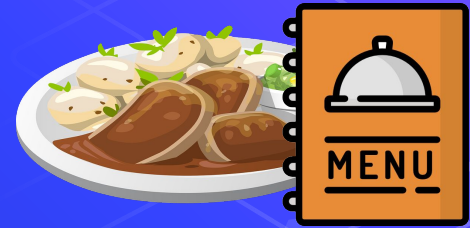
We have:



✓ Controller



✓ Model



Restaurant example

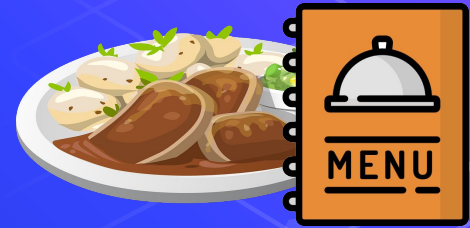
We have:



✓ Controller

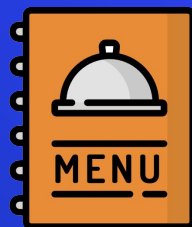


✓ Model

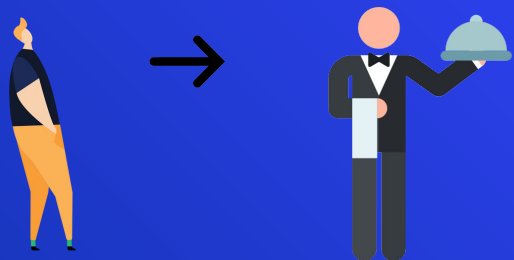


✓ View

Let's see the schema again



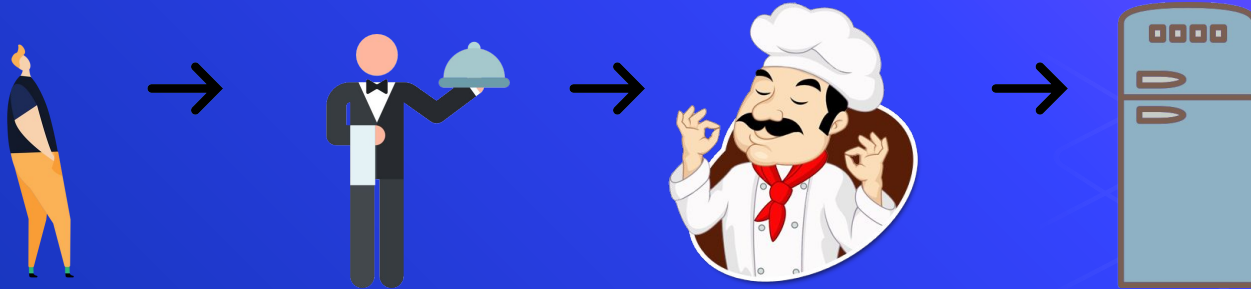
Let's see the schema again



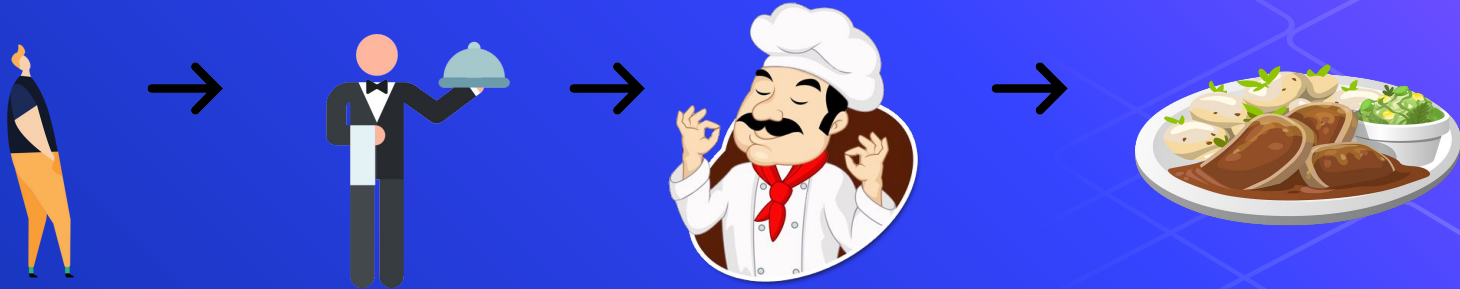
Let's see the schema again



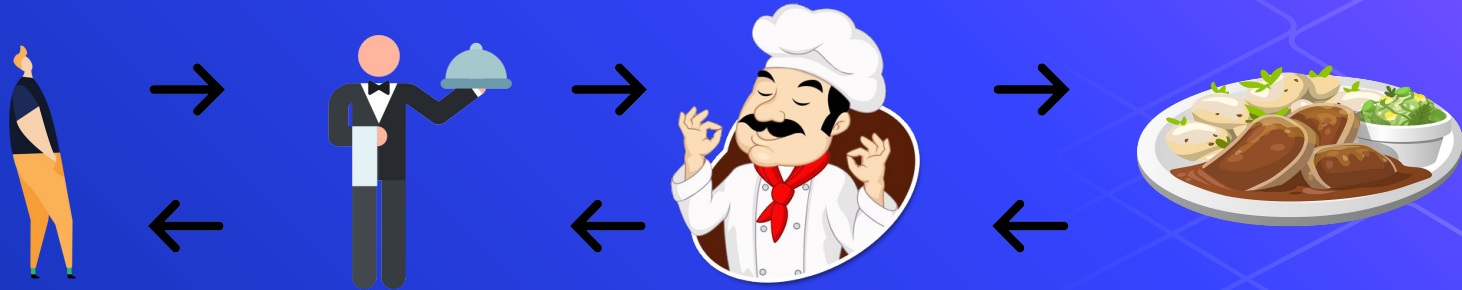
Let's see the schema again



Let's see the schema again



Let's see the schema again



3.

Advantages & disadvantages

What do I gain by using it?



Reasons to use it

- ✓ Cleaner, simpler, more easily maintainable and more robust systems.
- ✓ Increased team development speed.
- ✓ Multiple views from the same model
- ✓ Easy to carry out unit tests.
- ✗ You have to stick to the conventions and the pattern
- ✗ Learning curve

4. Where can we use it?

There are a lot of possibilities



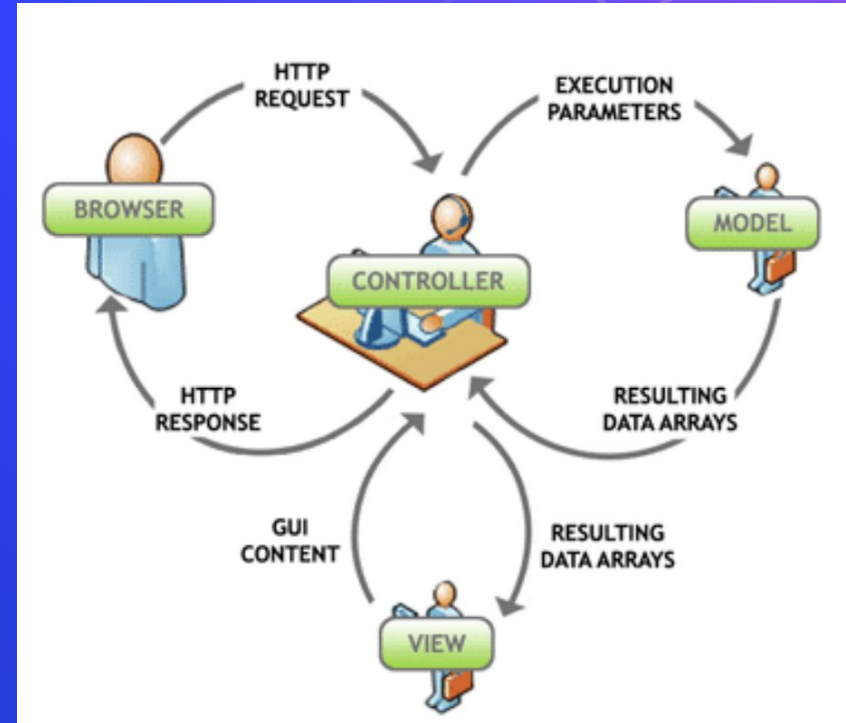
Where can we use it?

- Web applications
- In a lot programming languages such as
 - Java
 - Javascript
 - Python
 - Ruby
 - Pearl
 - SmallTalk, etc.



Use in web applications

- Originally developed for desktop computing
- Adapt for WWW in major programming languages
- Server- Browser relationship



MVC Pattern in Javascript

```
class Model {  
  constructor() {}  
}  
  
class View {  
  constructor() {}  
}  
  
class Controller {  
  constructor(model, view) {  
    this.model = model  
    this.view = view  
  }  
}  
  
const app = new Controller(new Model(), new View())
```



Can you see it?

View Layer in Javascript

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Todo App</title>
    ...
  </head>

  <body>
    <div id="root"></div>
    ...
  </body>
</html>
```

```
class View {
  constructor() {}

  // Create an element with an optional CSS class
  createElement(tag, className) {
    const element = document.createElement(tag)
    if (className) element.classList.add(className)

    return element
  }

  // Retrieve an element from the DOM
  getElement(selector) {
    const element = document.querySelector(selector)

    return element
  }
}
```

Model Layer in Javascript

```
class Model {
  constructor() {
    // The state of the model, an array of todo objects, prepopulated with some data
    this.todos = [
      {id: 1, text: 'Run a marathon', complete: false},
      {id: 2, text: 'Plant a garden', complete: false},
    ]
  }

  addTodo(todoText) {
    const todo = {
      id: this.todos.length > 0 ? this.todos[this.todos.length - 1].id + 1 : 1,
      text: todoText,
      complete: false,
    }







    this.todos.push(todo)
  }

  // Map through all todos, and replace the text of the todo with the specified id
  editTodo(id, updatedText) {
    this.todos = this.todos.map((todo) =>
      todo.id === id ? {id: todo.id, text: updatedText, complete: todo.complete} : todo,
    )
  }
}
```

Controller Layer in Javascript

```
class Controller {  
  constructor(model, view) {  
    this.model = model  
    this.view = view  
  
    // Display initial todos  
    this.onTodoListChanged(this.model.todos)  
  }  
  
  onTodoListChanged = (todos) => {  
    this.view.displayTodos(todos)  
  }  
}
```

See also

-  [Multi-tier architecture](#)
-  [Hierarchical model-view-controller](#)
-  [Model-view-adapter](#)
-  [Model-view-presenter](#)
-  [Observer pattern](#)
-  [Strategy pattern](#)

References

- 🌐 Presentation template by [SlidesCarnival](#)
- 🌐 Photographs by [Unsplash](#)
- 🌐 MVC [Wikipedia](#)
- 🌐 [Advantages & Disadvantages](#)
- 🌐 [Todo code example](#)
- 🌐 [More about MVC](#)



Thanks!

Any questions?

You can contact us at:

- ⬡ gabriel.rodriguez.30@ull.edu.es
- ⬡ vlad.lupu.19@ull.edu.es

