1 HOW TO CREATE (UDPDATE) LESSON

1. Create new class `LessonZ.cs` in **Lessons** folder.[1]

Example LessonZ.cs

```
public class LessonZ : LessonBase
{
        public LessonZ()
                {
                        var lessonZXmlRelativePath = @"Lessons\LessonZ.xml";
                        var lessonProvider = new LessonUserInterfaceProvider(lessonZRelativePath);
                        Steps = lessonProvider.LessonSteps;
                }
}
```

2. In folder **Steps/Validation/Validators**[1] create folder **LessonZ** and create validator classes.

Example LessonZStepXValidator.cs

```
namespace DotvvmAcademy.Steps.Validation.Validators.LessonZ
{
        [StepValidation(ValidatorKey = "LessonZStepXValidator")]
        public class LessonZStepXValidator: IDotHtmlCodeValidationObject
        {
                public void Validate(ResolvedTreeRoot resolvedTreeRoot)
                {
                        ValidatorHelper.ValidateBasicControls(resolvedTreeRoot);
                }
        }
}
```

By using reflection ValidatorProvider will be created new instance Validator Class (`LessonZStepXValidator.cs`) base on ValidatorKey from atribute. You can implement two interfaces for validate DotHtml (`IDotHtmlCodeValidationObject`) or C# (`ICSharpCodeValidationObject`) syntax which is bound on type (Step Type="DothtmlCode") in XML.

3. Now you can create xml doc *LessonY.xml* in Lessons folder.

*Steps will be lined up according to **position** from XML doc.*
If you want to put new step between steps which already exist,
you can simply create new step on the place between steps.

Example LessonZ.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Lesson>
        <!-- namespace of all validators assigned to current lesson-->
        <ValidatorsNamespace>
        DotvvmAcademy.Steps.Validation.Validators.LessonZ
        </ValidatorsNamespace>
        <Steps>
                <!--Required attribute-->
                <Step Type="Info">
                        <!--Required element-->
                        <Title>XXXXXXX</Title>
                         <!--Required element-->
                        <Description><![CDATA[XXXXXX]]></Description>
                        <!--Optional element-->
                        <ShadowBoxDescription><![CDATA[XXXXXX]]>
                        </ShadowBoxDescription>
                </Step>
                <Step Type="CsharpCode">
                        <Title>XXXXXXX</Title>
                        <ValidationKey>XXXXXX</ValidationKey>
                        <Description><![CDATA[XXXXXX]]></Description>
                        <!--Optional element-->
                        <ShadowBoxDescription><![CDATA[XXXXXX]]>
                        </ShadowBoxDescription>
                        <!--Required element-->
                        <StartupCode><![CDATA[XXXXXX]]></StartupCode>
                        <!--Required element-->
                        <FinalCode><![CDATA[XXXXXX]]></FinalCode>
                        <!--Optional element-->
                        <CodeDependencies>
                                <CodeDependency><![CDATA[XXXXXX]]>
                                </CodeDependency>
                                <CodeDependency><![CDATA[XXXXXX]]>
                                </CodeDependency>
                        </CodeDependencies>
                        <!--Optional element-->
                        <AllowedTypesConstructed>
                        <AllowedType><![CDATA[XXXXXX]]></AllowedType>
                        </AllowedTypesConstructed>
                        <!--Optional element-->
                        <AllowedMethodsCalled>
                                <AllowedMethod><![CDATA[XXXXXX]]>
                                </AllowedMethod>
                        </AllowedMethodsCalled>
                </Step>
        </Steps>
</Lesson>
```

`ValidatorClass` must be in the same namespace like `ValidatorsNamespace` in xml file!
`ValidatorClass` must have parameterless ctor!

*If you create new validation message, please use resources file **Steps/Validation/Validators/ValidationErrorMessages.resx**. Next point, in `DotvvmAcademy.Steps.Validation.Validator`, contains class `ValidationExtensions` with method `ExecuteSafe` which catch `RuntimeBinderException` exception.*

```
DotvvmAcademy. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . Project
├── . . .
├── Lessons. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . folder with all Lessons
│   ├── Lesson1.cs. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . lesson class
│   ├── Lesson1.xml. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . lesson data
│   ├── . . .
│   └── LessonBase.cs. . . . . . . . . . . . . . . . . . . . . . . . lesson base abstract class
├── . . .
├── Steps
│   ├── StepBuilder
│   ├── StepsBases
│   │   └── . . . .
│   ├── Validation
│   │   ├── Interfaces
│   │   ├── ValidatorProvision
│   │   ├── Validators
│   │   │   ├── Lesson1. . . . . . . . folder with validator classes for lesson
│   │   │   │   └── Lesson1Step3Validator.cs. . . . . . . . . . . . . . . Validator class
│   │   │   ├── . . .
│   │   │   ├── ValidationErrorMessages.resx. Validation Error messages
│   │   │   ├── ValidationExtensions.cs. . . . . . Methods like ExecuteSafe
│   │   │   └── ValidatorHelper.cs
│   │   └── . . .
│   ├── CodeStepCsharp.cs
│   ├── CodeStepDotHtml.cs
│   ├── ChoicesStep.cs
│   ├── ChoiceStepOption.cs
│   └── InfoStep.cs
└── . . .
```