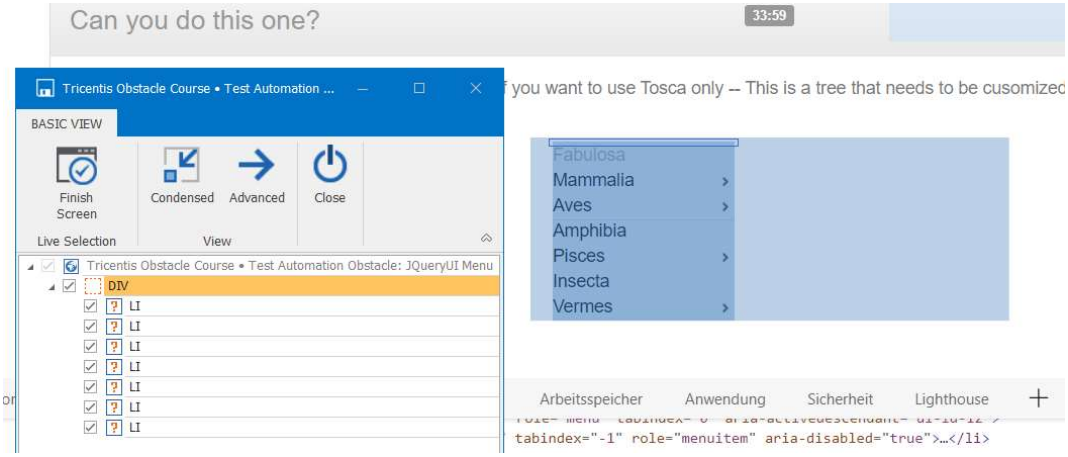
Link to obstacle

https://obstaclecourse.tricentis.com/Obstacles/Next?oid=98638

Initial Steering



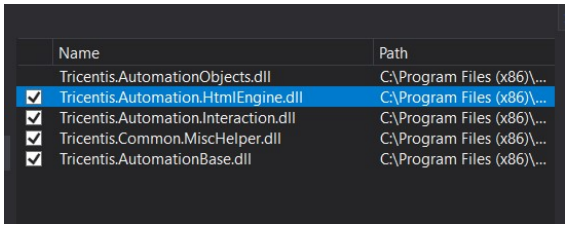
Hints

This tree implementation is much simpler. No Technicals are required like in 3a.

TBox API example:

https://documentation.tricentis.com/devcorner/1200/tboxapi/#topic37.html

References



References for Adpater classes

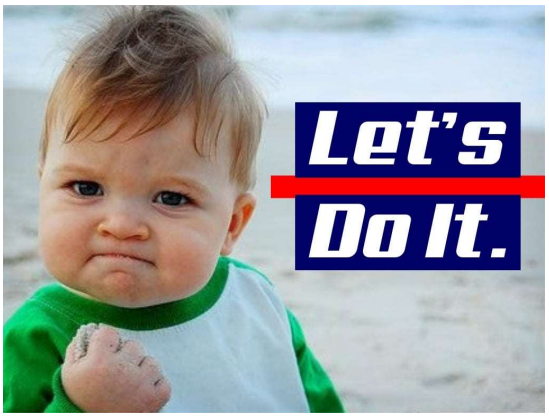
```
using Tricentis.Automation.Creation;
using Tricentis.Automation.Engines.Adapters;
using Tricentis.Automation.Engines.Adapters.Attributes;
using Tricentis.Automation.Engines.Adapters.Html.Generic;
using Tricentis.Automation.Engines.Technicals.Html;
```

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:

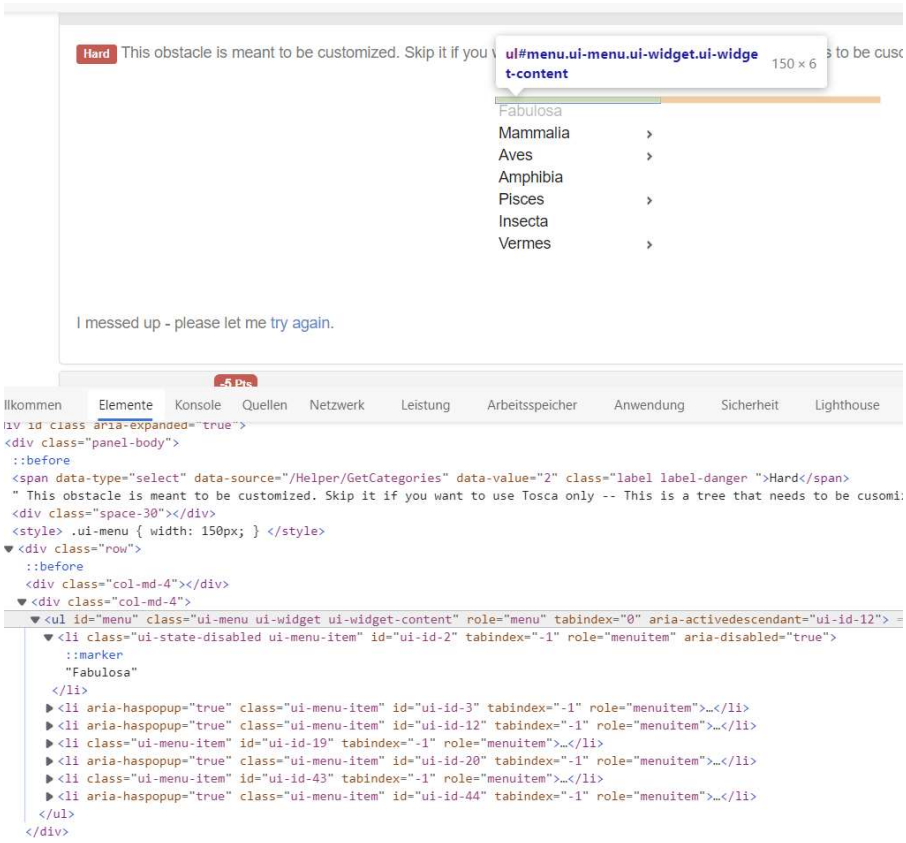
- Tree adapter + controller
- Tree node adapter + controller



TreeAdapter

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

Tree is an <ul> tag, with no separate Technical for -tags, we will use the IHtmlElementTechnical which is generic for all HTML elements👌




```
namespace ZootopiaHtmlTree.Html.Adapter
{
    [SupportedTechnical(typeof(IHtmlElementTechnical))]
    1 reference
    public class TreeAdapter : AbstractHtmlDomNodeAdapter<IHtmlElementTechnical>, ITreeAdapter
    {
        0 references
        public TreeAdapter(IHtmlElementTechnical technical, Validator validator) : base(technical, validator)
        {
            validator.AssertTrue(() => technical.Tag.ToLower().Equals("ul") && technical.ClassName.Contains("ui-menu"));
        }

        0 references
        public override string DefaultName => "Latin Zootopia Tree";
    }
}
```

TreeAdapterController

The first step we have to make is from the Tree to the first layer of nodes. This can be achieved relatively easy, because they are just the direct children of the <ul> element:

```
<ul id="menu" class="ui-menu ui-widget ui-widget-content" role="menu" tabindex="0" aria-activedescendant="ui-id-3">
  <li class="ui-state-disabled ui-menu-item" id="ui-id-2" tabindex="-1" role="menuitem" aria-disabled="true">…</li>
  <li aria-haspopup="true" class="ui-menu-item" id="ui-id-3" tabindex="-1" role="menuitem">…</li>
  <li aria-haspopup="true" class="ui-menu-item" id="ui-id-12" tabindex="-1" role="menuitem">…</li>
  <li class="ui-menu-item" id="ui-id-19" tabindex="-1" role="menuitem">…</li>
  <li aria-haspopup="true" class="ui-menu-item" id="ui-id-20" tabindex="-1" role="menuitem">…</li>
  <li class="ui-menu-item" id="ui-id-43" tabindex="-1" role="menuitem">…</li>
  <li aria-haspopup="true" class="ui-menu-item" id="ui-id-44" tabindex="-1" role="menuitem">…</li>
</ul>
</div>
```

```
namespace ZootopiaHtmlTree.Html.Adapter.Controller
{
    [SupportedAdapter(typeof(TreeAdapter))]
    1 reference
    public class TreeAdapterController : TreeContextAdapterController<TreeAdapter>
    {
        0 references
        public TreeAdapterController(TreeAdapter contextAdapter, ISearchQuery query, Validator validator) : base(contextAdapter, query, validator)
        {
        }

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

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

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

        // direct children of the <ul> element
        0 references
        protected override IEnumerable<IAssociation> ResolveAssociation(TreeNodeBusinessAssociation businessAssociation)
        {
            yield return new TechnicalAssociation("Children");
        }
    }
```

TreeNodeAdapter

```
//      This class represents (or interacts with) one of the nodes of the menu tree.
//      All that can be done with the node, will be implemented here.
```

A node is a <li> element, with no separate Technical for -tags, we will use the IHtmlElementTechnical which is generic for all HTML elements.

We have 2 types of nodes:
(important when we deal with *Name* from the Interface IMenuItemadapter)

- Node with subnodes (e.g. Mammalia)
- Node with no subnodes (e.g. Amphibia)


```
[SupportedTechnical(typeof(IHtmlElementTechnical))]
4 references
public class TreeNodeAdapter : AbstractHtmlDomNodeAdapter<IHtmlElementTechnical>, IMenuItemAdapter
{
    0 references
    public TreeNodeAdapter(IHtmlElementTechnical technical, Validator validator) : base(technical, validator)
    {
        validator.AssertTrue(() => technical.Tag.Equals("li", StringComparison.OrdinalIgnoreCase) && technical.ClassName.Contains("ui-menu-item"));
    }

    0 references
    public override string DefaultName => Name;
    #region Interface IMenuItemAdapter
    1 reference
    public string Name => GetName();


    /// <summary>
    /// Here we check if there is a span element inside the li - if yes, we'll have to use a regular expression
    /// to retrieve the name - InnerText would not work, since it would also retrieve the names of the subnodes.
    /// </summary>
    1 reference
    private string GetName()
    {
        if (Technical.Children.Get<IHtmlSpanTechnical>().Any())
        {
            Regex name = new Regex(@"span\>(?<title>.*?)\<ul", RegexOptions.Singleline);
            return name.Match(Technical.InnerHtml).Groups["title"].Value.Trim();
        }
        else
        {
            return Technical.InnerText;
        }
    }

    0 references
    public void Select()
    {
        Mouse.PerformMouseAction(MouseOperation.Click, ActionPoint);
    }
    #endregion
```

Tutorial for regular expressions:
https://www.softwaretestinghelp.com/csharp-regex-tutorial/

Test your expression:
https://regex101.com/

REGULAR EXPRESSION

1 match (93 steps, 0.9ms)

/ span\>(?<title>.*?)\<ul / gm

TEST STRING

```
<li aria-haspopup="true" class="ui-menu-item" id="ui-id-3" tabindex="-1"
role="menuitem"><span class="ui-menu-icon ui-icon ui-icon-carat-1-e">
</span>Mammalia<ul class="ui-menu ui-widget ui-widget-content ui-front"
role="menu" aria-expanded="false" style="display: none; top: -1.21249px;
left: 142.275px;" aria-hidden="true"><li class="ui-menu-item" id="ui-id-
4" tabindex="-1" role="menuitem">Primates</li><li class="ui-menu-item"
id="ui-id-5" tabindex="-1" role="menuitem">Bruta</li><li class="ui-menu-
item" id="ui-id-6" tabindex="-1" role="menuitem">Ferae</li><li class="ui-
menu-item" id="ui-id-7" tabindex="-1" role="menuitem">Bestiae</li><li
class="ui-menu-item" id="ui-id-8" tabindex="-1"
role="menuitem">Glires</li><li class="ui-menu-item" id="ui-id-9"
tabindex="-1" role="menuitem">Pecora</li><li class="ui-menu-item" id="ui-
id-10" tabindex="-1" role="menuitem">Belluae</li><li class="ui-menu-item"
id="ui-id-11" tabindex="-1" role="menuitem">Cete</li></ul></li>
```

TreeNodeAdapterController

The interesting parts of the controller class:

```
protected override IEnumerable<IAssociation> ResolveAssociation(DescendantsBusinessAssociation businessAssociation)
{
    yield return new TechnicalAssociation("All");
}
0 references
protected override IEnumerable<IAssociation> ResolveAssociation(TreeNodeBusinessAssociation businessAssociation)
{
    yield return new AlgorithmicAssociation("SubNodes");
}
0 references
protected override IEnumerable<ITechnical> SearchTechnicals(IAlgorithmicAssociation algorithmicAssociation)
{
    if (algorithmicAssociation.AlgorithmName == "SubNodes")
    {
        return GetSubNodes();
    }
    return base.SearchTechnicals(algorithmicAssociation);
}

1 reference
private IEnumerable<ITechnical> GetSubNodes()
{
    IHtmlElementTechnical ulElement = ContextAdapter.Technical.Children.Get<IHtmlElementTechnical>().FirstOrDefault(t => t.Tag.ToLower() == "ul");
    if (ulElement != null)
    {
        return ulElement.Children.Get<ITechnical>();
    }
    return new ITechnical[] { };
}
```