Use the Fetch API to POST, PUT, and DELETE Data 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: Add Modify API Endpoints

Open the **program.cs** file in the Minimal Web API server and add the following endpoints.

```
app.MapPost("/api/people", (Person entity) =>
{
  IResult ret;
  PersonRepository repo = new();

  Person? resp = repo.Insert(entity);

  if (resp != null) {
    ret =
Results.Created($"/api/people/{resp?.PersonId}", new
Response() {
      Status = 201,
      StatusText = "Created",
      Message = "New Person Added.",
      Data = resp
    });
  }
  else {
    ret = Results.BadRequest(new Response() {
      Status = 400,
      StatusText = "BadRequest",
      Message = "Unable to Add New Person.",
      Data = null
    });
  }

  return ret;
});

app.MapPut("/api/people/{id}", (int id, Person entity)
=>
{
  IResult ret;
  PersonRepository repo = new();
  Person? resp;

  // Check if Person Exists
  resp = repo.Get(id);
  if (resp == null) {
    ret = Results.NotFound(new Response() {
      Status = 404,
      StatusText = "NotFound",
      Message = $"Can't find Person with id='{id}' to
update.",
      Data = null
    });
  }
```

```
  else {
    // Attempt to Update Person
    resp = repo.Update(entity);
    ret = Results.Ok(new Response() {
      Status = 200,
      StatusText = "OK",
      Message = $"Person Updated.",
      Data = resp
    });
  }

  return ret;
});

app.MapDelete("/api/people/{id}", (int id) =>
{
  IResult ret;
  PersonRepository repo = new();

  if (repo.Delete(repo.Get(id) ?? new())) {
    ret = Results.NoContent();
  }
  else {
    ret = Results.NotFound(new Response() {
      Status = 404,
      StatusText = "NotFound",
      Message = $"Can't find Person with id='{id}' to
delete.",
      Data = null
    });
  }

  return ret;
});
```

Lab 2: POST Data Using the Fetch API

Open the **Views\Home\Index.cshtml** file and just **after** the tag that wraps up the personId input field, add some more input fields.

```
<div>
  <label for="firstName">First Name</label>
  <input type="text" id="firstName" value="Sheila" />
</div>
<div>
  <label for="lastName">Last Name</label>
  <input type="text" id="lastName" value="Cleverly" />
</div>
<div>
  <label for="emailAddress">Email Address</label>
  <input type="email" id="emailAddress"
value="sheilac@netinc.com" />
</div>
<div>
  <label for="startDate">Start Date</label>
  <input type="date" id="startDate" value="2001-10-10"
/>
</div>
```

Just **after** the *<button onclick="getRow();">Get a Row</button>*, add a new button.

```
<button onclick="insertRow();">Insert a Row</button>
```

Just after the **getRow**() function, add a new function to gather all the input field data and create a JSON object that can be used to either insert or update a row of data.

```
function createPersonObjectFromInput() {
  return {
    "personId":
Number.parseInt(document.getElementById("personId").valu
e),
    "firstName":
document.getElementById("firstName").value,
    "lastName":
document.getElementById("lastName").value,
    "emailAddress":
document.getElementById("emailAddress").value,
    "startDate":
document.getElementById("startDate").value
  };
}
```

Add a new function named insertRow() to perform the insert.

```
function insertRow() {
  let entity = createPersonObjectFromInput();

  fetch(`${SERVER_URI}/people`, {
    method: 'POST',
    body: JSON.stringify(entity),
    headers: {
      "content-type": 'application/json'
    }
  })
    .then(response => processResponseObject(response))
    .then(response => {
      console.log(response);
      // Extract new Person ID, place into input field
      document.getElementById("personId").value =
response.data.personId;
    })
    .catch(error => console.error(error));
}
```

Try It Out

Save all your changes. Run the MVC application.

Click on the **Insert a Row** button.

Bring up the Console Window and you should see a single JSON object.

Record the value of the **personId** that has been put into the Person ID input field. You will need this number for the next two labs.

Lab 3: PUT Data Using the Fetch API

Just **after** the *<button onclick="insertRow();">Insert a Row</button>*, add a new button.

```
<button onclick="updateRow();">Update a Row</button>
```

Just after the **insertRow**() function, add a new function named updateRow() to perform the update.

```
function updateRow() {
  let entity = createPersonObjectFromInput();

  fetch(`${SERVER_URI}/people/${entity.personId}`, {
    method: 'PUT',
    body: JSON.stringify(entity),
    headers: {
      "content-type": 'application/json'
    }
  })
    .then(response => processResponseObject(response))
    .then(response => console.log(response))
    .catch(error => console.error(error));
}
```

Try It Out

Save all your changes. Run the MVC application.

Fill in the **personId** value you recorded from the last lab into the Person ID input field.

Change the **First Name** input field to "Ginger".

Change the **Last Name** input field to "Hollywood".

Click on the **Update a Row** button.

Bring up the Console Window and you should see a single JSON object with the changes you made.

Lab 4: DELETE Data Using the Fetch API

Just **after** the *<button onclick="updateRow();">Update a Row</button>*, add a new button.

```
<button onclick="deleteRow();">Delete a Row</button>
```

Just after the **updateRow**() function, add a new function named deleteRow() to perform the update. Notice the use of **response** instead of response.json(). Because the response object sent back from a 204 no content does not have a data property to convert to a JSON object, you do not need to call the .json() method.

```
function deleteRow() {
  let id = document.getElementById("personId").value;

  fetch(`${SERVER_URI}/people/${id}`, {
    method: 'DELETE'
  })
    .then(response => processResponseObject(response))
    .then(response => console.log(response))
    .catch(error => console.error(error));
}
```

Try It Out

Save all your changes. Run the MVC application.

Fill in the **personId** value you recorded from the last lab into the Person ID input field.

Click on the **Delete a Row** button.

Bring up the Console Window and you should see a 204 is returned.