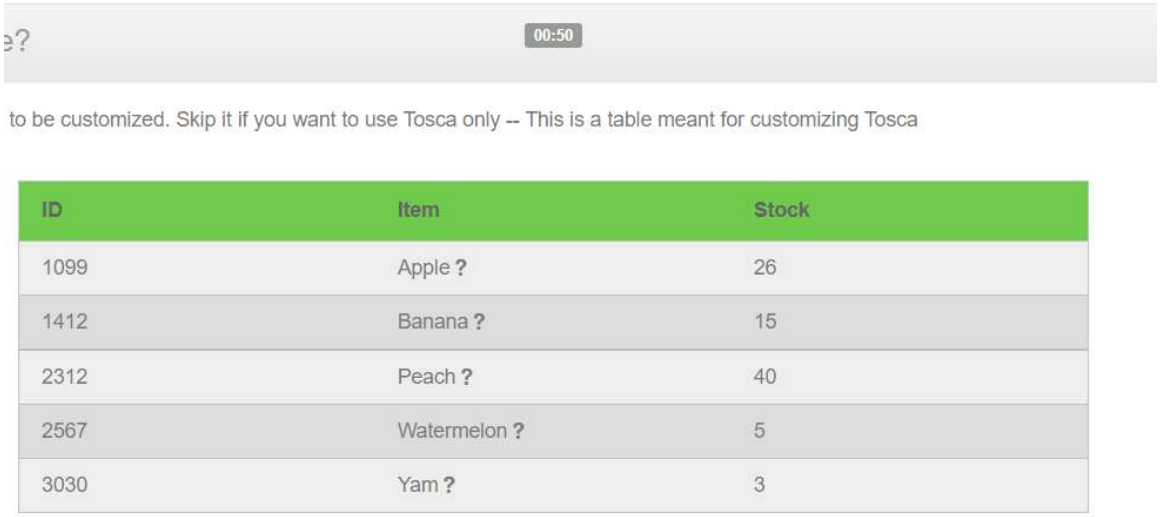
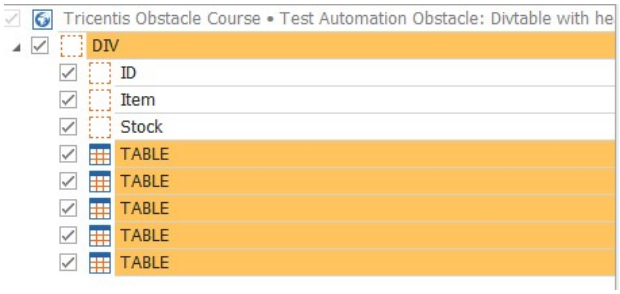
Link to obstacle:

https://obstaclecourse.tricentis.com/Obstacles/Next?oid=30300



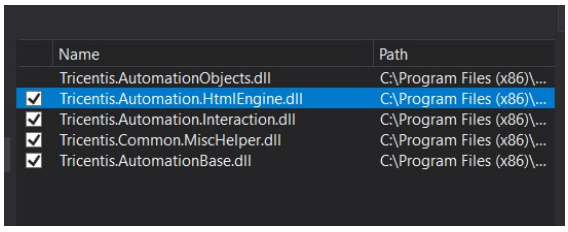
Initial Steering



Hints

In case of this table, some of the cells are td elements – so they are standard HTML table cells. No additional adapter has to be created for those – the platform handles these already for you. Just make sure that your controllers return them.

References



References for Adpater classes

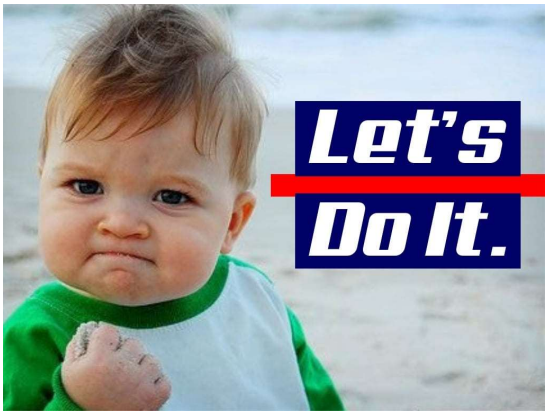
```
using Tricentis.Automation.Creation;
using Tricentis.Automation.Engines.Adapters;
using Tricentis.Automation.Engines.Adapters.Attributes;
using Tricentis.Automation.Engines.Adapters.Html.Generic;
using Tricentis.Automation.Engines.Technicals.Html;
```

References for Controller Classes

```
using System.Collections.Generic;
using Tricentis.Automation.AutomationInstructions.TestActions;
using Tricentis.Automation.AutomationInstructions.TestActions.Associations;
using Tricentis.Automation.Creation;
using Tricentis.Automation.Engines.Adapters.Controllers;
using Tricentis.Automation.Engines.Representations.Attributes;
```

What we will need:

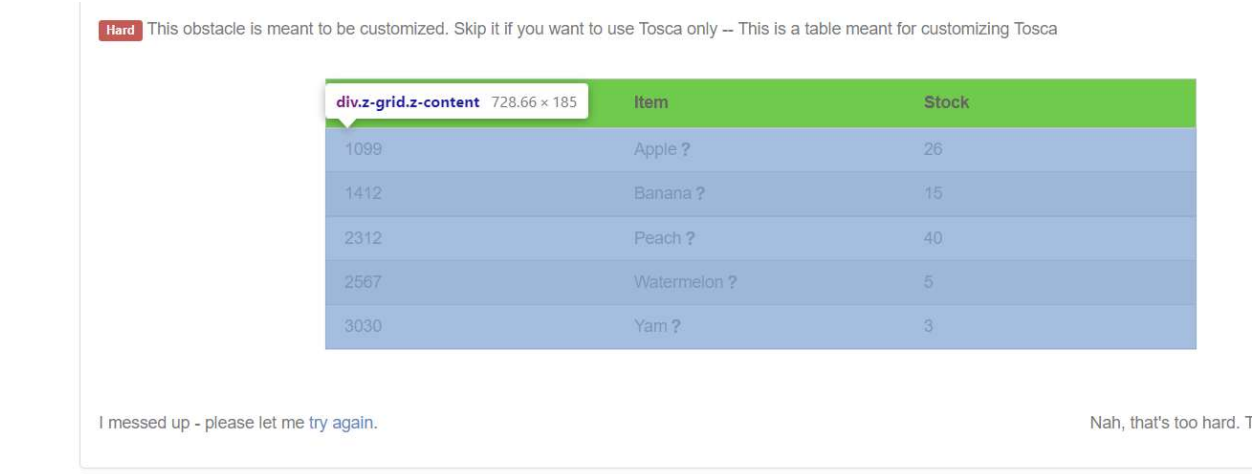
- Table adapter + controller
- Row adapter + controller
- Cell adapter

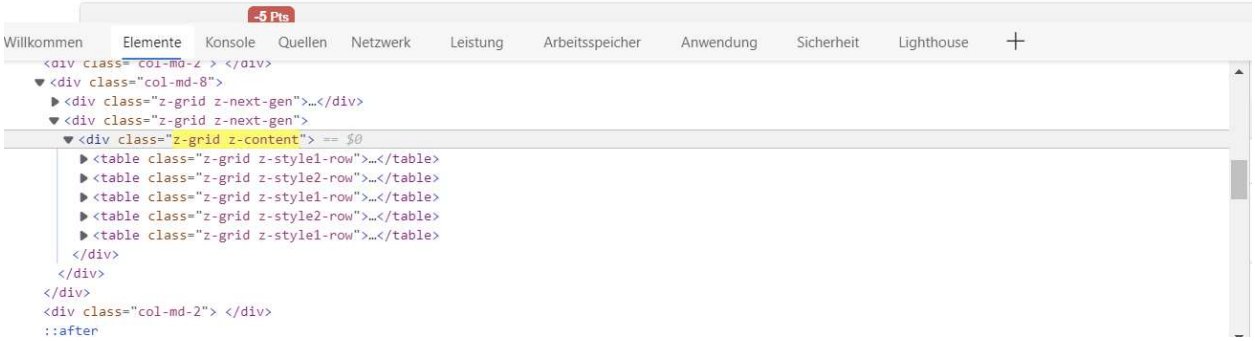


Adapter classes are the easiest part here, so we start with them 

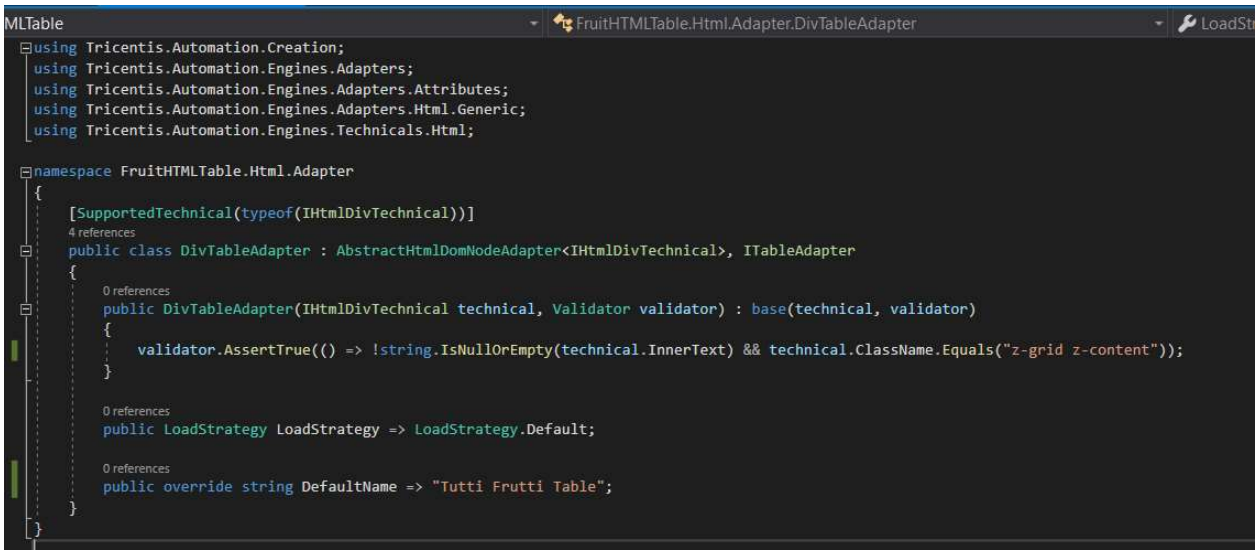
TableAdapter

```
//      This class also implements the ITableAdapter, telling the framework that this
//      Adapter works as a table. This allows the framework
//      to determine that this Adapter works for the according Representation
```





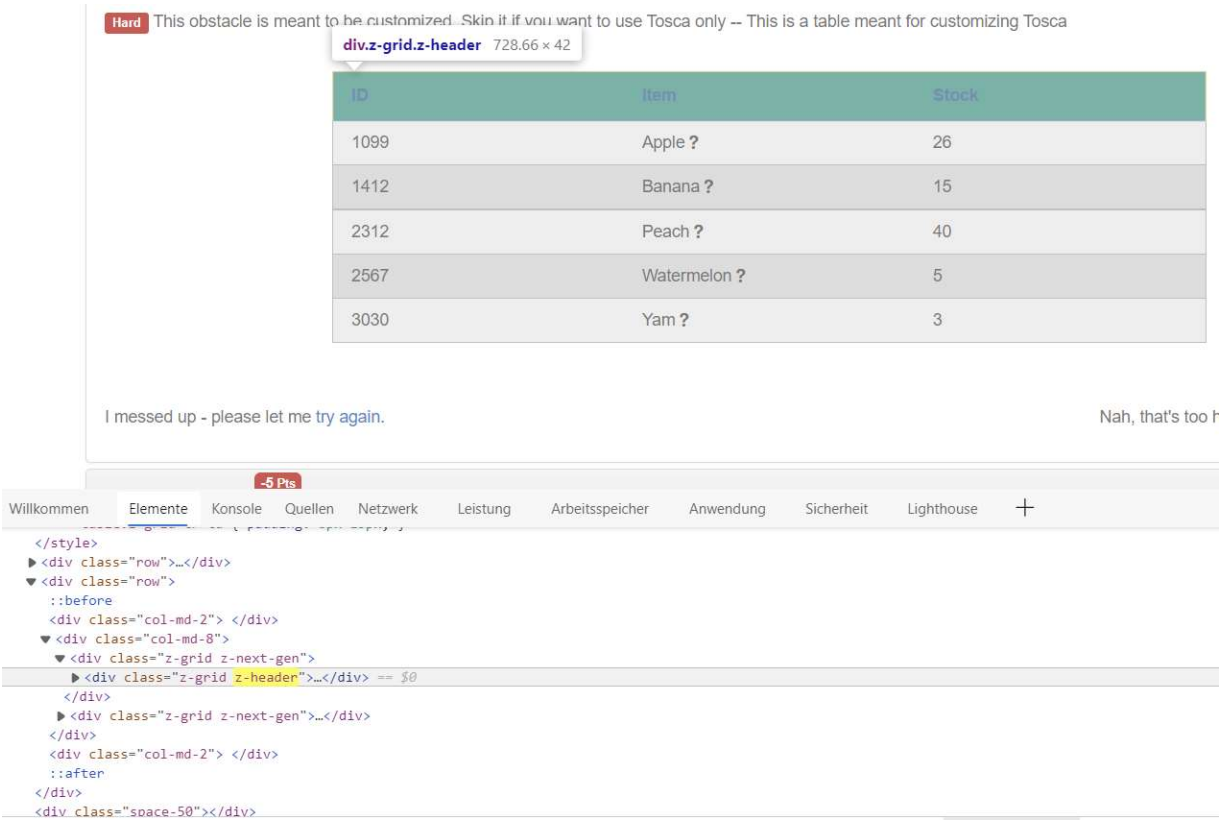
Do not get fooled by col-md-8 !



The table adapter controller will later deal with the question:
How are we getting the rows which are and <tr>?

RowAdapter

```
//      This class represents (or interacts with) the row. The class implements the
//      ITableRowAdapter, allowing the framework to identify this Adapter as the row.
```

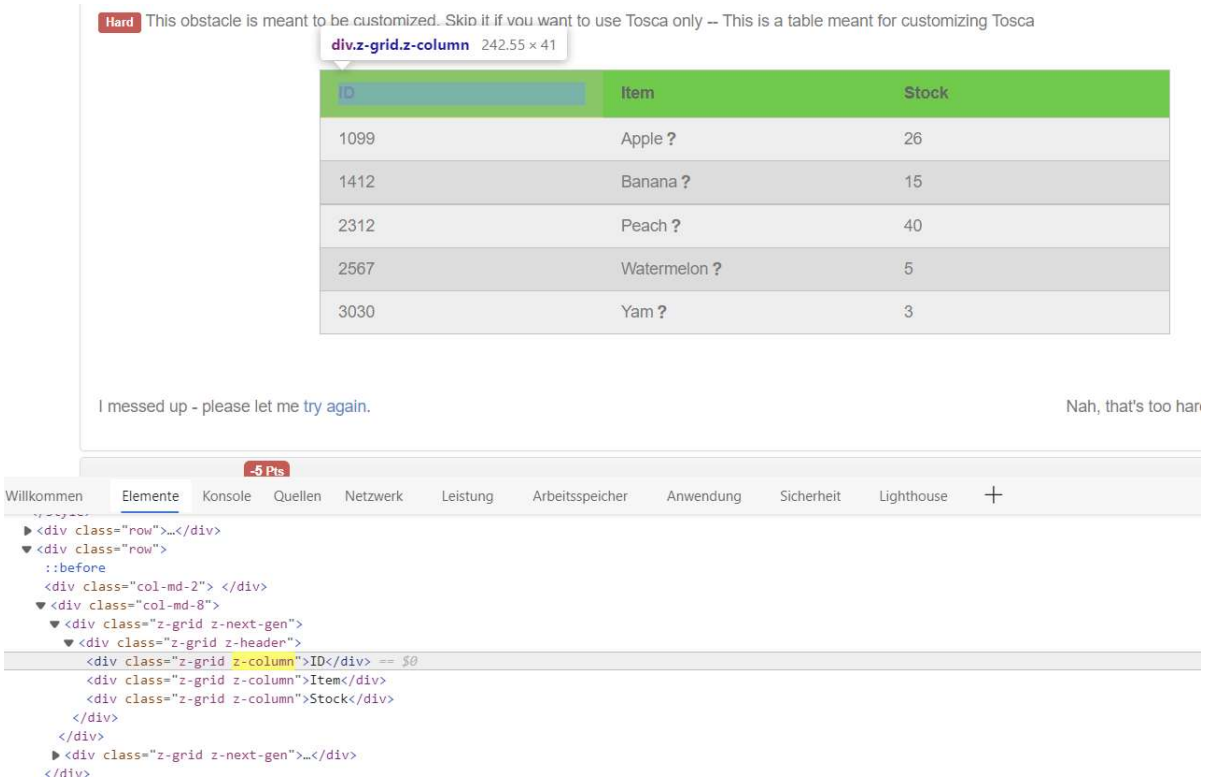



```
namespace FruitHTMLTable.Html.Adapter
{
    /// <summary>
    /// Header row is a <div></div>, all other rows are table <tr></tr>
    /// </summary>
    [SupportedTechnical(typeof(IHtmlDivTechnical))]
    4 references
    public class RowAdapter : AbstractHtmlDomNodeAdapter<IHtmlDivTechnical>, ITableRowAdapter
    {
        0 references
        public RowAdapter(IHtmlDivTechnical technical, Validator validator) : base(technical, validator)
        {
            validator.AssertTrue(() => technical.ClassName.Contains("z-header"));
        }

    }
}
```

Clarify the representation: header row

CellAdapter

```
//      This class represents (or interacts with) the cell. The class implements the
//      ITableCellAdapter, allowing the framework to identify this Adapter as a cell.
```




```
namespace FruitHTMLTable.Html.Adapter
{
    [SupportedTechnical(typeof(HtmlElementTechnical))]
    1 reference
    public class CellAdapter : AbstractHtmlDomNodeAdapter<IHtmlDivTechnical>, ITableCellAdapter
    {
        0 references
        public CellAdapter(IHtmlDivTechnical technical, Validator validator) : base(technical, validator)
        {
            validator.AssertTrue(() => technical.ClassName.Contains("z-column"));
        }

        0 references
        public int ColSpan => 1;

        0 references
        public int RowSpan => 1;

        0 references
        public string Text => Technical.InnerText;

    }
}
```

TableAdapterController

Tell the platform how to move from the table to the rows.

```
0 references
protected override IEnumerable<IAssociation> ResolveAssociation(DescendantsBusinessAssociation businessAssociation)
{
    yield return new TechnicalAssociation("All");
}
0 references
protected override IEnumerable<IAssociation> ResolveAssociation(ColumnsBusinessAssociation businessAssociation)
{
    throw new NotSupportedException();
}
0 references
protected override IEnumerable<IAssociation> ResolveAssociation(RowsBusinessAssociation businessAssociation)
{
    yield return new AlgorithmicAssociation("Rows");
}
0 references
protected override IEnumerable<ITechnical> SearchTechnicals(IAlgorithmicAssociation algorithmicAssociation)
{
    return (algorithmicAssociation.AlgorithmName != "Rows") ? base.SearchTechnicals(algorithmicAssociation) : GetRows();
}

1 reference
private IEnumerable<ITechnical> GetRows()
{
    return ContextAdapter.Technical.ParentNode.Get<IHtmlDivTechnical>()
                .ParentNode.Get<IHtmlDivTechnical>()
                .All.Get<ITechnical>();
}
#endregion
```

We make use of the *ResolveAssociation(DescendantBusinessAssociation)* to get *all* the rows of the table:

- Contextadapter is our table
  => z-content node in the html
- Parentnode of Contextadapter
  => z-next-gen node in the html
- Parentnode of z-next-gen
  => col-md-8 node
- All.Get<ITechnical>() grants us access to all elements below the col-md-8 node

```
::before
<div class="col-md-2"> </div>
▼ <div class="col-md-8">
  ▼ <div class="z-grid z-next-gen">
    ▶ <div class="z-grid z-header">…</div>
    </div>
  ▼ <div class="z-grid z-next-gen">
    ▼ <div class="z-grid z-content"> == $0
      ▶ <table class="z-grid z-style1-row">…</table>
      ▶ <table class="z-grid z-style2-row">…</table>
      ▶ <table class="z-grid z-style1-row">…</table>
      ▶ <table class="z-grid z-style2-row">…</table>
      ▶ <table class="z-grid z-style1-row">…</table>
    </div>
  </div>
```

RowAdapterController

Tell the platform how to get from row to cells.

We only have to deal with the header since the table rows already „know“ how to access their cells.

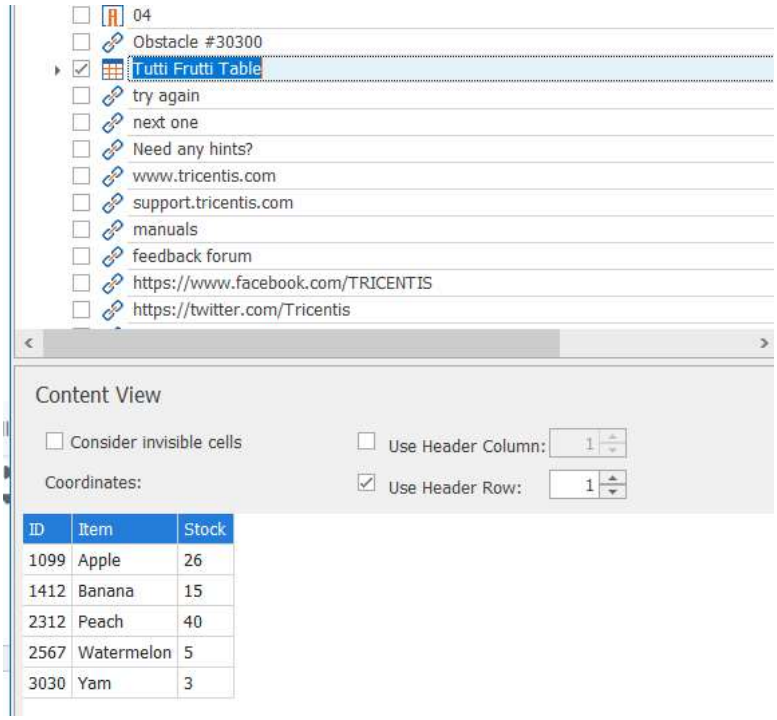
```
/// </summary>
[SupportedAdapter(typeof(RowAdapter))]
1 reference
public class RowAdapterController : TableRowContextAdapterController<RowAdapter>
{
    0 references
    public RowAdapterController(RowAdapter contextAdapter, ISearchQuery query, Validator validator) : base(contextAdapter, query, validator)
    {
    }

    0 references
    protected override IEnumerable<IAssociation> ResolveAssociation(DescendantsBusinessAssociation businessAssociation)
    {
        yield return new TechnicalAssociation("All");
    }

    0 references
    protected override IEnumerable<IAssociation> ResolveAssociation(CellsBusinessAssociation businessAssociation)
    {
        yield return new TechnicalAssociation("Children");
    }

    0 references
    protected override IEnumerable<IAssociation> ResolveAssociation(ChildrenBusinessAssociation businessAssociation)
    {
        yield return new TechnicalAssociation("Children");
    }

    0 references
    protected override IEnumerable<IAssociation> ResolveAssociation(ParentBusinessAssociation businessAssociation)
    {
        yield return new TechnicalAssociation("ParentNode");
    }
}
```

The cells are the direct children of z-header:

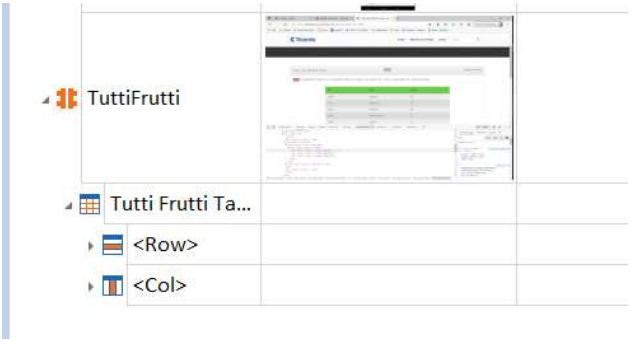
```
::before
<div class="col-md-2"> </div>
<div class="col-md-8">
  <div class="z-grid z-next-gen">
    <div class="z-grid z-header">
      <div class="z-grid z-column">ID</div> == $0
      <div class="z-grid z-column">Item</div>
      <div class="z-grid z-column">Stock</div>
    </div>
  </div>
  <div class="z-grid z-next-gen">…</div>
</div>
<div class="col-md-2"> </div>
::after
```

Tosca Commander

XScan

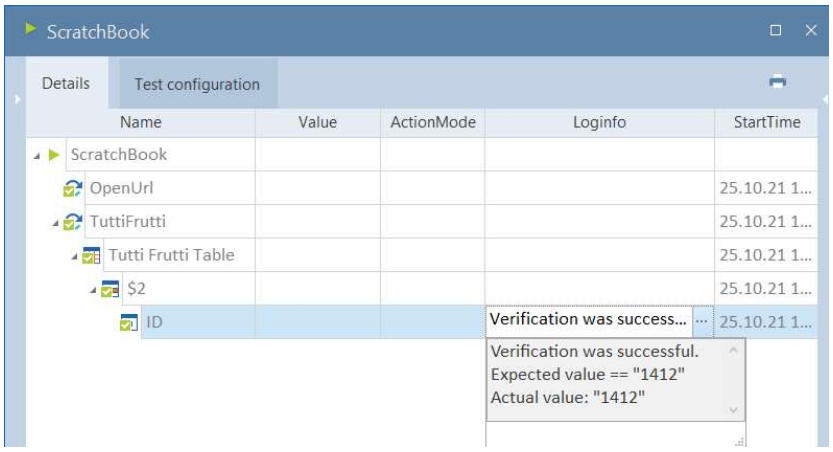


Module after steering the table:



Test Case

Exercise 2b | Verify Banana ID				PLANNED
OpenUrl				
Url	https://obstaclecourse.tricentis.com/Obstacles/Next?oid=30300	Input	String	
UseActiveTab		Input	String	
WebDriverBrowserArguments			String	
TuttiFrutti				
Tutti Frutti Table		Select	String	
$2		Select	String	
ID	1412	Verify	String	
<Cell>			String	
<Row>			String	
<Col>			String	


ScratchBook
Details
Test configuration
Name
Value
ActionMode
Loginfo
StartTime
ScratchBook
OpenUrl
25.10.21 1...
TuttiFrutti
25.10.21 1...
Tutti Frutti Table
25.10.21 1...
$2
25.10.21 1...
ID
Verification was success...
25.10.21 1...
Verification was successful.
Expected value == "1412"
Actual value: "1412"