

# Create a CRUD Web Page Labs

Perform these labs on your own computer using Visual Studio 2022 or later to ensure you understand the lessons presented in the corresponding videos and lectures.

## Lab 1: Create Website Using ASP.NET Core

Open Visual Studio 2022 and click on the **Create a New Project** button as shown in Figure 1.

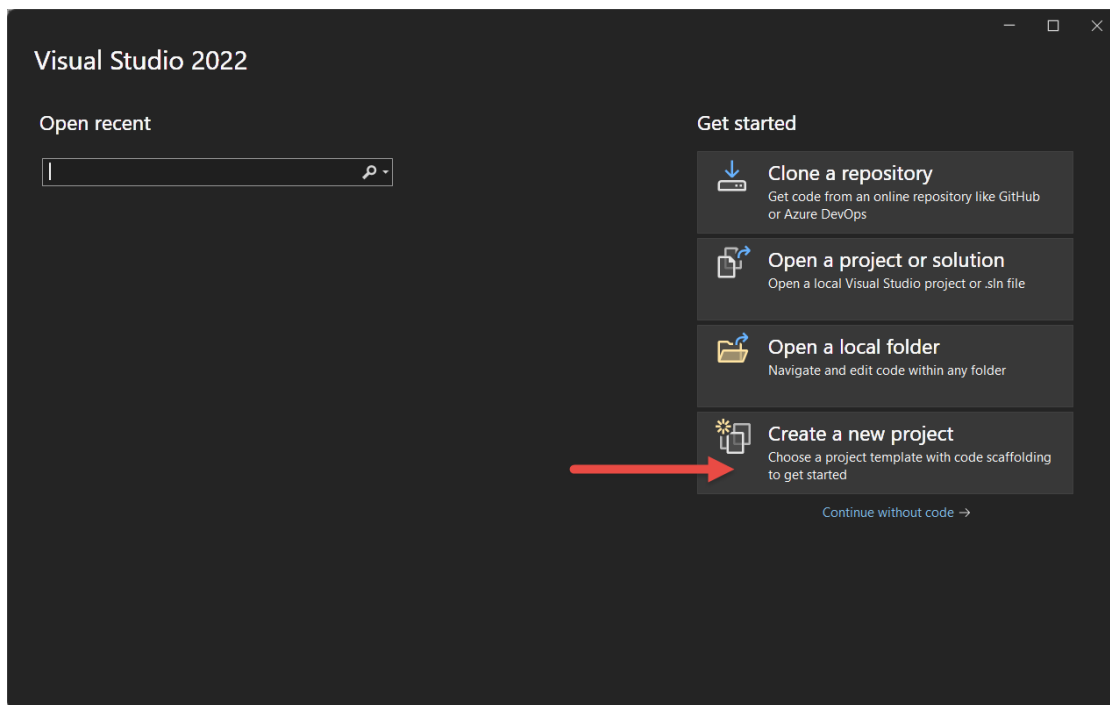


Figure 1: Create a new project in Visual Studio

## Create a New Project Screen

In the search box at the top right (Figure 2), type in **MVC**.

In the list below the box locate **ASP.NET Core Web App (Model-View-Controller)** and click on it.

Click the **Next** button to continue to the next screen.

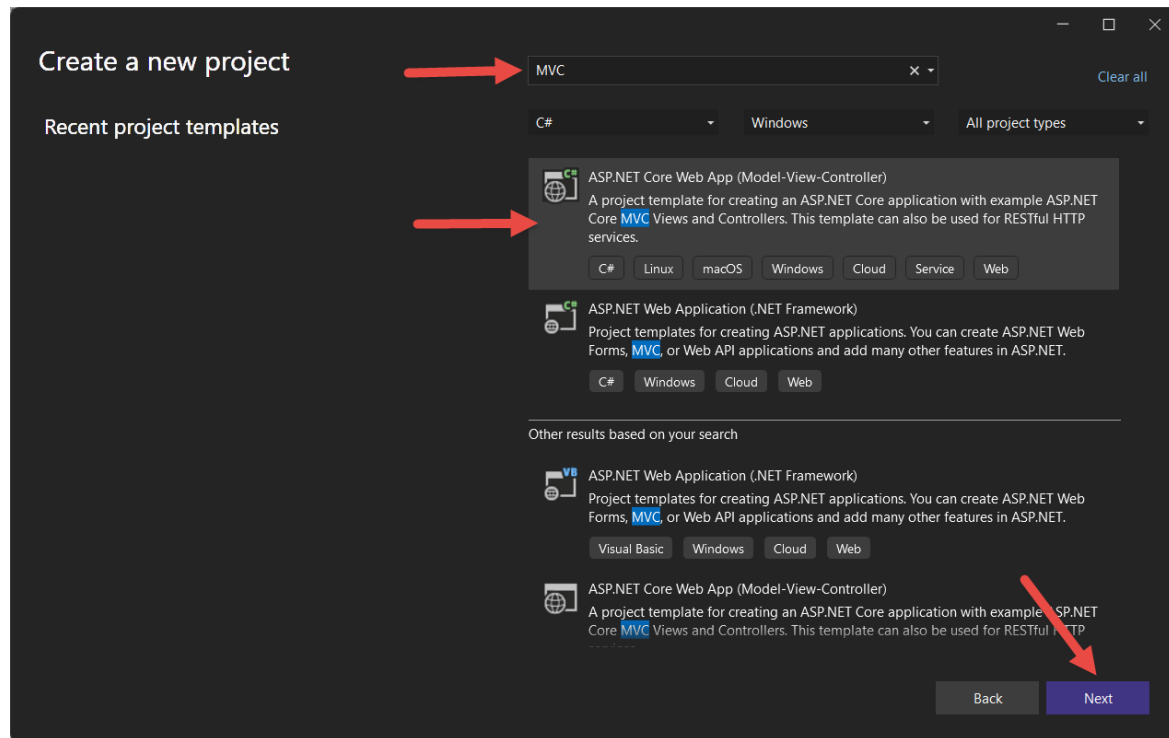


Figure 2: Create a new project screen

## Configure Your New Project Screen

As shown in Figure 3 below, fill in the **Project Name** with **PeopleMaintenance**.

Fill in the **Location** with the folder location where you typically place your projects.

Check the **Place solution and project in the same directory** option.

Click the **Next** button.

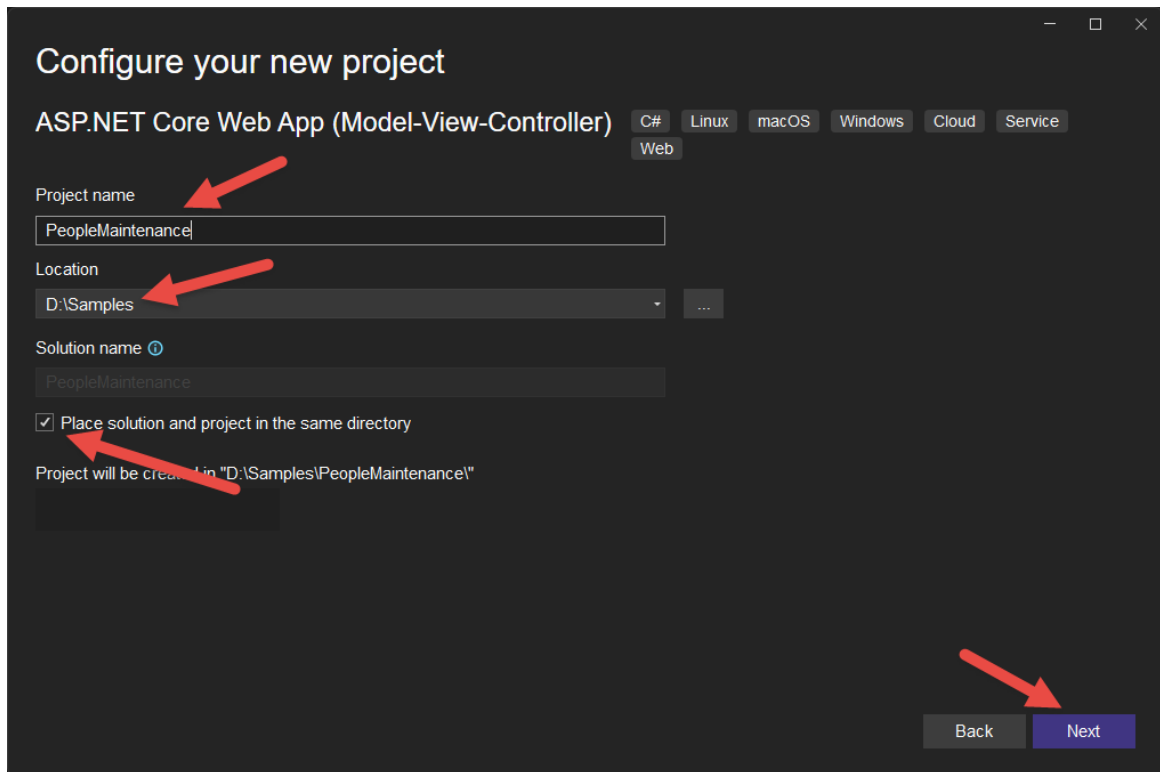


Figure 3: Configure your new project screen

## Additional Information Screen

On the Additional Information Screen (shown in Figure 4), change the **Target Framework** to **.NET 6.0 (Long-term support)** or **.NET 8.0 (Long-term support)**.

Set the Authentication Type to **None**.

Uncheck **Configure for HTTPS**.

Uncheck **Do not use top-level statements**.

Click the **Create** button to build the new MVC project.

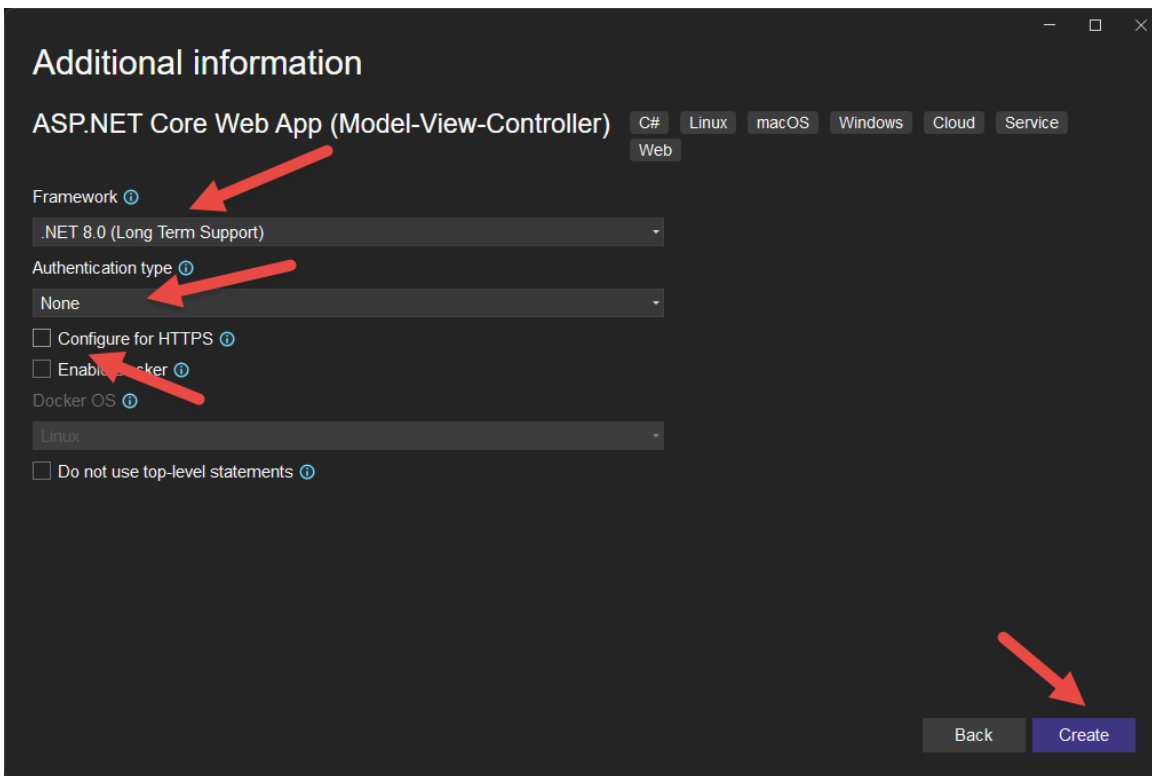


Figure 4: Additional information screen

## Try It Out

Run the MVC application by selecting **Debug | Start Debugging** from the Visual Studio menu (or press F5) and you should see a home page that looks like Figure 5.

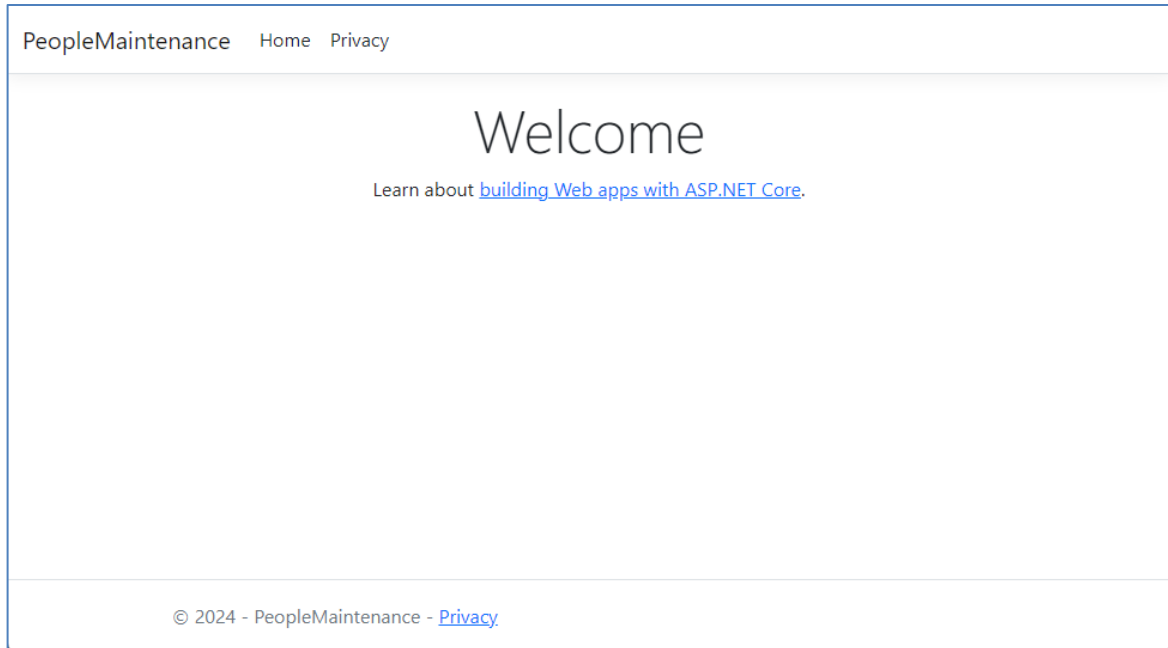


Figure 5: The default MVC application web page

## Lab 2: Add Bootstrap and Site Styles

Right mouse-click on the root folder and add a new folder named **styles**.

Right mouse-click on the **styles** folder and add a new file named **site.css**.

Put the following CSS styles into this new file.

```
.infoMessage {
    font-weight: bold;
    padding: 0.5em;
}

.errorMessage {
    font-weight: bold;
    background-color: red;
    color: white;
    padding: 0.5em;
}
```

Open the **index.html** file and make the <head> tag look like the following.

```
<head>
  <title>People Maintenance</title>
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/
css/bootstrap.min.css"
  integrity="sha384-
xOolHFLEh07PJGoPkLv1IbcEPTNtaed2xpHsD9ESMhqIYd0nLMwNLD69
Npy4HI+N" crossorigin="anonymous">
  <link rel="stylesheet" type="text/css"
href="/styles/site.css">
</head>
```

## Lab 3: Add the HTML for the CRUD Page

Replace the current `<body>` tag and add some `<script>` tags with the following code.

```

<body>
  <div class="container">
    <main role="main" class="pb-3">
      <h1>People Maintenance</h1>

      <div class="row mt-3 mb-3">
        <div class="col">
          <span id="message" class="infoMessage d-none">
          </span>
          <span id="error" class="errorMessage d-none">
          </span>
        </div>
      </div>

      <div id="list" class="d-none">
        <div class="row">
          <div class="col">
            <button type="button" class="btn btn-primary
mt-2 mb-2" onclick="peopleController.add();">
              Add Person
            </button>
          </div>
        </div>
        <table id="people" class="table table-bordered
table-striped table-collapsed">
          <thead>
            <tr>
              <th>Actions</th>
              <th>Person ID</th>
              <th>First Name</th>
              <th>Last Name</th>
              <th>Email Address</th>
              <th>Start Date</th>
            </tr>
          </thead>
          <tbody>
          </tbody>
        </table>
      </div>

      <form id="detail" class="d-none">
        <div class="form-group">
          <label for="personId">Person ID</label>
          <input type="text" id="personId"
name="personId" readonly="readonly" class="form-control"
/>
        </div>
      </form>
    </main>
  </div>

```

```

        <div class="form-group">
            <label for="firstName">First Name</label>
            <input type="text" id="firstName"
name="firstName" class="form-control" />
        </div>
        <div class="form-group">
            <label for="lastName">Last Name</label>
            <input type="text" id="lastName"
name="lastName" class="form-control" />
        </div>
        <div class="form-group">
            <label for="emailAddress">Email
Address</label>
            <input type="email" id="emailAddress"
name="emailAddress" class="form-control" />
        </div>
        <div class="form-group">
            <label for="startDate">Start Date</label>
            <input type="date" id="startDate"
name="startDate" class="form-control" />
        </div>
        <div class="form-group">
            <button type="button" class="btn btn-primary"
id="saveButton" onclick="peopleController.save();">
                Save
            </button>
            <button type="button" class="btn btn-info"
id="cancelButton" onclick="peopleController.cancel();">
                Cancel
            </button>
        </div>
    </form>
</main>
</div>
</body>

<script id="listTpl" type="text/html">
    {{#list}}
    <tr>
        <td>
            <button type="button" class="btn btn-primary"
onclick="peopleController.getEntity({{personId}});">
                Edit
            </button>
            &nbsp;
            <button type="button" class="btn btn-danger"
onclick="peopleController.deleteEntity({{personId}});">

```



```
        Delete
      </button>
    </td>
    <td>{{personId}}</td>
    <td>{{firstName}}</td>
    <td>{{lastName}}</td>
    <td>{{emailAddress}}</td>
    <td>{{startDateForJSON}}</td>
  </tr>
  {{/list}}
</script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/mustache.js/
4.1.0/mustache.min.js"></script>
<script src="/scripts/common.js"></script>
<script src="/scripts/people.js"></script>

<script>
  'use strict';

  window.onload = function () {
    peopleController.get();
  }
</script>
```

## Lab 4: Add a People Script File

Right mouse-click on the root folder and add a new folder named **scripts**.

Right mouse-click on the **scripts** folder and add a new file named **people.js**.

Into this file place the following code.

```
// *****
// Global Variable for people Page
// *****
let peopleController = (function () {
  // *****
  // Private Variables
  // *****
  let vm = {
    "list": [],
    "entity": {},
    "data": null,
    "mode": "list",
    "apiUrl": "https://localhost:7153/api/people",
    "ok": false,
    "status": 0,
    "statusText": "",
    "message": "",
    "lastResponse": {},
    "startDateFormatted": function () {
      if (this.startDate) {
        return new
Date(this.startDate).toLocaleDateString('en-us');
      }
      else {
        return "n/a";
      }
    }
  }

  // *****
  // Private Functions
  // *****
  function get() {
    vm.mode = "list";
    vm.entity = {};

    fetch(vm.apiUrl)
      .then(response => processResponseObject(response))
      .then(data => {
        // Check if response was successful
        if (vm.ok) {
          // Assign data to view model's list property
          vm.list = data.data;
          // Use template to build HTML table
          buildList(vm);
        }
        else {

```

```
        common.displayError(common.handleError(vm));
    }
    })
    .catch(error =>
common.displayError(common.handleNetworkError(error)));
}

function getEntity(id) {
    vm.mode = "edit";
    vm.list = [];

    fetch(`${vm.apiUrl}/${id}`)
        .then(response => processResponseObject(response))
        .then(data => {
            // Check if response was successful
            if (vm.ok) {
                // Assign data to view model's entity property
                vm.entity = data.data;

                // Display entity
                setInput();

                // Unhide Save/Cancel buttons
                common.displayButtons();

                // Unhide detail area
                common.displayDetail();
            }
            else {
                common.displayError(common.handleError(vm));
            }
        })
        .catch(error =>
common.displayError(common.handleNetworkError(error)));
}

function insertEntity() {
    setEntityFromInput();

    let options = {
        method: 'POST',
        body: JSON.stringify(vm.entity),
        headers: {
            'Content-Type': 'application/json'
        }
    };
};
```

```
    fetch(vm.apiUrl, options)
      .then(response => processResponseObject(response))
      .then(data => {
        if (vm.ok) {
          // Hide buttons while 'success message' is
displayed
          common.hideButtons();

          // Display a success message
          common.displayMessage(vm.message);

          // Redisplay entity returned from Web API
          setInput();

          // Clear messages
          clearMessages(true);
        }
        else {
          common.displayError(common.handleError(vm));
        }
      })
      .catch(error =>
common.displayError(common.handleNetworkError(error)));
  }

function updateEntity() {
  setEntityFromInput();

  let options = {
    method: 'PUT',
    body: JSON.stringify(vm.entity),
    headers: {
      'Content-Type': 'application/json'
    }
  };

  fetch(`${vm.apiUrl}/${vm.entity.personId}`, options)
    .then(response => processResponseObject(response))
    .then(data => {
      if (vm.ok) {
        // Hide buttons while 'success message' is
displayed
        common.hideButtons();

        // Display a success message
        common.displayMessage(vm.message);
      }
    })
    .catch(error =>
common.displayError(common.handleNetworkError(error)));
}
```

```
// Redisplay entity returned from Web API
setInput();

// Clear messages
clearMessages(true);
}
else {
  common.displayError(common.handleError(vm));
}
})
.catch(error =>
common.displayError(common.handleNetworkError(error)));
}

function deleteEntity(id) {
  if (confirm(`Delete Person ${id}?`)) {
    let options = {
      method: 'DELETE'
    };

    fetch(`${vm.apiUrl}/${id}`, options)
      .then(response =>
processResponseObject(response))
      .then(data => {
        // Check if response was successful
        if (vm.ok) {
          // Display success message
          common.displayMessage("Person Deleted.");

          // Redisplay all data
          get();

          // Clear messages
          clearMessages(false);
        }
        else {
          common.displayError(common.handleError(vm));
        }
      })
      .catch(error =>
common.displayError(common.handleNetworkError(error)));
    }
  }

  async function processResponseObject(response) {
    let resp = await
common.processResponseObject(response);
```

```
        vm.ok = resp.ok;
        vm.status = resp.status;
        vm.statusText = resp.statusText;
        vm.message = resp.message;
        vm.data = resp.data;
        vm.lastResponse = response;

        return vm;
    }

    function buildList(vm) {
        // Get HTML template from <script> tag
        let template =
document.getElementById("listTpl").text;

        // Call Mustache passing in the template and
        // the object with the collection of data
        let html = Mustache.render(template, vm);

        // Insert the rendered HTML into the DOM

document.getElementById("people").getElementsByTagName('
tbody')[0].innerHTML = html;

        // Display HTML table and hide <form> area
        common.displayList();
    }

    function setInput() {
        // Ensure start date is valid for input fields
        vm.entity.startDate =
common.fixDate(vm.entity.startDate);

        document.getElementById("personId").value =
vm.entity.personId;
        document.getElementById("firstName").value =
vm.entity.firstName;
        document.getElementById("lastName").value =
vm.entity.lastName;
        document.getElementById("emailAddress").value =
vm.entity.emailAddress;
        document.getElementById("startDate").value =
vm.entity.startDate;
    }

    function setEntityFromInput() {
```

```
        vm.entity.personId =
Number.parseInt(document.getElementById("personId").value);
        vm.entity.firstName =
document.getElementById("firstName").value;
        vm.entity.lastName =
document.getElementById("lastName").value;
        vm.entity.emailAddress =
document.getElementById("emailAddress").value;
        vm.entity.startDate =
document.getElementById("startDate").value;
    }

    function cancel() {
        // Hide detail area
        document.getElementById("detail").classList.add('d-
none');
        // Clear any messages
        common.displayMessage("");
        // Display all data
        get();
    }

    function clearInput() {
        vm.entity.personId = 0;
        vm.entity.firstName = "";
        vm.entity.lastName = "";
        vm.entity.emailAddress = "";
        vm.entity.startDate = new Date(2024, 0, 1);

        setInput();
    }

    function add() {
        vm.mode = "add";

        // Display empty entity
        clearInput();

        // Make sure buttons are displayed
        common.displayButtons();

        // Unhide detail area
        common.displayDetail();
    }

    function save() {
```

```
// Determine method to call based on the mode
property
  if (vm.mode === "add") {
    insertEntity();
  } else if (vm.mode === "edit") {
    updateEntity();
  }
}

function clearMessages(redisplayList) {
  setTimeout(() => {
    if (redisplayList) {
      get();
    }

    // Clear message
    common.displayMessage("");
  }, 2000);
}
// *****
// Public Functions
// *****
return {
  "get": get,
  "getEntity": getEntity,
  "cancel": cancel,
  "add": add,
  "save": save,
  "deleteEntity": deleteEntity
};
})();
```

## Lab 5: Add a Common Script File

Right mouse-click on the **scripts** folder and add a new file named **common.js**.

Into this file place the following code.



```
// *****
// Global Variable for Common Functions
// *****
let common = (function () {
  // *****
  // Private Functions
  // *****
  async function processResponseObject(response) {
    let text = "";
    let body = {};
    // Create custom response object
    let ret = {
      "ok": response.ok,
      "status": response.status,
      "statusText": response.statusText,
      "url": response.url,
      "message": "",
      "data": {}
    };

    try {
      // Get text first
      text = await response.text();
      // Attempt to convert to JSON
      body = JSON.parse(text);
      // Set JSON body into data property
      ret.data = body.data;
      ret.message = body.message;
      if (ret.data == null) {
        ret.data = text;
      }
    }
    catch {
      // Set text into data property
      ret.data = text;
    }

    return ret;
  }

  function handleNetworkError(error) {
    let msg = "A network error has occurred. Check the
    apiUrl property to ensure it is set correctly.";

    console.error(error + " - " + msg);

    return msg;
  }
}
```

```
}

function handleError(vm) {
  let msg = "";

  switch (vm.status) {
    case 400:
      msg = JSON.stringify(vm.data);
      break;
    case 404:
      if (vm.data) {
        msg = vm.data;
      }
      else {
        msg = `${vm.statusText} - ${vm.apiUrl}`;
      }
      break;
    case 500:
      msg = JSON.parse(vm.data).message;
      break;
    default:
      msg = JSON.stringify(vm.data);
      break;
  }

  if (msg) {
    console.error(msg);
  }

  return msg;
}

function fixDate(dt) {
  let ret = new Date();

  // Ensure a null was not passed in
  if (dt) {
    ret = new Date(dt);
  }

  let year = ret.getFullYear();
  let month = (ret.getMonth() +
1).toString().padStart(2, '0');
  let day = (ret.getDate()).toString().padStart(2,
'0');
  ret = `${year}-${month}-${day}`;
}
```

```
        return ret;
    }

    /*******
    /** Generic Display Methods
    /*******
    function displayList() {
        document.getElementById("list").classList.remove('d-
none');
        document.getElementById("detail").classList.add('d-
none');
    }

    function displayMessage(msg) {
        if (msg) {
            document.getElementById("message").textContent =
msg;

document.getElementById("message").classList.remove('d-
none');
        }
        else {

document.getElementById("message").classList.add('d-
none');
        }
    }

    function displayError(msg) {
        if (msg) {
            document.getElementById("error").textContent =
msg;

document.getElementById("error").classList.remove('d-
none');
        }
        else {
            document.getElementById("error").classList.add('d-
none');
        }
    }

    function displayButtons() {

document.getElementById("saveButton").classList.remove('
d-none');
```

```
document.getElementById("cancelButton").classList.remove('d-none');
}

function displayDetail() {
    document.getElementById("list").classList.add('d-none');

    document.getElementById("detail").classList.remove('d-none');
}

function hideButtons() {

document.getElementById("saveButton").classList.add('d-none');

document.getElementById("cancelButton").classList.add('d-none');
}

// *****
// Public Functions
// *****
return {
    "processResponseObject": processResponseObject,
    "handleNetworkError": handleNetworkError,
    "handleError": handleError,
    "fixDate": fixDate,
    "displayList": displayList,
    "displayMessage": displayMessage,
    "displayError": displayError,
    "displayButtons": displayButtons,
    "displayDetail": displayDetail,
    "hideButtons": hideButtons
};
})();
```

## Try It Out

Run the MVC application and you should see your data appear.