Model View Controller

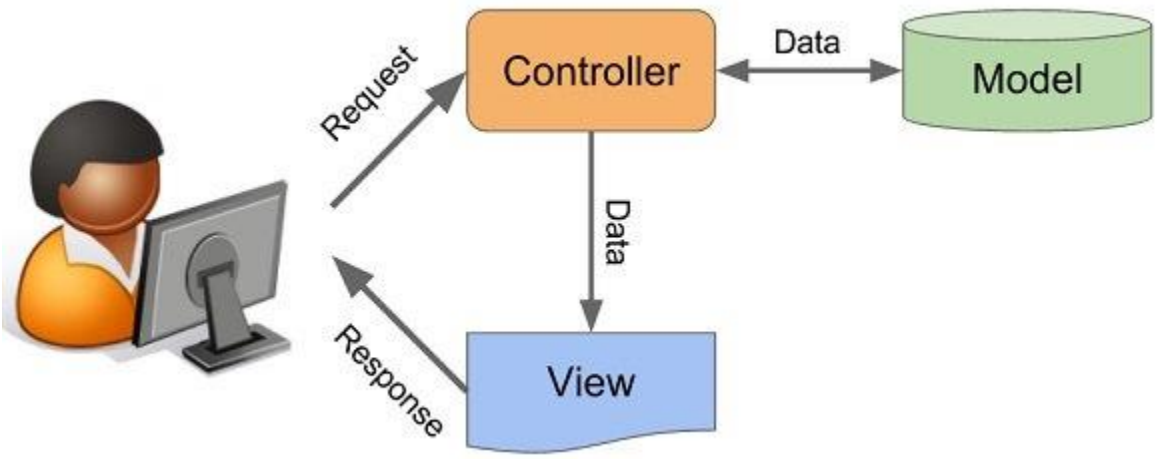

Figure 1

In the Model View Controller Paradigm (MVC) there are 3 separate parts, those 3 parts are a Model, View, and Controller. I've written a small program to visualize the way it works. This program reads numbers from a text file and allows you to set them all to 0 or 1 then saves them to that file. The MVC components are explained below followed by the code my program uses.

Model

The Model is the part of your software that deals with gathering and storing data from and to your database, whether this is SQL database or simply a text file in which you save your values. To create a Model you will build a class that read the data from whatever source you're using and store it in the Model object any changes that need to be made to this data should be done in the Model class. This class will be the only class that saves the data as well.

```
#ifndef MODEL_H
#define MODEL_H
#include <fstream>
#include <vector>
#include <string>

class Model{
private:
    std::vector<int> values;
    std::string file_name;
public:
    Model(const std::string &file_name){
        this->file_name = file_name;
    }
    Model(){ }
    std::string getString(){
        return this->file_name;
    }
    void setOnes(){
        for(auto i = values.begin(); i != values.end(); ++i){
            *i = 1;
        }
    }
    void setZeros(){
        for(auto i = values.begin(); i != values.end(); ++i){
            *i = 0;
        }
    }
    void getNumbers(){
        std::ifstream input;
        input.open(this->file_name);
        std::string current_val;
        while(getline(input,current_val,' ')){
            this->values.push_back(stoi(current_val));
        }
        input.close();
    }
    void saveNumbers(){
        std::ofstream output;
        output.open(this->file_name);
        for(auto i = this->values.begin(); i != this->values.end(); ++i){
            output << *i << ' ';
        }
        output.close();
    }

    std::vector<int> Values(){
        return this->values;
    }
};
#endif
```

View

The View is what you will use to display any information to the user, in this application it is primary using the Console Window. However, typically a view could be anything that display information, a webpage, a UI, or a chat window. In my View class the only functions are PrintValues, which prints all of the data values in the data to the console, and the Print function, which takes a string and prints it to the console.

```cpp
#ifndef VIEW_H
#define VIEW_H
#include <iostream>
#include <vector>
#include "Model.h"


class View{

private:
    Model *model;

public:
    View(Model &model){
        this->model = &model;
    }

    View(){ }

    void PrintValues(){
        std::cout << "Current Data Values: ";
        std::vector<int> values = (*model).Values();
        for(auto i = values.begin(); i != values.end(); ++i){
            std::cout << *i << ' ';
        }
        std::cout << std::endl;
    }

    void Print(std::string a){
        std::cout << a << std::endl;
    }

};

#endif
```

Controller

The Controller is the part of your code that will take all the users input and decide what to do. It can instruct the Model to update its data, but it should never do it directly. In my Controller it has an onLoad function to instruct the View to display the values. And has a getInput which gets the users input and instructs the Model on how to alter its data.

```cpp
#ifndef CONTROLLER_H
#define CONTROLLER_H
#include "Model.h"
#include "View.h"

class Controller{
private:
    Model *model;
    View *view;

public:
    Controller( Model &model, View &view){
        this->model = &model;
        this->view = &view;
    }

    void onLoad(){
        (*this->view).PrintValues();
    }

    void getInput(){
        char input;
        std::cin >> input;
        if(input == '1'){
            (*model).setOnes();
        }else if(input == '0'){
            (*model).setZeros();
        }
    }


};
#endif
```

Main

The main function ties all these components together into one program that can read and write information to and from a file, display that information to the user, and update that file with new values.

```cpp
#include <iostream>
#include "Model.h"
#include "View.h"
#include "Controller.h"

using namespace std;

int main(){
    string file = "nums.txt";
    Model model(file);
    View view(model);
    Controller controller(model,view);
    model.getNumbers();
    controller.onLoad();
    view.Print("What would you like to set the numbers to 1 or 0?");
    controller.getInput();
    view.PrintValues();
    model.saveNumbers();
    view.Print("Bye!");
}
```

Resources:

Figure 1 - https://helloacm.com/model-view-controller-explained-in-c/