Validate Data Lab

Lab 1: Add Validation Helper

NOTE:	Unlike in Web API with MVC Controllers, the minimal API does not support automatic model binding and validation.

Right mouse-click on the project and add a new folder named **ValidationClasses**.

Right mouse-click on the ValidationClasses folder and add a new class named **ValidationHelper**.

```
using System.ComponentModel.DataAnnotations;

namespace AdvWorksAPI.ValidationClasses;

public static class ValidationHelper
{
  public static Dictionary<string, string[]>
Validate<T>(T? entity)
  {
    Dictionary<string, string[]> ret = new();

    if (entity != null) {
      // Create instance of ValidationContext object
      ValidationContext context = new(entity,
serviceProvider: null, items: null);
      List<ValidationResult> results = new();

      // Call TryValidateObject() method
      if (!Validator.TryValidateObject(entity, context,
results, true)) {
        // Get validation results
        foreach (ValidationResult item in results) {
          string propName = string.Empty;
          if (item.MemberNames.Any()) {
            propName = ((string[])item.MemberNames)[0];
          }
          // Create new dictionary object
          ret.Add(propName, new string[] {
            item.ErrorMessage ?? "Unknown Validation
Error"
          });
        }
      }
    }

    return ret;
  }
}
```

Lab 2: Validate the Customer Class

Open the **Customer.cs** file and add some validation data annotations. Add the items in bold to the appropriate places within the Customer class.

```
[Table("Customer", Schema = "SalesLT")]
public partial class Customer
{
  // CONSTRUCTOR CODE HERE

  [Key]
  [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
  [Required()]
  public int CustomerID { get; set; }

  [Required()]
  public bool NameStyle { get; set; }

  [StringLength(8, MinimumLength = 2, ErrorMessage =
"{0} must be between {2} and {1} characters long.")]
  public string? Title { get; set; }

  [Required()]
  [StringLength(50, MinimumLength = 2, ErrorMessage =
"{0} must be between {2} and {1} characters long.")]
  public string FirstName { get; set; }

  [StringLength(50, MinimumLength = 0, ErrorMessage =
"{0} must be between {2} and {1} characters long.")]
  public string? MiddleName { get; set; }

  [Required()]
[StringLength(50, MinimumLength = 2, ErrorMessage = "{0}
must be between {2} and {1} characters long.")]
  public string LastName { get; set; }

  // REST OF THE CODE HERE
}
```

Lab 3: Modify the Insert() Method

Open the **CustomerRouter.cs** file and modify the Insert() method.

Add a call to the ValidateEntity() method

```
protected virtual IResult Insert(Customer entity,
IRepository<Customer> repo)
{
  IResult ret;
  Dictionary<string, string[]> msgs;

  // Serialize entity
  SerializeEntity<Customer>(entity);

  try {
    if (entity != null) {
      // Validate the Entity object
      msgs =
ValidationHelper.Validate<Customer>(entity);
      if (msgs == null || msgs.Count == 0) {
        // Attempt to update the database
        entity = repo.Insert(entity);

        // Return a '201 Created' with the new entity
        ret =
Results.Created($"/{UrlFragment}/{entity.CustomerID}",
entity);
      }
      else {
        ret = Results.ValidationProblem(msgs);
      }
    }

    // REST OF THE CODE HERE

  }

  return ret;
}
```

Try it Out

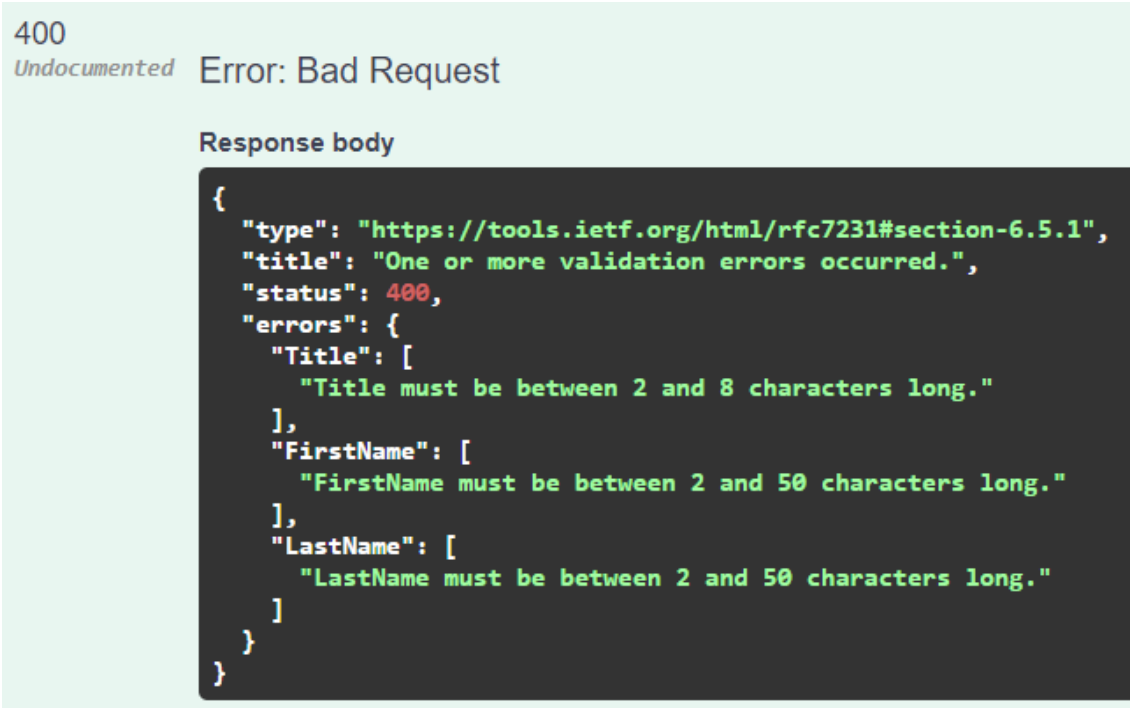
Run the application and click on the **POST /api/Customer** button

Add the following input into the **Request Body**

```
{
  "NameStyle": true,
  "Title": "M",
  "FirstName": "A",
  "MiddleName": "B",
  "LastName": "S",
  "Suffix": "",
  "CompanyName": "Smythe Motors",
  "SalesPerson": "Gene",
  "EmailAddress": "Amy.Smythe@smythemotors.com",
  "Phone": "(977) 333-9938",
  "PasswordHash": "123bbdeic3332",
  "PasswordSalt": "235asdf"
}
```

Click the **Execute** button.

You should see the following errors appear.



Lab 4: Modify the Update Method

Locate the Update() method and make the same change

```
protected virtual IResult Update(int id, Customer
entity, IRepository<Customer> repo)
{
  IResult ret;
  Dictionary<string, string[]> msgs;

  // Serialize entity
  SerializeEntity<Customer>(entity);

  try {
    if (entity != null) {
      // Validate the Entity object
      msgs =
ValidationHelper.Validate<Customer>(entity);
      if (msgs == null || msgs.Count == 0) {
        // Attempt to locate the data to update
        Customer? current = repo.Get(id);
        if (current != null) {
          // Combine changes into current record
          entity = repo.SetValues(current, entity);

          // Attempt to update the database
          current = repo.Update(current);

          // Pass back a '200 Ok'
          ret = Results.Ok(current);
        }
        else {
          InfoMessage = $"Can't find Customer Id '{id}'
to update.";
          // Did not find data, return '404 Not Found'
          ret = Results.NotFound(InfoMessage);
          // Log an informational message
          _Logger.LogInformation("{InfoMessage}",
InfoMessage);
        }
      }
      else {
        ret = Results.ValidationProblem(msgs);
      }
    }

    // REST OF THE CODE HERE

  }
```

```
  return ret;
}
```

Try it Out

Run the application and click on the **PUT /api/Customer** button

Put the value 3333 into the **id** field.

Add the following input into the **Request Body**

```
{
  "CustomerID" : 3333,
  "NameStyle": true,
  "Title": "M",
  "FirstName": "A",
  "MiddleName": "B",
  "LastName": "S",
  "Suffix": "",
  "CompanyName": "Smythe Motors",
  "SalesPerson": "Gene",
  "EmailAddress": "Amy.Smythe@smythemotors.com",
  "Phone": "(977) 333-9938",
  "PasswordHash": "123bbdeic3332",
  "PasswordSalt": "235asdf"
}
```

Click the **Execute** button

You should see the same validation errors as you saw in the last lab.