Test Charter 1

Date: 12/08/23

Tester: Matt Mewis

Test Charter:

Explore the 'Number of Units Required' field on the buy energy form

With different data sets

To discover if energy quantity values are validated before being submitted

Consider:

- Numeric values
- Alphanumeric values
- Special characters
- Negative values
- 0 value
- Value above the maximum available quantity

Testing Notes

I tested the 'Number of Units Required' field for the Gas energy type

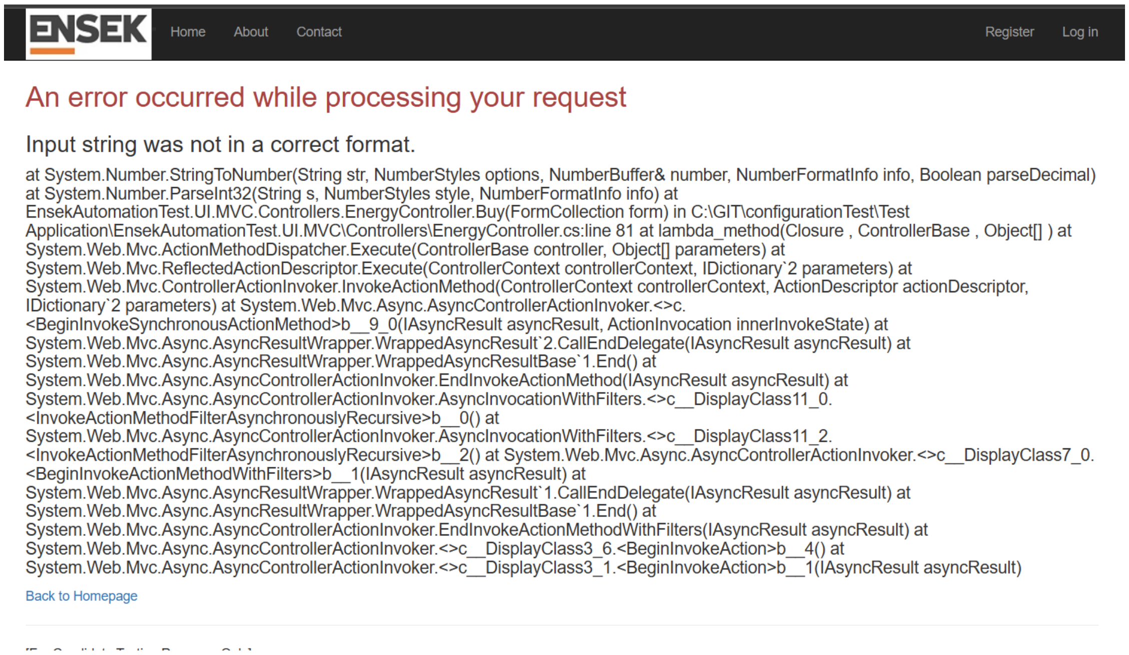
The default quantity was 0, I made no change and clicked 'Buy'. Result: No validation failure (not expected)

I entered a quantity between 1 and 3000. Clicked 'Buy': Result: No validation failure (expected)

I entered a negative quantity, -10, and clicked 'Buy': Result: No validation failure. The available quantity rose by the +10. (not expected)

I entered a number that was above the maximum available quantity, 5000. Result: No validation Failure. The available quantity of the energy type changed to -2000 (not expected)

I then selected 'Buy' when no value was in the field at all. Result: An error page appeared. This stated "An error occurred while processing your request - Input string was not in a correct format.".



An error occurred while processing your request

Input string was not in a correct format.

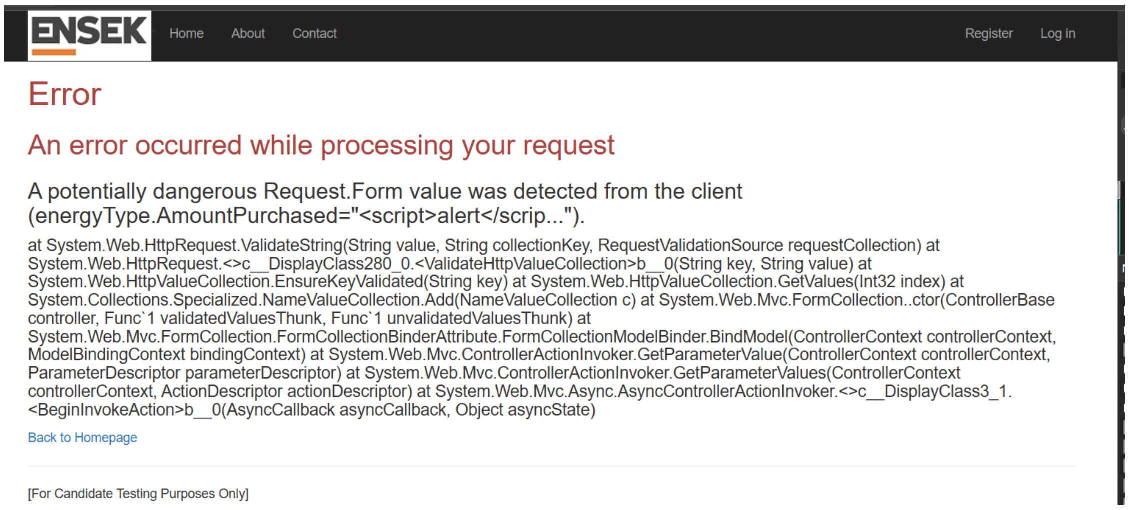
```
at System.Number.StringToNumber(String str, NumberStyles options, NumberBuffer& number, NumberFormatInfo info, Boolean parseDecimal)
at System.Number.ParseInt32(String s, NumberStyles style, NumberFormatInfo info) at
EnsekAutomationTest.UI.MVC.Controllers.EnergyController.Buy(FormCollection form) in C:\GIT\configurationTest\Test
Application\EnsekAutomationTest.UI.MVC\Controllers\EnergyController.cs:line 81 at lambda_method(Closure , ControllerBase , Object[] ) at
System.Web.Mvc.ActionMethodDispatcher.Execute(ControllerBase controller, Object[] parameters) at
System.Web.Mvc.ReflectedActionDescriptor.Execute(ControllerContext controllerContext, IDictionary`2 parameters) at
System.Web.Mvc.ControllerActionInvoker.InvokeActionMethod(ControllerContext controllerContext, ActionDescriptor actionDescriptor,
IDictionary`2 parameters) at System.Web.Mvc.Async.AsyncControllerActionInvoker.<>c.
<BeginInvokeSynchronousActionMethod>b__9_0(IAsyncResult asyncResult, ActionInvocation innerInvokeState) at
System.Web.Mvc.Async.AsyncResultWrapper.WrappedAsyncResult`2.CallEndDelegate(IAsyncResult asyncResult) at
System.Web.Mvc.Async.AsyncResultWrapper.WrappedAsyncResultBase`1.End() at
System.Web.Mvc.Async.AsyncControllerActionInvoker.EndInvokeActionMethod(IAsyncResult asyncResult) at
System.Web.Mvc.Async.AsyncControllerActionInvoker.AsyncInvocationWithFilters.<>c__DisplayClass11_0.
<InvokeActionMethodFilterAsynchronouslyRecursive>b__0() at
System.Web.Mvc.Async.AsyncControllerActionInvoker.AsyncInvocationWithFilters.<>c__DisplayClass11_2.
<InvokeActionMethodFilterAsynchronouslyRecursive>b__2() at System.Web.Mvc.Async.AsyncControllerActionInvoker.<>c__DisplayClass7_0.
<BeginInvokeActionMethodWithFilters>b__1(IAsyncResult asyncResult) at
System.Web.Mvc.Async.AsyncResultWrapper.WrappedAsyncResult`1.CallEndDelegate(IAsyncResult asyncResult) at
System.Web.Mvc.Async.AsyncResultWrapper.WrappedAsyncResultBase`1.End() at
System.Web.Mvc.Async.AsyncControllerActionInvoker.EndInvokeActionMethodWithFilters(IAsyncResult asyncResult) at
System.Web.Mvc.Async.AsyncControllerActionInvoker.<>c__DisplayClass3_6.<BeginInvokeAction>b__4() at
System.Web.Mvc.Async.AsyncControllerActionInvoker.<>c__DisplayClass3_1.<BeginInvokeAction>b__1(IAsyncResult asyncResult)
```

Back to Homepage

It also showed a stack trace, which may expose information that a hacker could use to identify a vector to exploit the underlying system.

I then entered a quantity of 'test' and clicked 'Buy'. Result: Unfriendly error - the same as in the above screenshot.

Next I entered <script>alert</script> into the field and clicked 'Buy'. this was an attempted XXS attack. Result: No validation (not expected). The system detected the danger (good) but threw an unfriendly error.



Error

An error occurred while processing your request

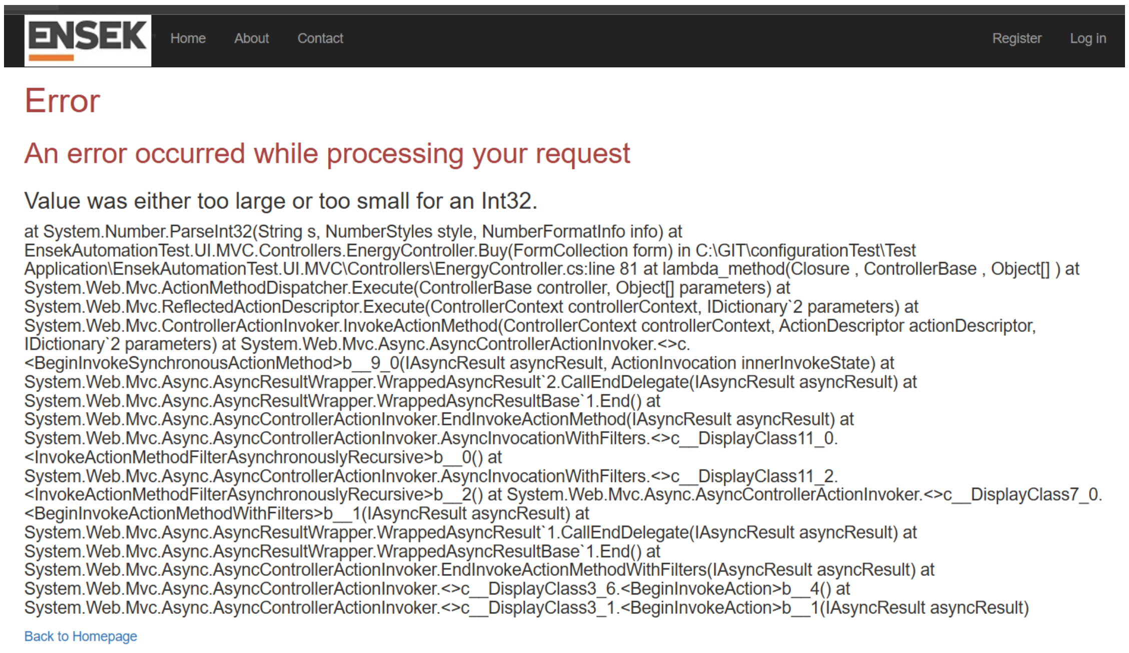
A potentially dangerous Request.Form value was detected from the client (energyType.AmountPurchased="<script>alert</scrip...").

```
at System.Web.HttpRequest.ValidateString(String value, String collectionKey, RequestValidationSource requestCollection) at
System.Web.HttpRequest.<>c__DisplayClass280_0.<ValidateHttpValueCollection>b__0(String key, String value) at
System.Web.HttpValueCollection.EnsureKeyValidated(String key) at System.Web.HttpValueCollection.GetValues(Int32 index) at
System.Collections.Specialized.NameValueCollection.Add(NameValueCollection c) at System.Web.Mvc.FormCollection..ctor(ControllerBase
controller, Func`1 validatedValuesThunk, Func`1 unvalidatedValuesThunk) at
System.Web.Mvc.FormCollection.FormCollectionBinderAttribute.FormCollectionModelBinder.BindModel(ControllerContext controllerContext,
ModelBindingContext bindingContext) at System.Web.Mvc.ControllerActionInvoker.GetParameterValue(ControllerContext controllerContext,
ParameterDescriptor parameterDescriptor) at System.Web.Mvc.ControllerActionInvoker.GetParameterValues(ControllerContext
controllerContext, ActionDescriptor actionDescriptor) at System.Web.Mvc.Async.AsyncControllerActionInvoker.<>c__DisplayClass3_1.
<BeginInvokeAction>b__0(AsyncCallback asyncCallback, Object asyncState)
```

Back to Homepage

[For Candidate Testing Purposes Only]

 I then entered a numeric value that was very long. Result: No validation (not expected). This resulted in the below unfriendly error:



Error

An error occurred while processing your request

Value was either too large or too small for an Int32.

```
at System.Number.ParseInt32(String s, NumberStyles style, NumberFormatInfo info) at
EnsekAutomationTest.UI.MVC.Controllers.EnergyController.Buy(FormCollection form) in C:\GIT\configurationTest\Test
Application\EnsekAutomationTest.UI.MVC\Controllers\EnergyController.cs:line 81 at lambda_method(Closure , ControllerBase , Object[] ) at
System.Web.Mvc.ActionMethodDispatcher.Execute(ControllerBase controller, Object[] parameters) at
System.Web.Mvc.ReflectedActionDescriptor.Execute(ControllerContext controllerContext, IDictionary`2 parameters) at
System.Web.Mvc.ControllerActionInvoker.InvokeActionMethod(ControllerContext controllerContext, ActionDescriptor actionDescriptor,
IDictionary`2 parameters) at System.Web.Mvc.Async.AsyncControllerActionInvoker.<>c.
<BeginInvokeSynchronousActionMethod>b__9_0(IAsyncResult asyncResult, ActionInvocation innerInvokeState) at
System.Web.Mvc.Async.AsyncResultWrapper.WrappedAsyncResult`2.CallEndDelegate(IAsyncResult asyncResult) at
System.Web.Mvc.Async.AsyncResultWrapper.WrappedAsyncResultBase`1.End() at
System.Web.Mvc.Async.AsyncControllerActionInvoker.EndInvokeActionMethod(IAsyncResult asyncResult) at
System.Web.Mvc.Async.AsyncControllerActionInvoker.AsyncInvocationWithFilters.<>c__DisplayClass11_0.
<InvokeActionMethodFilterAsynchronouslyRecursive>b__0() at
System.Web.Mvc.Async.AsyncControllerActionInvoker.AsyncInvocationWithFilters.<>c__DisplayClass11_2.
<InvokeActionMethodFilterAsynchronouslyRecursive>b__2() at System.Web.Mvc.Async.AsyncControllerActionInvoker.<>c__DisplayClass7_0.
<BeginInvokeActionMethodWithFilters>b__1(IAsyncResult asyncResult) at
System.Web.Mvc.Async.AsyncResultWrapper.WrappedAsyncResult`1.CallEndDelegate(IAsyncResult asyncResult) at
System.Web.Mvc.Async.AsyncResultWrapper.WrappedAsyncResultBase`1.End() at
System.Web.Mvc.Async.AsyncControllerActionInvoker.EndInvokeActionMethodWithFilters(IAsyncResult asyncResult) at
System.Web.Mvc.Async.AsyncControllerActionInvoker.<>c__DisplayClass3_6.<BeginInvokeAction>b__4() at
System.Web.Mvc.Async.AsyncControllerActionInvoker.<>c__DisplayClass3_1.<BeginInvokeAction>b__1(IAsyncResult asyncResult)
```

Back to Homepage

The tests noted above were against the 'Gas' type field.

I replicated each issue against the 'electricity' type and 'Oil' types. the same issues were present.

Issues:

A common theme for all issues below is - there is no field validation. This results in undesirable behaviour and unfriendly error messages.

1. No validation against a 0 value. This is not a valid transaction as nothing was purchased.

2. A user can enter and successfully submit a negative number. This then increases the available quantity for the energy type selected. Negative numbers should not be able to be submitted.

3. User able to enter a value of 5000, 2000 above the maximum quantity available for the energy type. This made the energy type -2000. This does not feel right as you cannot purchase more than is available.

4.User is able to select 'Buy' when no value is in a field resulting in an unfriendly error.

5. User is able to input the text 'test' and select 'Buy' when no value is in a field resulting in an unfriendly error. This appears to happen with any non-numeric value (other than -)

6. No validation in place to prevent me entering a potentially dangerous string. This resulted in an unfriendly error.

7. No validation to prevent a very large number being submitted. This resulted in an unfriendly error message.

8. Unfriendly error messages can confuse users and reduce a users confidence in a website/system. It may expose too much information to a potential attacker, increasing risk to the system and the wider organisation.

Bug #1 created covering the lack of validation