MVC

What it is and how it works

What is MVC?

Model

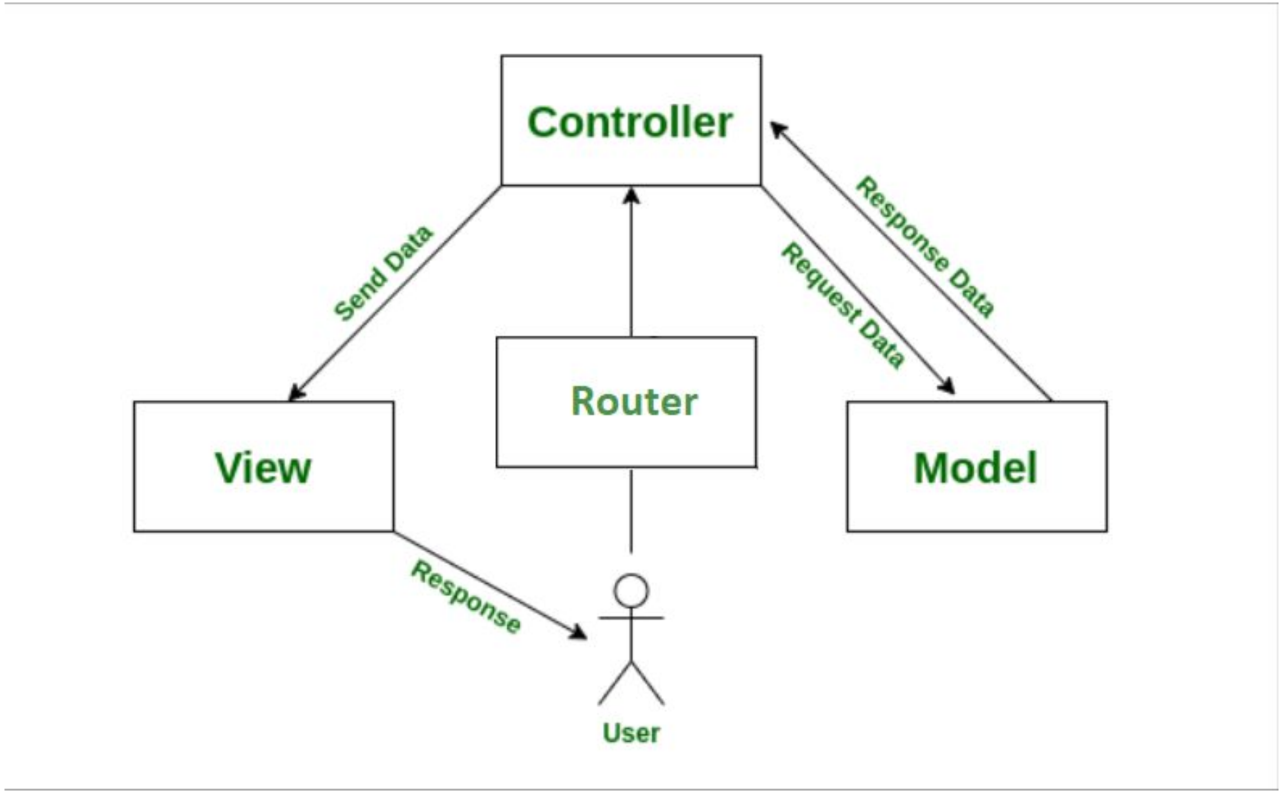
View

Controller

What is MVC for?

Pattern for programming to break up code and easily share it across many different collaborators

Code is separated into sections that have their own purpose

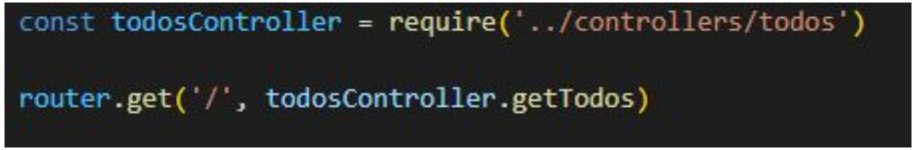


Router

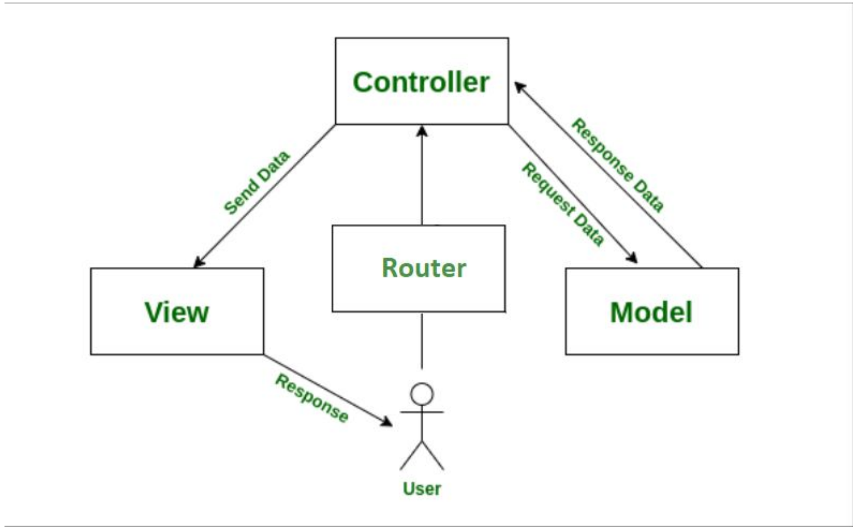
It starts with the User making a Request to the Router Section.

The Router finds a specific Controller to do a specific action.

A separate Router section allows for flexible and easy updates.



```
const todosController = require('../controllers/todos')

router.get('/', todosController.getTodos)
```

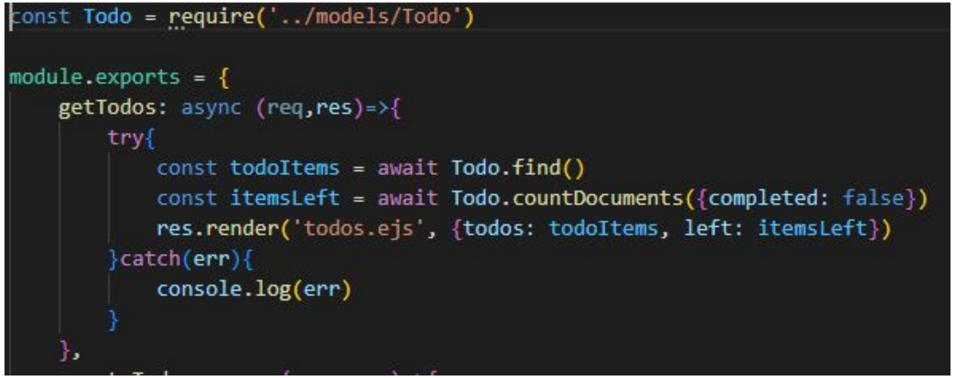


Controller

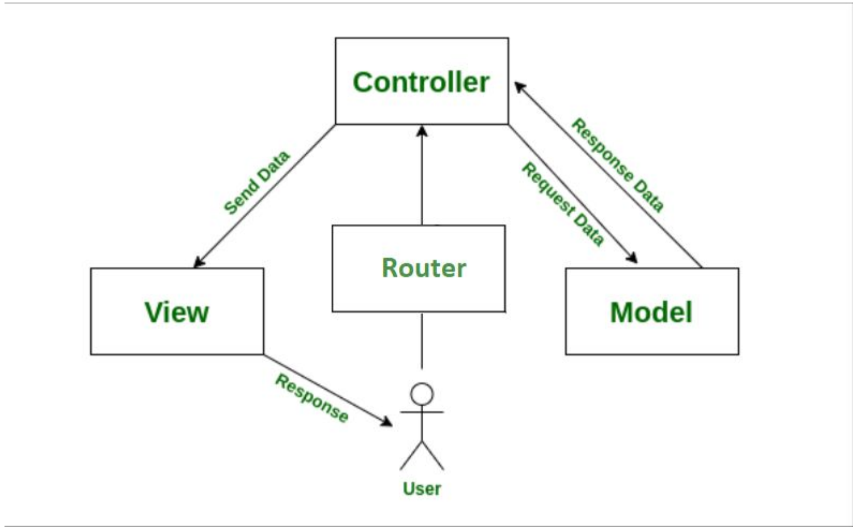
The Router specifies a route and Method to find the specific Controller.

Controller Section first gathers information from Model Section.

Then, it manipulates Model to act.



```
const Todo = require('../models/Todo')

module.exports = {
    getTodos: async (req,res)=>{
        try{
            const todoItems = await Todo.find()
            const itemsLeft = await Todo.countDocuments({completed: false})
            res.render('todos.ejs', {todos: todoItems, left: itemsLeft})
        }catch(err){
            console.log(err)
        }
    },
```



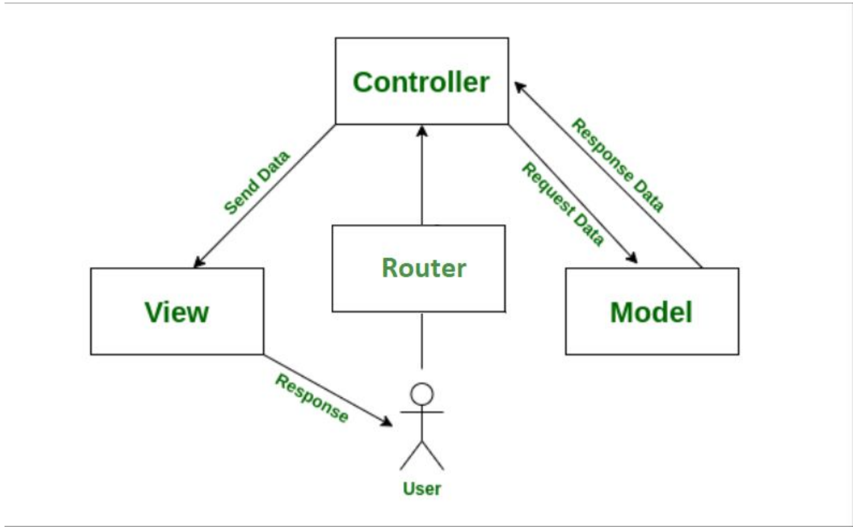
Model

Model Section handles data logic and interacts with the database.

The Controller should never do data logic directly.

Controller should handle Success/Failure of the Model.

Model sends Response back to Controller.



Tools for Model: MongoDB

1. MongoDB is a schema-less NoSQL document database.
2. JSON documents can be stored and the structure can vary because it is not enforced like SQL databases.
3. MongoDB calls its equivalent Tables and Databases, “Collections”. Its records are called “Documents”. Columns of data are called, “Fields”.
4. SQL uses Schemas to define the data structure. MongoDB uses a Document shape to define data structure.

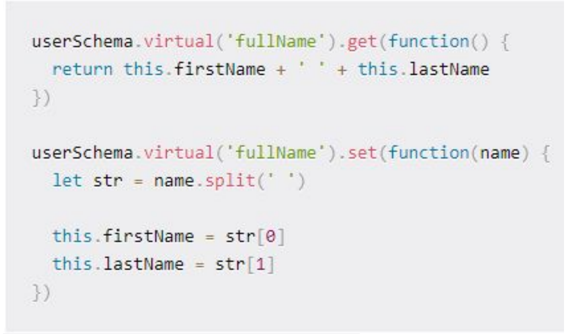
Tools for Model: Mongoose Model

Mongoose provides a Schema and Model interact with MongoDB.

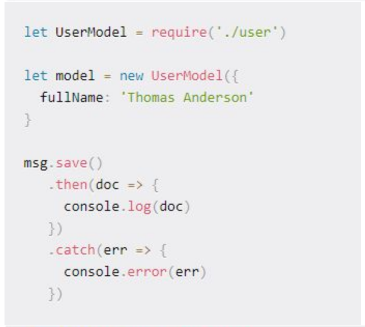
A Mongoose Model provides an interface to the database for creating, querying, updating, deleting records, etc.

This can replace CRUD method to make shorter code.

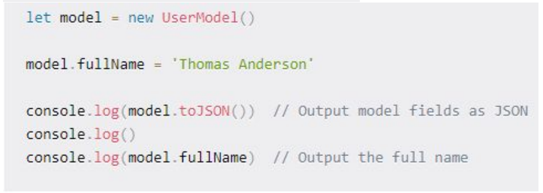
Mongoose Model is a wrapper on the Mongoose Schema.



```
userSchema.virtual('fullName').get(function() {
  return this.firstName + ' ' + this.lastName
})

userSchema.virtual('fullName').set(function(name) {
  let str = name.split(' ')

  this.firstName = str[0]
  this.lastName = str[1]
})
```

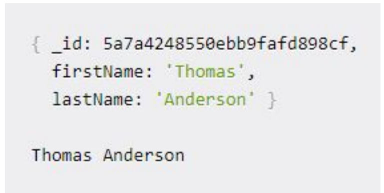


```
let UserModel = require('./user')

let model = new UserModel({
  fullName: 'Thomas Anderson'
}

msg.save()
   .then(doc => {
     console.log(doc)
   })
   .catch(err => {
     console.error(err)
   })
```



```
let model = new UserModel()

model.fullName = 'Thomas Anderson'

console.log(model.toJSON())  // Output model fields as JSON
console.log()
console.log(model.fullName)  // Output the full name
```

Above outputs:

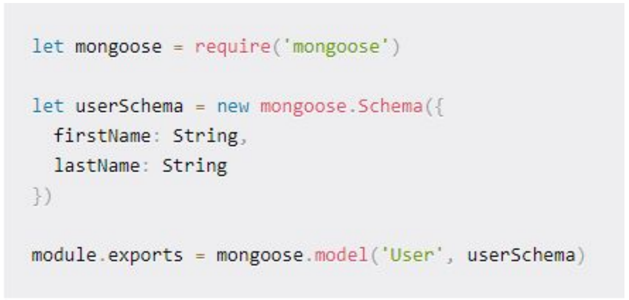


```
{ _id: 5a7a4248550ebb9fafd898cf,
  firstName: 'Thomas',
  lastName: 'Anderson' }

Thomas Anderson
```

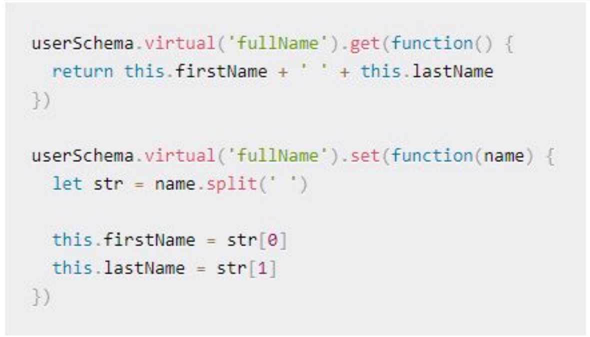
Tools for Model: Mongoose Schema

A Mongoose schema defines the structure of the document, default values, validators, among other things.

Mongoose handles the Schema structure and lets you focus on coding.



```
let mongoose = require('mongoose')

let userSchema = new mongoose.Schema({
  firstName: String,
  lastName: String
})

module.exports = mongoose.model('User', userSchema)
```



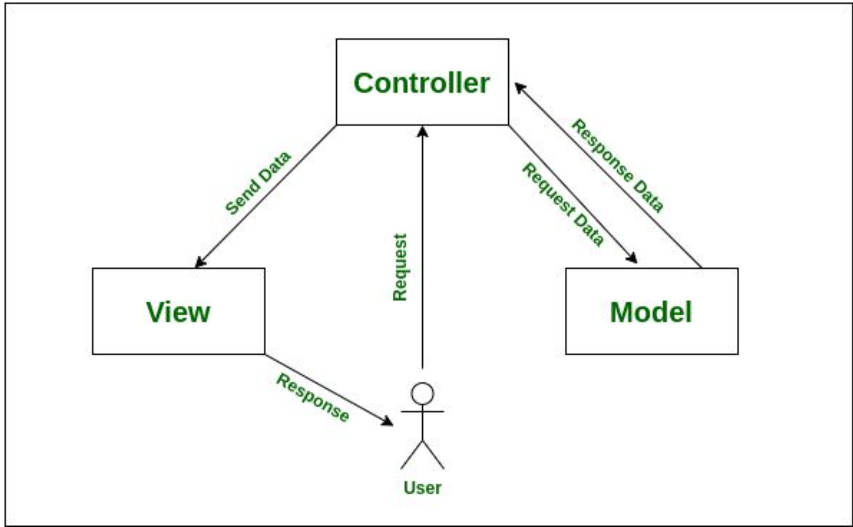
```
userSchema.virtual('fullName').get(function() {
  return this.firstName + ' ' + this.lastName
})

userSchema.virtual('fullName').set(function(name) {
  let str = name.split(' ')

  this.firstName = str[0]
  this.lastName = str[1]
})
```

View

After Controller receives the Response, it will send the data to the View Section.

View will handle the presentation of the data to the user, rendering dynamically.

View and Model should never interact.



Example: Catastrophe

1. User requests pictures of cats.
2. Controller directs Model to bring up pictures of cats
3. Model queries database and responds with cat pictures to Controller
4. Controller sends pictures to View for presentation
5. View Renders pictures back to User.