# Singe Page Application

# Agenda

**Software design patterns**

**Model View Controller**

# Design pattern

**from Wikipedia:**

**A general reusable solution**
**to a commonly occurring problem**
**within a given context in software design**

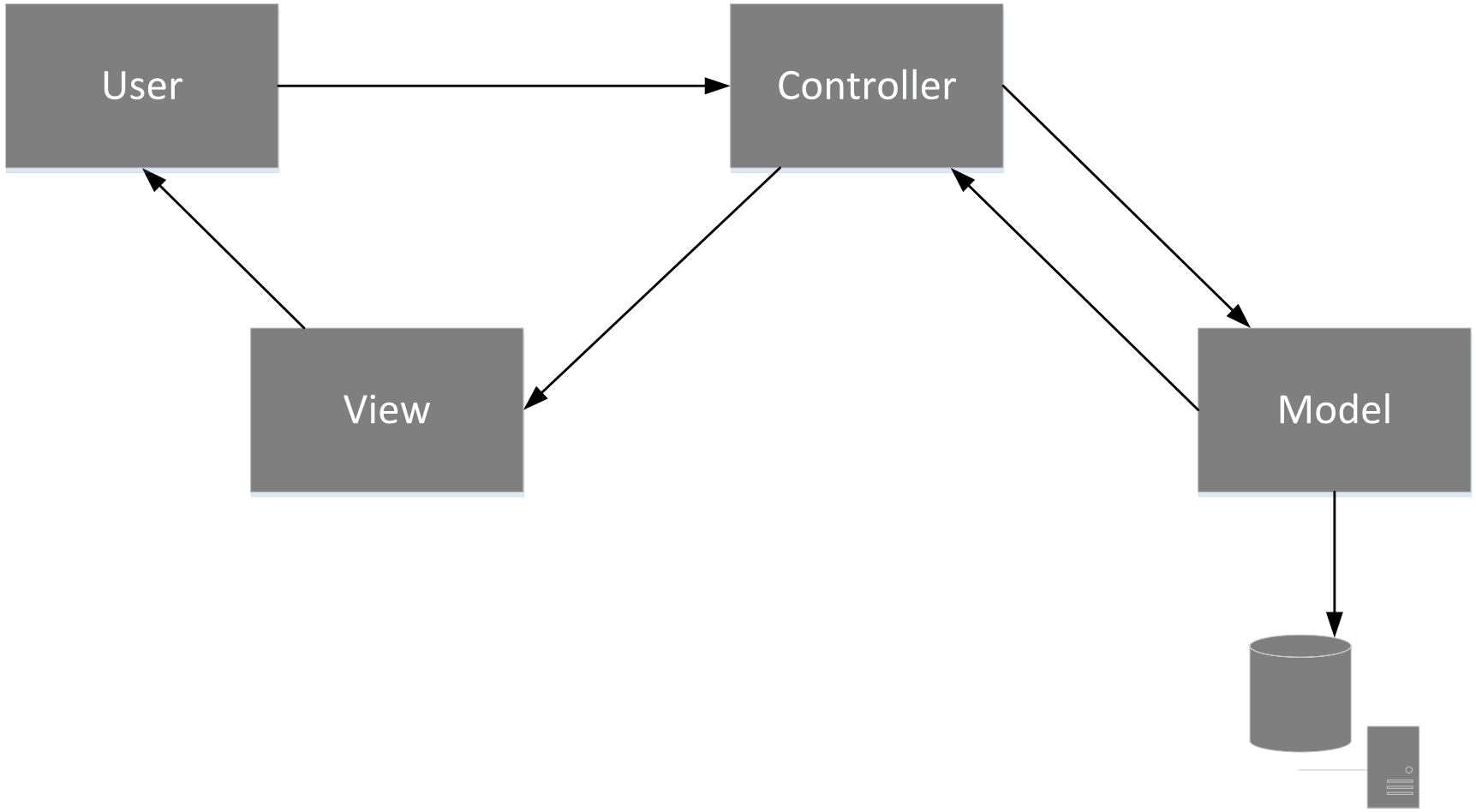
# MVC Design Pattern

**M** **Model**

**V** **View**

**C** **Controller**

# MVC architecture



# MVC App

**Currency Exchange Table**

- **currency.html**
- **model.js**
- **view.js**
- **controller.js**

**Data object**

- **{code: …, rate: …, name: …}**

# HTML head

```
<!DOCTYPE html>
<html>
    <head>
        <title>Currency Exchange Table</title>
        <meta charset="UTF-8">
        <script src="model.js"></script>
        <script src="view.js"></script>
        <script src="controller.js"></script>
    </head>
```

# HTML body

```
    <body onload="controller.updateView()">
        <h1>Currency Exchange Table</h1>
        <!-- list of currency exchange rates -->
        <div id="dataview"></div>
        <hr><!-- form to add new currency -->
        <h3>Add new currency</h3>
        <p>Currency code (e.g. EUR):<br>
        <input type="text" id="code" maxlength="3" ></p>
        <p>Exchange rate (e.g. 4.02):<br>
        <input type="text" id="rate" maxlength="4"></p>
        <p>Currency name (e.g. Euro):<br>
        <input type="text" id="name" maxlength="15"></p>
        <input type="button" value="Add new currency" onclick="view.addCurrency()">
    </body>
</html>
```

# View

```
    clear: function () {
        // remove old data
        var dataView = document.getElementById('dataview');
        while (dataView.firstChild) {
            dataView.removeChild(dataView.firstChild);
        }
    },
    addCurrency: function () {
        var currency = {
                code: document.getElementById('code').value,
                rate: document.getElementById('rate').value,
                name: document.getElementById('name').value
        }
        controller.storeCurrency(currency);
    }
};
```

# Controller

```
var controller = {
        getCurrency: function () {
            return this.model.getData();
        },
        storeCurrency: function (currency) {
            this.model.addData(currency);
            this.updateView();
        },
        delCurrency: function (code) {

        },
        updateView: function () {
            this.data = this.getCurrency();
            this.view.display(this.data);
        },
        model: model,
        view: view,
        data: null
    };
```

# Model

```
var model = {
        getData: function () {return this.storage; },
        addData: function (data) {this.storage.push(data); },
        delData: function (code) {},
        storage: [] // array of objects {code, rate, name}
    };
```

# View

```
var view = {
        display: function (data) {
            // create new data view to be displayed
            var elem, text, i;
            this.clear();
            for (i = 0; i < data.length; i++) {
                elem = document.createElement("h3");
                text = document.createTextNode(
                        data[i].code + "/" + data[i].name + " " + data[i].rate
                        );
                elem.appendChild(text);
                document.getElementById('dataview').appendChild(elem);
            }
        },
```

# To do

# Create drop-down list

1. **Modify the Currency app.**
2. **Instead of typing currency code and name, create a drop-down list with currency codes and names.**
3. **Add to the drop-down list the following items: EUR, USD, GBP, CHF, CZK, HRK, HUF**
4. **Modify the addCurrency method in the view.js to create an object to store in the database.**

# Create HTML table

1. **Modify the app view to display the data in an HTML table as in the example.**
2. **Change the source code of the display method to create table elements:**
   - Table
   - Table header
   - Table rows
   - Table data cells

| Code | Name | Rate |
|---|---|---|
| USD | Dollar | 3.98 |
| EUR | Euro | 4.32 |
| HUF | Forint | 0.14 |
| … | … | … |

# Handle events

1. **Add a new app feature to remove selected currency item from the table.**
2. **After double click on a table row, the row is to be deleted.**
3. **During creating the table, add the event handler (double click) to all table rows executing the controller method:**

   controller.delCurrency(XXX)

   where XXX is the currency code in quotation marks.
4. **Complete the following methods:**
   - Controller.delCurrency()
   - Model.delData()

# Improve app look and feel

**Use the bootstrap framework  to improve look and feel of the app.**

# Store data in local storage

1. **Modify the app model.**
2. **Store the app data in the local storage.**

# Store data in local SQL database

1. **Modify the app model.**
2. **Store the app data in the a Web SQL database.**

# Retrieve data from Web Service

1. **Modify the app to read the current currency exchange rates from a web service.**
2. **Instead of typing the currency exchange rate, use the National Polish Bank Web API (http://api.nbp.pl/)**
3. **Optimise the app to cache external data sources locally (local storage, local database).**