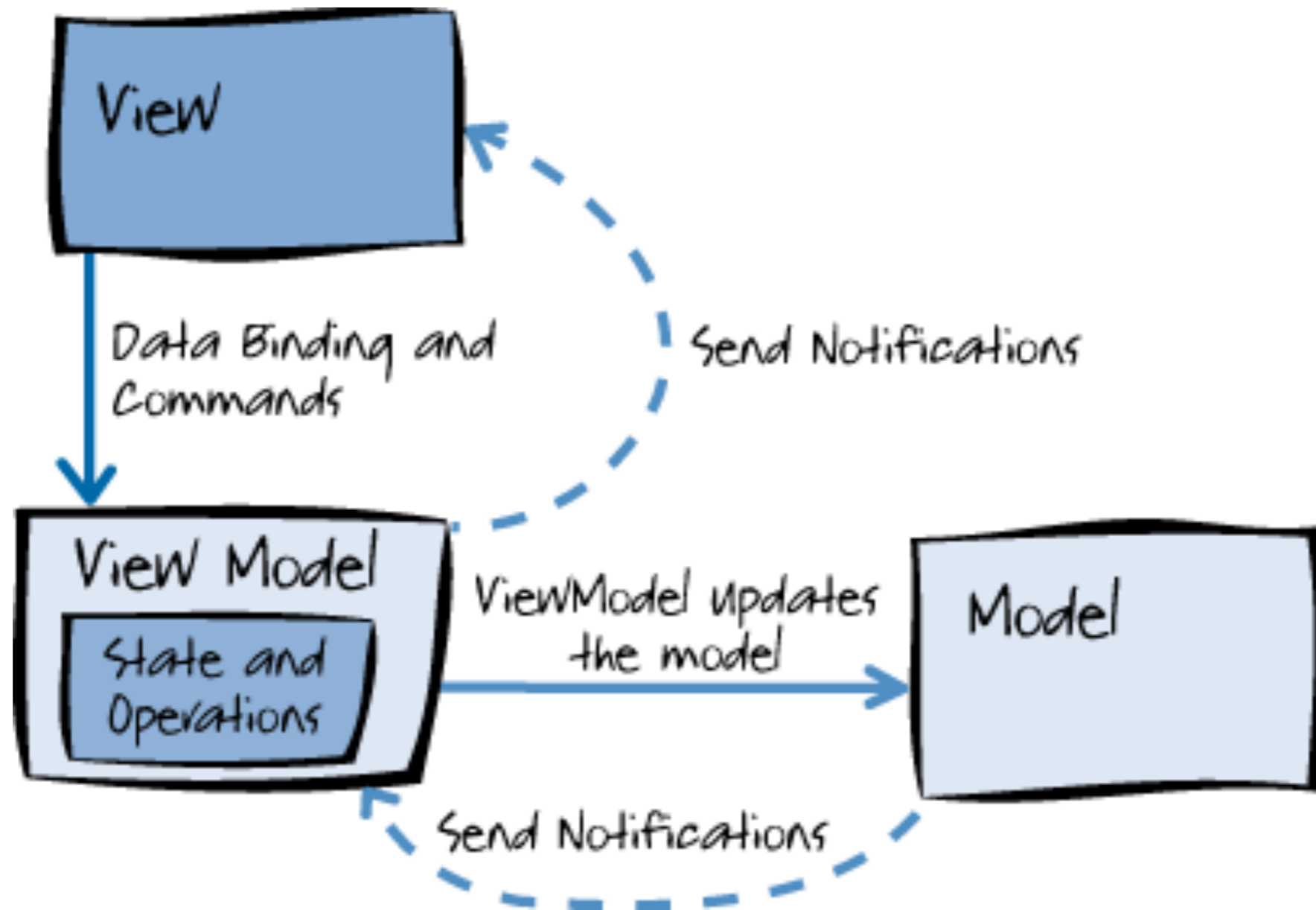


MVVM in Aurelia

MVVM



Benefits of MVVM

1. Separation of Skills: This enables a separation of responsibilities on teams have a designers and programmers
2. Views are agnostic from the code that runs behind them, enabling the same view-models to be reused across multiple views
3. No duplicated code to update views - rely on databinding to keep view and view-model in sync.
4. Since view-model is separated from view view-model classes can be tested independently
5. The Model can be shared across multiple view-models, and can be used to centralise resource access (e.g. Remote API access).

View

Add a Candidate

First Name

Last Name

Office

Add

```
<template>
  <form submit.trigger="addCandidate()" class="ui form stacked segment">
    <h3 class="ui dividing header"> Add a Candidate </h3>
    <div class="field">
      <label>First Name </label> <input value.bind="firstName">
    </div>
    <div class="field">
      <label>Last Name </label> <input value.bind="lastName">
    </div>
    <div class="field">
      <label>Office </label> <input value.bind="office">
    </div>
    <button class="ui blue submit button">Add</button>
  </form>
</template>
```

View-Model

```
import {inject} from 'aurelia-framework';
import DonationService from '../services/donation-service';

@Inject(DonationService)
export class Candidate {

  firstName = '';
  lastName = '';
  office = '';

  constructor(ds) {
    this.donationService = ds;
  }

  addCandidate() {
    this.donationService.addCandidate(this.firstName, this.lastName, this.office);
  }
}
```

Model

```
import {inject} from 'aurelia-framework';
import Fixtures from './fixtures';

@Inject(Fixtures)
export default class DonationService {

  donations = [];
  methods = [];
  candidates = [];

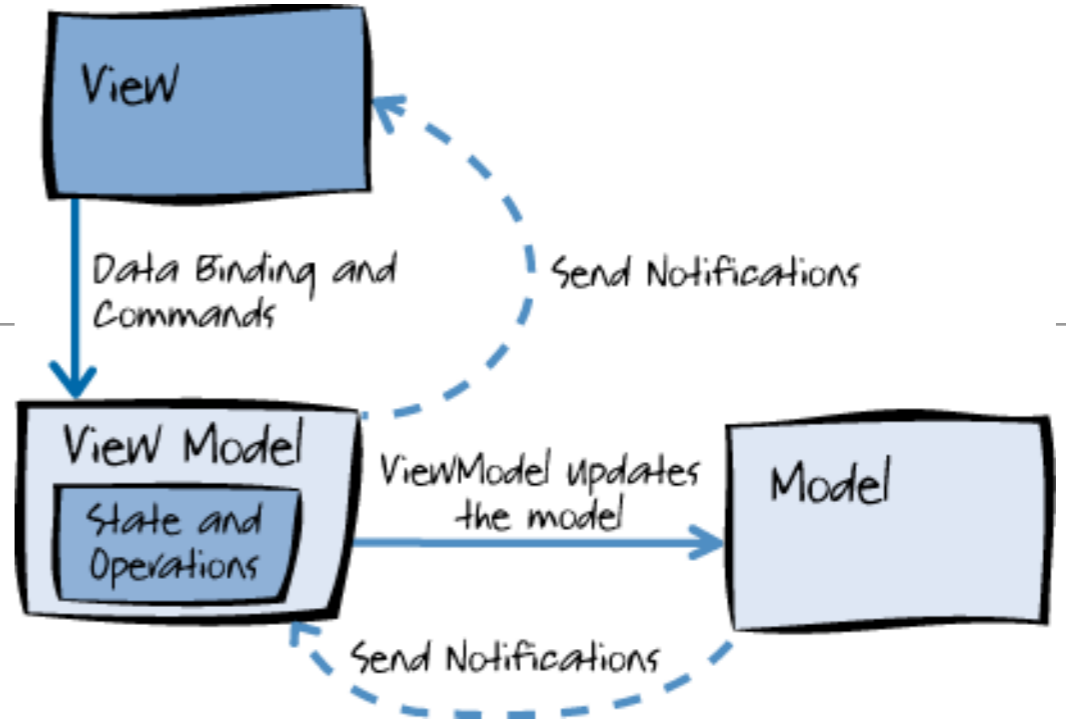
  constructor(data) {
    this.donations = data.donations;
    this.candidates = data.candidates;
    this.methods = data.methods;
  }

  donate(amount, method, candidate) {
    const donation = {
      amount: amount,
      method: method,
      candidate: candidate
    };
    this.donations.push(donation);
    console.log(amount + ' donated to ' + candidate.firstName + ' ' + candidate.lastName + ': ' + method);
  }

  addCandidate(firstName, lastName, office) {
    const candidate = {
      firstName: firstName,
      lastName: lastName,
      office: office
    };
    this.candidates.push(candidate);
  }
}
```

candidate.html

candidate.js



donation-service.js



```

<template>
  <form submit.trigger="addCandidate()" class="ui form stacked segment">
    <h3 class="ui dividing header"> Add a Candidate </h3>
    <div class="field">
      <label>First Name </label> <input value.bind="firstName">
    </div>
    <div class="field">
      <label>Last Name </label> <input value.bind="lastName">
    </div>
    <div class="field">
      <label>Office </label> <input value.bind="office">
    </div>
    <button class="ui blue submit button">Add</button>
  </form>
</template>

```

candidate.html

candidate.js

```

import {inject} from 'aurelia-framework';
import Fixtures from './fixtures';

@Inject(Fixtures)
export default class DonationService {

  donations = [];
  methods = [];
  candidates = [];

  constructor(data) {
    this.donations = data.donations;
    this.candidates = data.candidates;
    this.methods = data.methods;
  }

  donate(amount, method, candidate) {
    const donation = {
      amount: amount,
      method: method,
      candidate: candidate
    };
    this.donations.push(donation);
    console.log(amount + ' donated to ' + candidate.firstName + ' '
      + candidate.lastName + ': ' + method);
  }

  addCandidate(firstName, lastName, office) {
    const candidate = {
      firstName: firstName,
      lastName: lastName,
      office: office
    };
    this.candidates.push(candidate);
  }
}

```

donation-
service.js

```

import {inject} from 'aurelia-framework';
import DonationService from '../services/donation-service';

@Inject(DonationService)
export class Candidate {

  firstName = '';
  lastName = '';
  office = '';

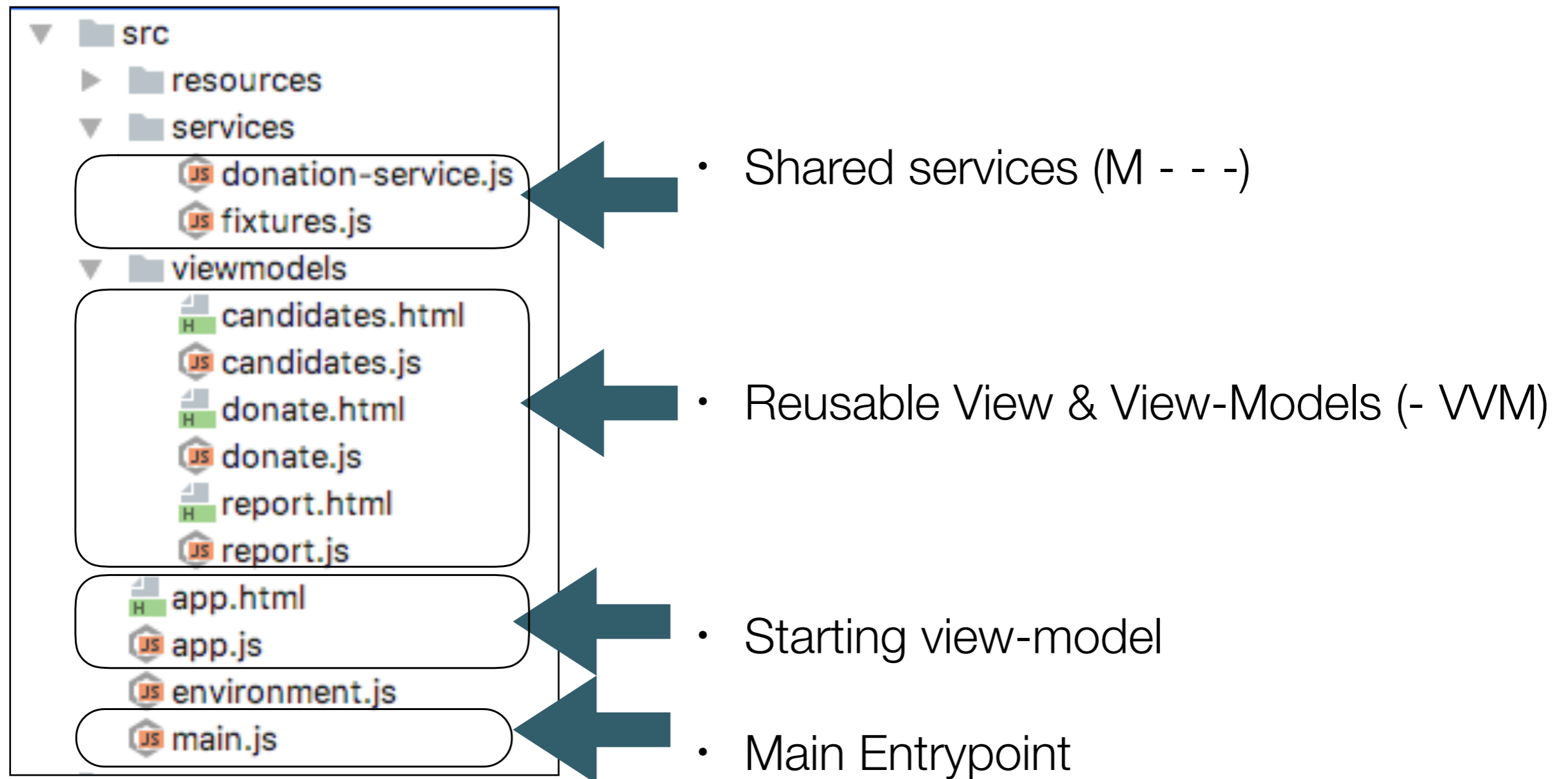
  constructor(ds) {
    this.donationService = ds;
  }

  addCandidate() {
    this.donationService.addCandidate(this.firstName,
      this.lastName, this.office);
  }
}

```



Project Structure



DI in DonationService

- Fixtures contains sample initial data for test purposes

```
import {inject} from 'aurelia-framework';  
import Fixtures from './fixtures';
```

```
@inject(Fixtures)
```

```
export default class DonationService {
```

```
  donations = [];  
  methods = [];  
  candidates = [];
```

```
  constructor(data) {  
    this.donations = data.donations;  
    this.candidates = data.candidates;  
    this.methods = data.methods;  
  }
```

```
  donate(amount, method, candidate) {  
    const donation = {  
      amount: amount,  
      method: method,  
      candidate: candidate  
    };  
    this.donations.push(donation);  
    console.log(amount + ' donated to ' + candidate.firstName + ' '  
      + candidate.lastName + ': ' + method);  
  }
```

```
  addCandidate(firstName, lastName, office) {  
    const candidate = {  
      firstName: firstName,  
      lastName: lastName,  
      office: office  
    };  
    this.candidates.push(candidate);  
  }  
}
```



```
export default class Fixtures {  
  methods = ['Cash', 'PayPal'];  
  candidates = [  
    {  
      firstName: 'Lisa',  
      lastName: 'Simpson'  
    },  
    {  
      firstName: 'Bart',  
      lastName: 'Simpson'  
    }  
  ];  
  donations = [  
    {  
      amount: 23,  
      method: 'cash',  
      candidate: this.candidates[0]  
    },  
    {  
      amount: 212,  
      method: 'paypal',  
      candidate: this.candidates[1]  
    }  
  ];  
}
```

Report View-Model

```
import {inject} from 'aurelia-framework';
import DonationService from '../services/donation-service';

@Inject(DonationService)
export class Report {

  donations = [];

  constructor(ds) {
    this.donationService = ds;
    this.donations = this.donationService.donations;
  }
}
```

```
<template>

<article class="ui stacked segment">
  <h3 class="ui dividing header"> Donations to Date </h3>
  <table class="ui celled table segment">
    <thead>
      <tr>
        <th>Amount</th>
        <th>Method donated</th>
        <th>Candidate</th>
      </tr>
    </thead>
    <tbody>
      <tr repeat.for="donation of donations">
        <td> ${donation.amount}</td>
        <td> ${donation.method}</td>
        <td> ${donation.candidate.lastName}, ${donation.candidate.firstName}</td>
      </tr>
    </tbody>
  </table>
</article>

</template>
```

- donations and array in Report view-model, bound to view

Initialised to reference

donations array in DonationService

View now bound to DonationService donations

Donate View-Model

```
import {inject} from 'aurelia-framework';
import DonationService from '../services/donation-service';

@inject(DonationService)
export class Donate {

  amount = 0;

  methods = [];
  selectedMethod = '';

  candidates = [];
  selectedCandidate = '';

  constructor(ds) {
    this.donationService = ds;
    this.methods = ds.methods;
    this.selectedMethod = this.methods[0];
    this.candidates = ds.candidates;
    this.selectedCandidate = this.candidates[0];
  }

  makeDonation() {
    this.donationService.donate(this.amount,
                                this.selectedMethod,
                                this.selectedCandidate);
  }
}
```

```
<template>

  <form submit.trigger="makeDonation()" class="ui form stacked segment">

    <h3 class='ui dividing header'> Make a Donation </h3>
    <div class="grouped inline fields">
      <div class="field">
        <label>Amount</label> <input type="number" value.bind="amount">
      </div>
    </div>

    <h4 class="ui dividing header"> Select Method </h4>
    <div class="grouped inline fields">

      <div class="field" repeat.for="method of methods">
        <div class="ui radio checkbox">
          <input type="radio" model.bind="method" checked.bind="selectedMethod">
          <label>${method}</label>
        </div>
      </div>
      <div class="ui circular label"> ${selectedMethod} </label>
    </div>

    <h4 class="ui dividing header"> Select Candidate </h4>
    <div class="grouped inline fields">
      <div class="field" repeat.for="candidate of candidates">
        <div class="ui radio checkbox">
          <input type="radio" model.bind="candidate" checked.bind="selectedCandidate">
          <label>${candidate.lastName}, ${candidate.firstName}</label>
        </div>
      </div>
      <div class="ui circular label"> ${selectedCandidate.firstName} ${selectedCandidate.lastName} </label>
    </div>

    <button class="ui blue submit button">Donate</button>

  </form>
</template>
```

- makeDonation updates DonationService model
- Databinding ensures report is updated

Make a Donation

Amount

Select Method

- Cash
 PayPal

Cash

Select Candidate

- Simpson, Lisa
 Simpson, Bart
 simpson, donald

Bart Simpson

Donate

Donations to Date

Amount	Method donated	Candidate
23	cash	Simpson, Lisa
212	paypal	Simpson, Bart
2	Cash	Simpson, Bart

Add a Candidate

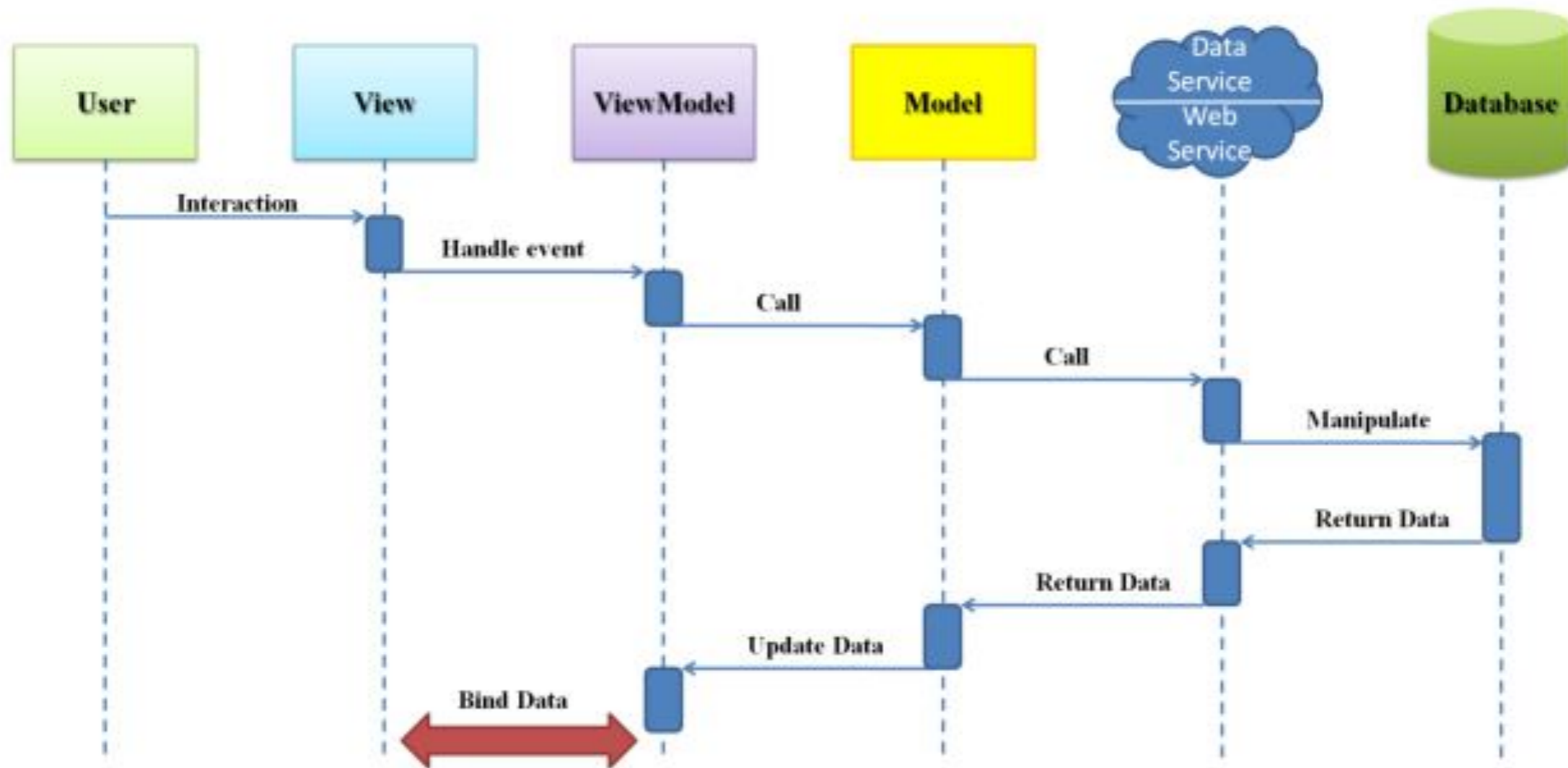
First Name

Last Name

Office

Add

MVVM Sequence Diagram



Sequence Diagram of MVVM