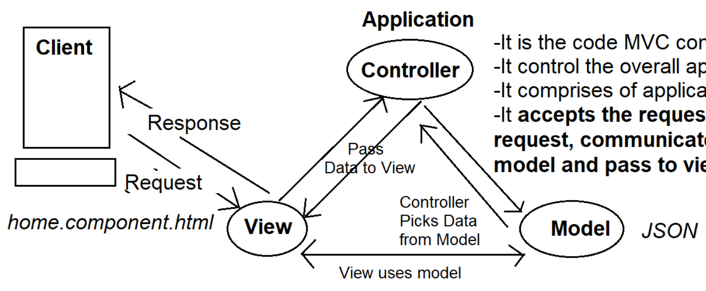
# Data Binding in Angular

- Data binding is a technique used in web application development to bind the data to front-end application UI.
- It is the process of collecting a value, storing in a reference and binding to user interface to render as output.
- Angular uses “MVC and MVVM” frameworks client side.
- These frameworks are responsible for handling data binding in Angular.

## What is MVC?

- MVC is a software architectural pattern.
- Architectural patterns are same like design patterns but have a broader scope.
- They are responsible for both building and controlling the application flow.
- Model-View-Controller
- 1970’s Trygve and formulated with “Small Talk”.
- Code reusability and code separation concerns.
- Various technologies are using MVC framework

| | |
|---|---|
| Java | Spring |
| PHP | Cake PHP, Code Igniter |
| Python | Django, Flask |
| Ruby | Ruby on Rails |
| .NET | ASP.NET MVC |
| JavaScript | Spine, Angular JS |



-It is the code MVC component
-It control the overall applicaiton flow.
-It comprises of application specific logic
-It **accepts the request** from View, it will **process the request, communicates with Model, get data from model and pass to view**.

-Describes the application UI
-Template that renders HTML
-It is under the control of View Engine
-View Engine is responsible for processing and rendering output to UI.
-Angular uses **Ivy** as View Engine

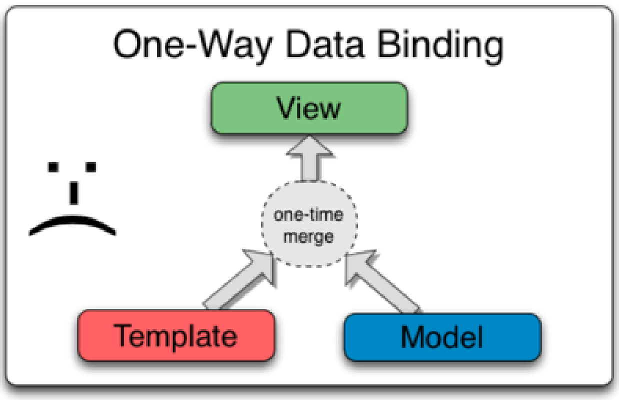
- Represents the data we are working with
- It defines rules for manipulating data.
- It even validates the data.
- It comprises of POJO objects
  [Plain Old JavaScript Object]
- Application data is in model
- Apart from providing POJO, model have significance

**How data is passed from controller to view?**

Angular uses various data binding technique to pass data from controller to view.

- One-way binding
- Two-way binding

**What is One-Way Binding?**



It is the technique where data is passed from controller to view.
It is only one-direction.
It is read-only and forward-only.
It is one-time rendering.
Any change in view will not update back to controller.
You can handle one-way binding in angular by using following approaches

a) Interpolation

b) Binding to attribute [Attribute Binding]
c) Binding to property [Property Binding]

**Interpolation:**

- It uses a data binding expression “{{ }}”
- It is not actually binding to any DOM element
- It is just displaying in the UI as a literal.
- **It allows an expression that evaluates value.**

Ex:

**> ng g c databinding --skipTests**

**Databinding.component.ts**

```
import { Component } from '@angular/core';

@Component({

  selector: 'app-databinding',

  templateUrl: './databinding.component.html',

  styleUrls: ['./databinding.component.css']

})

export class DatabindingComponent {

   product = {

     Name: 'Samsung TV',

     Price: 45000.55,

     InStock: true

   };

   disableButton = true;

}
```

**Databinding.component.html**

```
<div>
   <h2>Product Details</h2>
   <dl>
      <dt>Name</dt>
      <dd>{{product.Name}}</dd>
      <dt>Price</dt>
      <dd>{{product.Price}}</dd>
      <dt>Stock</dt>
      <dd>{{(product.InStock)?"Available":"Out of Stock"}}</dd>
   </dl>
   <button disabled="{{disableButton}}" class="btn btn-
primary">Submit</button>
   {{disableButton}}
</div>
```

**Note: Try to change the value for "disableButton=false" and observe,** the disable attribute of button is not set to false.

## Bind to Attribute & Bind to Property

- What is difference between Attribute and Property?
  **Attribute**
  - HTML elements are defined with attributes statically.
  - We configure tag with attributes.
  Ex: <img src="some.jpg" class="img-thumbnail"> **class and src** are attributes.
  - Attributes are immutable.
  - Their state can't change dynamically.
  **Property**

**-** HTML elements are defined with properties dynamically.
**-** We configure element with properties.
Ex:
```
var pic = new Image();
pic.src = “some.jpg”
pic.className = “img-thumbnail”
```
- Properties are mutable.
- Their state can be changed dynamically.

**Note: Many time HTML element attribute is not having relative property to handle dynamically**.
Ex:
<table height=”300”> height is an attribute but it is not available as property.
You can control height dynamically by using CSS property.

**Binding to Property:**

- Angular can bind any dynamic value to HTML element by using property binding technique.
- The properties are defined in HTML element by using “[]”
- “[]” specifies that element gets value dynamically.
- Properties will not allow interpolation; you have to binding only a dynamic value.

  Ex:
  ```
  public path = “assets/tv.jpg”;
  <img [src]=”{{path}}”>  // invalid
  <img src=”{{path}}”>    // valid
  <img [src]=”path”>      // valid
  ```

**Binding to Attribute:**

- If you have to bind to any attribute then use “[attr.attributeName]”

- You can bind to attribute when the relative property is not available for element.

  Ex:

  **Databinding.component.ts**

  ```
      tableHeight = '100';
      tableWidth = '400';
  ```

  **Databinding.component.html**

  ```
      <table border="1" [attr.height]="tableHeight"
  [width]="tableWidth" >
         <tr>
           <td>Name</td>
         </tr>
      </table>
  ```

Note:

| | |
|---|---|
| [attr.height]="tableHeight" | **Attribute Binding** |
| [width]="tableWidth" | **Property Binding** |

Ex:

**Databinding.component.ts**

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-databinding',
  templateUrl: './databinding.component.html',
  styleUrls: ['./databinding.component.css']
})
export class DatabindingComponent {
```

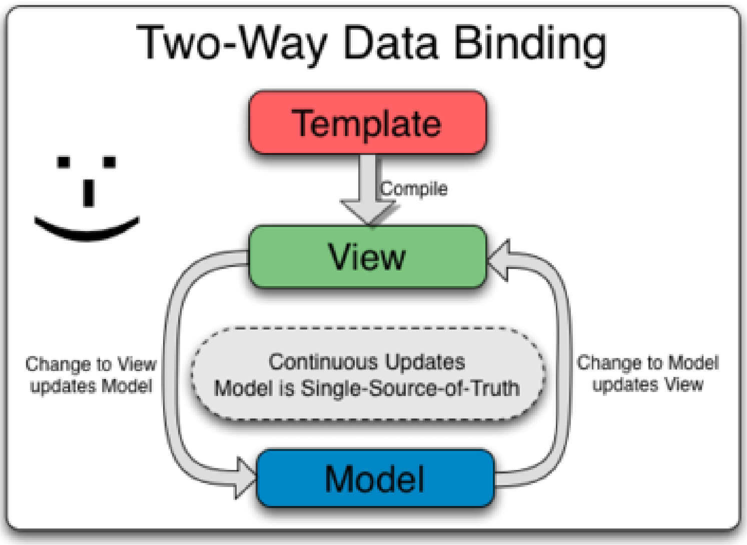
```
  product = {
    Name: 'Samsung TV',
    Price: 45000.55,
    InStock: true
  };
  disableButton = true;
}
```

**Databinding.component.html**

```
<div>
   <h2>Product Details</h2>
   <dl>
      <dt>Name</dt>
      <dd [innerText]="product.Name"></dd>
      <dt>Price</dt>
      <dd [innerHTML]="product.Price"></dd>
      <dt>Stock</dt>
      <dd [innerHTML]="(product.InStock==true)?'Available':'Out of
Stock'"></dd>
   </dl>
   <button [disabled]="disableButton" class="btn btn-
primary">Submit</button>
   {{disableButton}}
</div>
```

## Two Way Data Binding

- The Model data is bound to View.
- Changes in View will update the Model.
- Model handle continuous changes.
- The model changes are update to view and any change in view will update back to model.
- Model is referred as "**Single-Source-of-Truth**"
- It contains information about the value before and after change.

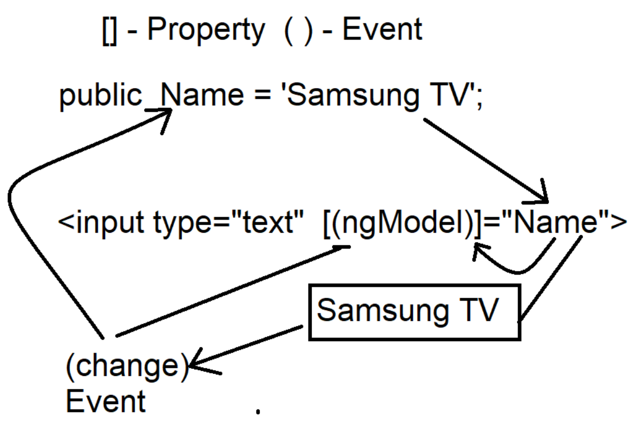

Two-Way Data Binding
Template
Compile
View
Change to View updates Model
Continuous Updates Model is Single-Source-of-Truth
Change to Model updates View
Model

- **Model** is "single source of truth"
- Model create a reference in memory to store the value.
- Controller can access the value and present in View.
- Any change in value from View will be updated back to model reference in memory.
- You can configure a model reference by using "**NgModel**" directive. [ngModel].
- **NgModel** is a member of "**FormsModule**" in **"@angular/forms**" library.
- You have to import FormModule and register in "app.module.ts"
- NgModel uses Property binding and Event binding techniques to handle two-way binding.
- **Property binding** gets the value and binds to View.

- **Event Binding** notifies the changes in value and update back to model.
- **NgModel** property binding is designated with “[]”
- **NgModel** event binding is designated with “()”
- **NgModel** uses the **[value]** property of any element to bind dynamic value.
- **NgModel** uses the **(change)** event to identify the change in value.

  **Syntax:**

  <input type=”text” [(ngModel)]=”referenceName”>
- **The memory reference for any element is created by controller and gives to Model.**


[] - Property ( ) - Event
public Name = 'Samsung TV';
<input type="text" [(ngModel)]="Name">
Samsung TV
(change)
Event

**Ex: Update the model changes immediately to View through controller.**

- **Go to “app.module.ts”**

```
import { FormsModule } from '@angular/forms';

imports: [
   BrowserModule,
   FormsModule
 ],
```
- **Add a new component**

```
> ng g c twowaybinding –skipTests
```
- **Twowaybinding.component.ts**

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-twowaybinding',
  templateUrl: './twowaybinding.component.html',
  styleUrls: ['./twowaybinding.component.css']
})
export class TwowaybindingComponent {
  Name = 'TV';
  Price = 0;
  ShippedTo = 'Hyd';
  InStock = false;
}
```

- **Twowaybinding.component.html**

```
<h2 class="text-primary text-center">Two Way Binding</h2>
<div class="row">
 <div class="col-3">
  <div class="form-group">
    <label>Name</label>
    <div>
        <input type="text" [(ngModel)]="Name" class="form-
control">
    </div>
  </div>
  <div class="form-group">
    <label>Price</label>
    <div>
        <input type="text" [(ngModel)]="Price" class="form-
control">
    </div>
  </div>
  <div class="form-group">
    <label>Shipped To</label>
    <div>
      <select [(ngModel)]="ShippedTo" class="form-control">
        <option>Delhi</option>
        <option>Hyd</option>
      </select>
    </div>
   </div>
   <div class="form-group">
    <label>In Stock</label>
    <div>
      <input [(ngModel)]="InStock" type="checkbox"> Yes
    </div>
   </div>
  </div>
  <div class="col-9">
     <h4>Product Details</h4>
    <dl class="row">
      <dt class="col-sm-3">Name</dt>
      <dd class="col-sm-9">{{Name}}</dd>
      <dt class="col-sm-3">Price</dt>
      <dd class="col-sm-9">{{Price}}</dd>
      <dt class="col-sm-3">Shipped To</dt>
      <dd class="col-sm-9">{{ShippedTo}}</dd>
      <dt class="col-sm-3">Stock</dt>
      <dd class="col-sm-9">{{(InStock)==true?"Available":"Out of
Stock"}}</dd>
    </dl>
  </div>
</div>
```

Ex: **Details are update on Update Button Click**

**Twowaybinding.component.ts**

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-twowaybinding',
  templateUrl: './twowaybinding.component.html',
  styleUrls: ['./twowaybinding.component.css']
})
export class TwowaybindingComponent {
  //Attached to Form Controls
  Name = 'TV';
  Price = 0;
  ShippedTo = 'Hyd';
  InStock = false;
  //New Product Details
  updatedProduct = {
    Name: '',
    Price: 0,
    ShippedTo: '',
    InStock: false
  };
   onUpdateButtonClick(){
     if(this.Name=='') {
       alert('Name Required');
     } else {
      this.updatedProduct = {
        Name: this.Name,
        Price: this.Price,
        InStock: this.InStock,
        ShippedTo: this.ShippedTo
      };
    }
  }
}
```

**Twowaybinding.component.html**

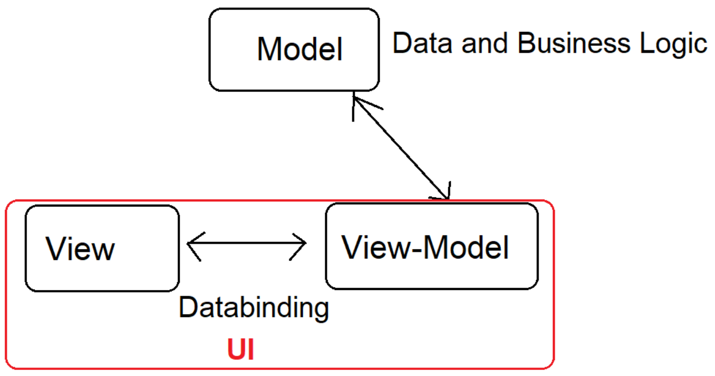
```
<h2 class="text-primary text-center">Two Way Binding</h2>
<div class="row">
 <div class="col-3">
  <div class="form-group">
    <label>Name</label>
    <div>
        <input type="text" [(ngModel)]="Name" class="form-control">
    </div>
  </div>
  <div class="form-group">
    <label>Price</label>
    <div>
        <input type="text" [(ngModel)]="Price" class="form-control">
    </div>
  </div>
  <div class="form-group">
    <label>Shipped To</label>
    <div>
        <select [(ngModel)]="ShippedTo" class="form-control">
            <option>Delhi</option>
            <option>Hyd</option>
        </select>
    </div>
  </div>
  <div class="form-group">
    <label>In Stock</label>
    <div>
        <input [(ngModel)]="InStock" type="checkbox"> Yes
    </div>
  </div>
  <div class="form-group">
    <button (click)="onUpdateButtonClick()" class="btn btn-info btn-
block">Update Details</button>
  </div>
 </div>
 <div class="col-9">
     <h4>Product Details</h4>
    <dl class="row">
```

```
        <dt class="col-sm-3">Name</dt>
        <dd class="col-sm-9">{{updatedProduct.Name}}</dd>
        <dt class="col-sm-3">Price</dt>
        <dd class="col-sm-9">{{updatedProduct.Price}}</dd>
        <dt class="col-sm-3">Shipped To</dt>
        <dd class="col-sm-9">{{updatedProduct.ShippedTo}}</dd>
        <dt class="col-sm-3">Stock</dt>
        <dd class="col-sm-
9">{{(updatedProduct.InStock)==true?"Available":"Out of
Stock"}}</dd>
   </dl>
 </div>
</div>
```

## MVVM

## (Model-View View-Model)

- It is a software architectural pattern.
- View-Model is responsible for handling all types of interactions.
- All configurations and interactions are managed at View level.
- It uses a model reference in View to store and manipulate the data.
- It will not depend on controller.



- Configurations and data are manipulated from view.
- Lot of burden on page.
- Makes the page heavy.
- Slow rendering.
- Regular extensions not possible.
- Hard to test.
- Good for component if it is having a simple straight forward functionality without much extensions required.
- It will reduce the number of requests and can improve page load time.

**Configuring Component for MVVM**

- "ngModel" is a directive used for HTML element to make it dynamic.
- Every element requires a model reference defined by using "#".
  Ex: #Name
- "ngModel" is assigned to reference.
  Ex: "#Name=ngModel"
- Every element requires a form element name, Form can't submit value of any element without name.
  Ex: name="Name"

Syntax:

```
<input type="text"  ngModel  #Name="ngModel"
name="Name">
```

- You can access value of any element in the view by using model reference name.
- Every model reference is provided with several properties
    - value
    - invalid
    - valid
    - pristine
    - dirty
    - touched
    - untouched
    - errors [object]  etc.
- "value" property returns the value of element.
  Syntax:
  {{Name.value}}

Ex:

- Create a new component with inline template and styles
  ```
  > ng g c  mvvm-demo  --inlineTemplate --inlineStyle --skipTests
  ```
- This will add only

**"mvvm-demo.component.ts"**

```
import { Component, OnInit } from '@angular/core';

@Component({

 selector: 'app-mvvm-demo',

 template: `

 <div>

  <div class="row">

    <div class="col-3">

      <h2>Register</h2>

      <div class="form-group">
```

```
            <label>Name</label>
            <div>
               <input ngModel #Name="ngModel" name="Name"
type="text" class="form-control">
            </div>
         </div>
         <div class="form-group">
            <label>Price</label>
            <div>
               <input ngModel #Price="ngModel" name="Price"
type="text" class="form-control">
            </div>
         </div>
         <div class="form-group">
            <label>Shipped To</label>
            <div>
               <select ngModel #ShippedTo="ngModel"
name="ShippedTo" class="form-control">
                  <option>Delhi</option>
                  <option>Hyd</option>
               </select>
            </div>
         </div>
         <div class="form-group">
            <label>In Stock</label>
```

```
                <div>
                    <input ngModel #InStock="ngModel" name="InStock"
type="checkbox"> Yes
                </div>
            </div>
        </div>
        <div class="col-9">
            <h2>Details</h2>
            <table class="table table-hover">
                <colgroup span="1" style="font-weight: bold; background-
color: bisque;"></colgroup>
               <tbody>
                   <tr>
                       <td>Name</td>
                       <td>{{Name.value}}</td>
                   </tr>
                   <tr>
                       <td>Price</td>
                       <td>{{Price.value}}</td>
                   </tr>
                   <tr>
                       <td>Shipped To</td>
                       <td>{{ShippedTo.value}}</td>
                   </tr>
                <tr>
                   <td>Stock</td>
                   <td>{{(InStock.value==true)?"Available":"Out of
Stock"}}</td>
                </tr>
             </tbody>
          </table>
       </div>
    </div>
</div>
  `,
  styles: []
})
export class MvvmDemoComponent {


}
```

**Directives of Angular**

- Directive is a function.
- Angular directive function is responsible for converting the static DOM element into dynamic DOM element.
- Directive makes HTML more declarative.
- Directives have different types of functionalities, they can be used as
    - Element
    - Attribute

    - Class
    - Comment
- **Directives are used as Elements to return markup.**
  <ng-form> </ng-form>
  <router-outlet> </router-outlet>
- **Directives are used as Attributes to extend markup.**
  <input ngModel>
- **Directives are used as classes to make HTML more interactive and responsive.**
  <style>
  **.**ng-valid {
  }
  </style>
  <span class=”ng-valid”> </span>
- **Directive are used as comments to target legacy browsers**
  <!-- ngModel:Name -->
- **Angular directives are classified into 3 groups**
    - Component Directives
    - Structural Directives
    - Attribute Directives