From Peter McTyiner – winter 2017 – not used

**“…AddForm” and model validation error**

In a scenario that uses an “…AddForm” view model class, we must write code to handle a model validation error. Why?

Well, think about the typical “add new” use case:

1. Create an “…AddForm” object, and **optionally** configure it with starter data
2. It is sent to the view – the view is expecting a model of type “…AddForm”
3. The browser user enters data, some of it invalid, and submits; the data must conform to the packaging format that’s defined by the “…Add” view model class
4. In the POST method, model state is valid will be false

Before changes, the code returns the just-submitted data package (an “…Add” object) to the view. However, the view is expecting an “…AddForm” object. What happens?

Boom. Error.

Can we fix this problem? Yes. Easy. How?

Map the data in the submitted “…Add” object to a new “…AddForm” object. That way, the view is happy, and we pass on the data – good and bad – to the view, so that it can render error message information (and preserve the user-entered data).

Old code, before changes:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | // Validate the input  if (!ModelState.IsValid)  {    // Uh oh, problem with the data, show the form again, with the data    return View(newItem);  }    // Process the input  var addedItem = m.CustomerAdd(newItem);    if (addedItem == null)  {    // Uh oh, some problem adding, show the form again    return View(newItem);  }  else  // etc. |

New code, after changes:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | // Validate the input  if (!ModelState.IsValid)  {    // Uh oh, problem with the data, show the form again, with the data    return View(m.mapper.Map<CustomerAddForm>(newItem));  }    // Process the input  var addedItem = m.CustomerAdd(newItem);    if (addedItem == null)  {    // Uh oh, some problem adding, show the form again    return View(m.mapper.Map<CustomerAddForm>(newItem));  }  else  // etc. |

**What if we normally configure the “…AddForm” object with starter data?**

Assuming that we don’t want to lose either the starter data or the user-entered data, we simply change the sequence of tasks. And, we cannot use AutoMapper. Here’s the new sequence:

1. Create an “…AddForm” object, and configure it with starter data
2. Then, with many lines of code, set the value of each property in the “…AddForm” object with the values that are in the matching properties of the user-submitted object

New code, after changes:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | // Validate the input  if (!ModelState.IsValid)  {    // Uh oh, problem with the data, show the form again, with the data    var form = new CustomerAddForm();    // Add starter data    form.Age = 23;    form.City = "Toronto";    // Add data from the user-submitted object    form.Foo = newItem.Foo;    form.Bar = newItem.Bar;    // etc. for each property in the newItem object    return View(form);  }    // Process the input  var addedItem = m.CustomerAdd(newItem);    if (addedItem == null)  {    // Uh oh, some problem adding, show the form again    var form = new CustomerAddForm();    // Add starter data    form.Age = 23;    form.City = "Toronto";    // Add data from the user-submitted object    form.Foo = newItem.Foo;    form.Bar = newItem.Bar;    // etc. for each property in the newItem object    return View(form);  }  else  // etc. |