# Portfolio Nr. 6 - Überprüfung und Verarbeitung eines String mit ASP.Net

Portfolio

Fakultät für Wirtschaft

Studiengang Wirtschaftsinformatik

Studienjahrgang 2018

Kurs C

DUALE HOCHSCHULE BADEN-WÜRTTEMBERG
VILLINGEN-SCHWENNINGEN


Bearbeiter:

David Bährens

Betreuender Dozent:

Prof. Dr. Kimmig

Dualer Partner:

DATEV eG



# Inhaltsverzeichnis

# Abkürzungsverzeichnis

| | |
|---|---|
| **Abb.** | Abbildung |
| **ASP** | Active Server Pages |
| **FCL** | Framework Class Library |
| **CLR** | Common Language Runtime |
| **UI** | User Interface |
| **OS** | Operating System |
| **MS** | Microsoft |
| **z. B.** | zum Beispiel |
| **API** | Application-Programming-Interface |
| **ASP** | Active Server Pages |
| **REST** | Representational State Transfer |
| **MVC** | Model-View-Controller |
| **VS** | Visual Studio |

# Abbildungsverzeichnis

# Tabellenverzeichnis

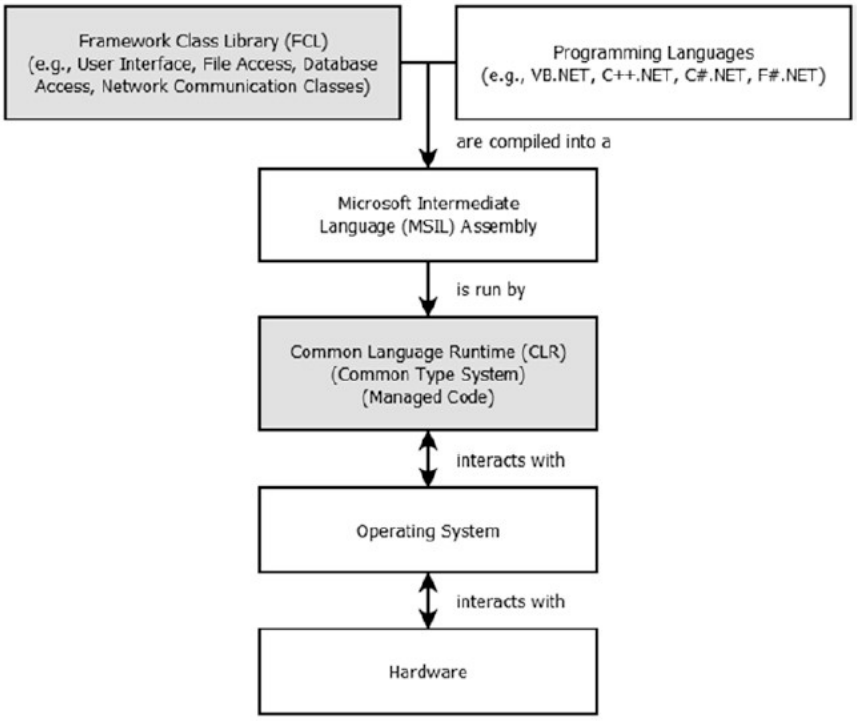
# 1 Einleitung

# 2 Theoretische Grundlagen

## 2.1 ASP.Net

Bei ASP.Net handelt es sich um ein modulares und serverseitiges Web-Framework zur Entwicklung von dynamischen Web-Anwendungen. ASP steht für „Acrive Server Pages“. Dieses ist Teil des Microsoft (MS) Softwareentwicklungs und Execution-Framework .Net. .Net dient unter Windows zur Erstellung von Anwendungsprogrammen. Dessen wesentlichen beiden Bestandteile sind zum einen die Framework Class Library (FCL) und die Common Language Runtime (CLR). FCL ist eine umfangreiche Klassenbibliothek in .Net. Sie enthält beispielsweise User Interface (UI)-, File Access- oder Netzwerk-Kommunikationsklassen. Bei CLR handelt es sich um die Laufzeitumgebung in der .Net Anwendungen ausgeführt werden. .Net Programme, beispielsweise eine C# Anwendung, greifen nicht direkt auf das Betriebssystem (OS) zu. Stattdessen wir der Programmcode in die sogenannte MS Intermediate Language Assemby kompiliert und dann in der CLR ausgeführt. Die CLR wiederum greift dann direkt auf das darunterliegende OS zu (siehe Abb. 1).[1]

Abbildung 1: .Net Framework




Quelle: Beasley, Robert E., .Net Basics, 2020, S. 9

Das .Net Framework wird jedoch fortlaufend durch .Net Core abgelöst. Hierbei han-

[1] Vgl. Beasley, Robert E., .Net Basics, 2020, S. 8

delt es sich um eine Open-Source-Plattform von MS und eine Modernisierung des .Net Frameworks. Ein besonderer Vorteil dieses moderneren Frameworks ist seine Plattformunabhängigkeit. Da ASP.Net auf .Net basiert erfolgt die Ablösung hier analog mit ASP.Net Core.[2]

ASP.Net wiederum stellt verschiedene Frameworks für die Entwicklung von Web-Anwendungen zur Verfügung. Die wichtigsten sind Folgende. Zum einen gibt es das inzwischen veraltete, ereignisgesteuerte Framework „Web Forms“. Bei diesem wurden Oberflächen über einen Designer mit einer Drag-and-Drop Mechanik erstellt und die Logik wurde über einen Eventhandler implementiert. In der Vergangenheit kam Web Forms jedoch teilweise mit der Zustandslosigkeit des Webs in Konflikt.
Ein moderneres, aktionsgesteuertes Framework stellt dagegen ASP.Net MVC dar. Dieses folgt den MVC Pattern, wodurch UI, Logik und Daten voneinander getrennt werden. MVC steht dabei für die drei wesentlichen Bestandteile in die eine MVC-Anwendung zerlegt wird, „Model-View-Controller“. Das Model gibt dabei die Datenstruktur, die View die Darstellung bzw. die UI und der Controller beinhaltet die Logik und verbindet das Model mit der View.[3]
Dann sind da noch die Web Pages, die über die neue Razor Syntax verfügen. Razor Pages sind eine moderne Alternative zur Entwicklung von dynamischen Websites und sie stellen den Nachfolger von ASP.Net MVC dar. Sie werden in einem gesonderten Kapitel erläutert, da sie für diese Arbeit von größerer Bedeutung sind.
Zuletzt ist noch ASP.Net Web API zu nennen, mit dessen Hilfe Web-Schnittstellen wie z. B. REST entwickelt werden können.[4]

Die .Net Entwicklung ist mit einer Reihe kompatibler Programmiersprachen möglich. Hierzu zählen beispielsweise Visual Basic, C# oder F#. Diese Arbeit bezieht sich im folgenden lediglich auf C#. C# ist eine objektorientierte und typsichere höhere Programmiersprache von MS. Ursprünglich war sie primär auf Windows ausgerichtet, inzwischen ist sie jedoch sehr universell und kann für die Entwicklung von Web-Apps eingesetzt werden.[5]

[2] Augsten, Stephan, .Net Core, 2020
[3] Rouse, Margaret, MVC, 2016
[4] Gutsch, Jürgen, ASP.Net, 2017
[5] Augsten, Stephan, C#, 2019

## 2.2 Razor Pages

### 2.2.1 Allgemein

Razor Pages basieren auf ASP.Net MVC und zeichnen sich durch die Razor Syntax aus. Mit dieser können statische HTML Webseiten mit C# dynamisch gemacht werden. Dies zeichnet sich dadurch aus, dass eine Webseite durch eine .cshtml Datei erstellt wird, also eine Kombination aus C#, mittels der Razor Syntax und HTML. Der dort enthaltene Code wird dabei serverseitig in reines HTML übersetzt. Razor Pages vereinen die Vorteile einer verhältnismäßig einfachen Syntax mit einem leichtgewichtigen und dennoch mächtigen Framework. Anders als ASP.Net MVC nutzen Razor Pages das Model-View-ViewModel-Pattern statt echtem MVC. Hierbei handelt es sich um eine spezielle Form der MVC-Architektur, bei der kein Controller, sondern stattdessen ein ViewModel, das bei Razor Pages PageModel genannt wird, eingesetzt wird. Dieses ist eine spezielle Implementierung eines Controllers, welcher die Logik und Programmcode hinter einer View darstellt. Jede View hat ein eigenes ViewModel, statt einem zentralen Controller, der alle Views steuert. Model und View funktionieren analog zu ihrer Funktionalität bei MVC. Die View erhält ihre benötigten Daten dabei mittels Data Binding. Analog zu MVC ist der Zweck MVVM's die Trennung von Logik, UI und Daten. Diese erfolgt bei MVVM allerdings seitenbasiert.[6] Razor Pages sind automatisch auch bei einem ASP.Net MVC Projekt aktiviert und anders herum kann auch in einer Razor Page mit der MVC-Architektur gearbeitet werden falls dies gewünscht wird. [7]

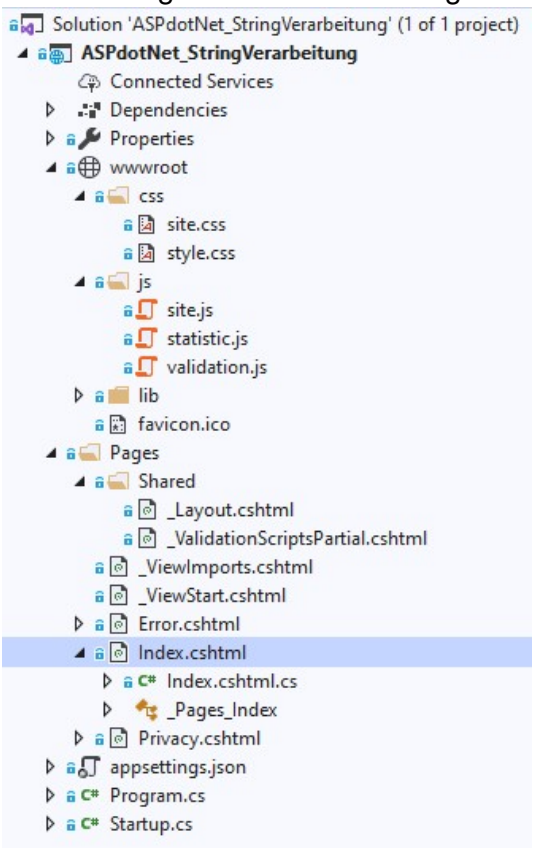
### 2.2.2 Aufbau eines Razor Page Projektes

Ein Standard Razor Projekt besteht aus verschiedenen unterschiedlichen Dateien (siehe Abb. 2). Die wohl wichtigsten befinden sich in dem „Page“ Ordner. Dieser enthält .cshtml Dateien und .cshtml.cs Dateien. Die .cshtml Dateien sind die Views bzw. die eigentliche Razor Page. Mit Hilfe der Razor Syntax könnte hierin auch Logik implementiert werden und somit jeglicher Quellcode in einer Datei gebündelt werden. Dies widerspricht allerdings dem Prinzip der Datentrennung bei MVVM und ist kein guter Programmierstil. Die .cshtml.cs dagegen sind die PageModels der Razor Page. Diese können sowohl die Datenstruktur, als auch die Logik beinhalten. Dies erkennt man im Übrigen auch aus der Abbildung 2, da hier offensichtlich kein separates Model implementiert wurde. Ein solches würde, falls benötigt, in einem separatem Model-Ordner, direkt unter dem Wurzelverzeichnis implementiert werden. Diese

[6] o. V., MVVM, 2017
[7] o. V., Razor Pages, 2019

sind, genau wie die PageModels, C# Dateien. PageModels vereinen damit eine breite Menge an Funktionalitäten, wie beispielsweise HTTPContext, den ModelState oder die Behandlung von Requests und Responses, wie z. b. HTTP-Anfragen, die in MVC getrennt würden.[8] Der Startpunkt der Website bildet die Index.cshtml. Eine weitere wichtige Page-Datei ist die _Layout.cshtml, welche ein Design-Template für jede andere Page darstellt. In ihr kann z. B. der Header der Website für alle Razor Pages festgelegt werden. Insgesamt erfüllen Pages mit einem „_“ als Präfix jeweils eine besondere Aufgabe, sind allerdings nicht über einen URL-Aufruf erreichbar. Der Aufruf von „https://Hostname/_Layout“ z. B. würde daher zu einem 404 Http Error führen. Offensichtlich werden Razor Pages, anders als bei ASP.Net MVC, Dateien nach dem Zweck gegliedert. Es gibt nicht nur einen Controller der die gesamte Geschäftslogik in sich vereint, sondern stattdessen gibt es viele C# Dateien, PageModel darstellen und jeweils einer View zugeordnet sind.

Abbildung 2: Aufbau Razor Pages




In wwwroot befinden sich statische Dateien, wie CSS-Style-Sheets, Bilder oder JavaScript(JS)-Dateien. Im lib Ordner wiederum befinden sich Drittanbieter-Pakete. Defaultmäßig sind dies JQuery und Bootstrap, was im Verlauf der Arbeit noch erläutert wird.

Darüber hinaus befinden sich direkt in dem Wurzelverzeichnis eine Konfigurationsdatei im JSON-Format für die gesamte Anwendung namens „appsettings.json“, eine „Program.cs“, die den Startpunkt der Anwendung darstellt und nach eine „Startup.cs“. In letzterer können z. B. benötigte Services hinzugefügt und weitere Konfigurationen vorgenommen werden.[9] Folgendes Beispiel zeigt, wie der Razor Page Service in eine ASP.Net Webanwendung integriert werden können:

```
1  public void ConfigureServices(IServiceCollection services)
2  {
3      services.AddRazorPages();
```

[8] Jones, Matthew, Razor vs. MVC, 2019
[9] o. V., Aufbau Razor Pages, 2020

```
}
```

### 2.2.3 Aufbau einer Razor Page

Eine Razor Page zeichnet sich, wie bereits erwähnt, vor allem auch durch die Razor Syntax aus, die in den HTML-Code eingefügt wird. Der Server erkennt sie durch ein vorangestelltes „@“. So kann beispielsweise ganzer C#-Code einfach in den HTML-Code eingefügt werden und ihn dadurch mit Funktinalität ausstatten. Dies geschieht innerhalb eines Codeblocks: `@{ C#-Code }`. Darüber hinaus können auch einzelne Funktionen, wie eine if-Funktion, oder auch eine Schleife mit einem vorangestellten @ benutzt werden: `@if ( Condition ) { C#-Code }`. Es ist auch möglich von der View aus auf bestimmte Variablen des PageModels zuzugreifen. Dies geschieht über `@Model.variable`. Um auf diese Variable zugreifen zu können muss sie allerfings als Public deklariert werden. Eine weitere Möglichkeit, Daten vom PageModel an die View zurückzugeben ist ViewData, wobei es sich um ein dictionary verschiedener Objekte handelt. Dieses dictionary wird automatisch an die View übergeben. Somit kann jederzeit auf die darin enthaltenen Objekte, über den jeweiligen Key, zugegriffen werden. ViewData Attribute werden wie folgt im Page Model definiert und anschließend über `@ViewData["Key"]` aufgerufen.[10]

```
public class IndexModel : PageModel
{
    [ViewData]
    public string Key { get; set; }
    // Oder alternativ
    ViewData["Key"] = "Value"
}
```

Außerdem existieren verschiedene Tag-Helpers, also wiederverwendbarer HTML-Code für die Vereinfachung von ASP Funktionalitäten. Ein Beispiel hierfür wäre der Validation Message Tag Helper `asp-validation-for`, der einem span-Element eine Validation Error Message zuordnet, indem er ihm das HTML-Element `data-valmsg-for` zuweist. Bei einem Client seitigen Validationsfehler zeigt jQuery die Fehlermeldung innerhalb des spans. Der Tag kann allerdings auch für eine serverseitige Validation verwendet werden. Diese könnte beispielsweise so aussehen:

```
<input type="text" asp-for="IBAN" />
<span asp-validation-for="IBAN"> </span>
```

```
public class IndexModel : PageModel
```

[10] o. V., ViewData, o. D.

```
{
    [BindProperty]
    [Required]
    public string IBAN { get; set; }
}
```

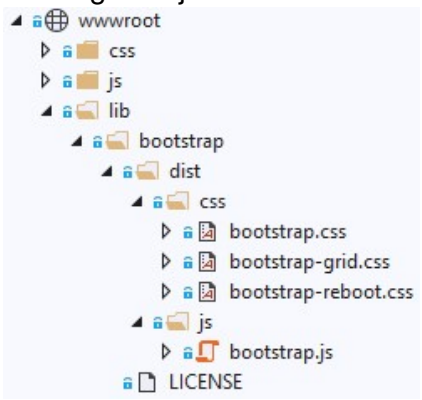
Wurde nun keine IBAN, bei einem Request an den Server, eingegeben wird unterhalb des Textfeldes eine Fehlermeldung ausgegeben. Ein weiteres Beispiel wäre der `asp-page-handler`, mit dem ein spezieller Page Handler ausgeführt werden kann. Bei Handler Methoden handelt es sich um Funktionen, die automatisch bei einem entsprechenden HTTP-Request ausgeführt werden. Die Handler Methode `OnGet()` wird beispielsweise bei einer Get-Anfrage an den Server ausgeführt. Mit dem `asp-page-handler` können nun verschiedene Page Handler implementiert werden. So würde ein `OnPostProcessing()` Handler nur bei einem bestätigen von `<button type=,,submit‘‘ asp-page-handler=,,Processing‘‘>Submit</button>` ausgeführt werden.[11] [12]

Jede Page beginnt stehts mit `@page` in der ersten Zeile. Hierdurch weiß ASP.Net, dass es diese datei wie eine Razor Page behandeln muss. Hierdurch kann die Page selber Aktionen ausführen und ist nicht auf einen Controller angewiesen. Des Weiteren muss ein `@model` angegeben werden, mit dem dazugehörigen Model dieser View. Bei Razor Pages ist dies in der Regel (i. d. R.) das PageModel.[13]

## 2.3 Bootstrap

Bei Bootstrap handelt es sich um ein Frontend-Framework zur optischen Gestaltung einer Website. Es dient primär der schnellen und einfachen Umsetzung eines Responsive Web-Designs, mit dem Websites für jede Displaygröße optimal gestaltet sein sollen. Damit verfolgt Bootstrap stark die Mobile-first Philosophie. Ursprünglich handelte es sich bei Bootstrap um eine Technologie von Twitter, die dann als Open-Source-Projekt veröffentlicht wurde. Wenn man sich seine Bestandteile anschaut, dann basiert es überwiegend auf CSS, aber auch auf HTML und JS. Seine Bestandteile sind zum

Abbildung 3: Bootstrap in Razor Page Projekt mit VS




[11] o. V., Page Handler, 2018
[12] Anderson, Rick; Mullen, Taylor; Vicarel, Dan, Razor Syntax, 2020
[13] Jones, Matthew, Razor vs. MVC, 2019

einen das Design von Basis-HTML-Elementen und
JS Plugins, welche zumeist auf jQuery basieren, bei dem es sich um eine freie JS-Bibliothek handelt. Darüber hinaus gibt es noch verschiedene Komponenten, bei denen es sich um von Bootstrap vordefinierte CSS-Klassen zur Gestaltung von HTML-Elementen handelt. Bootstrap ist zudem leicht anpassbar, sollten die vordefinierten Gestaltungsmöglichkeiten nicht den eigenen Wünschen entsprechen, wie wenn man beispielsweise ein neues Farblayout implementieren möchte.[14]

Bootstrap eignet sich besonders gut für die ASP.Net Einbindung, denn bei der Erstellung eines Standard Razor Page Projektes mit Visual Studio (VS) ist Bootstrap bereits vorinstalliert und kann direkt verwendet werden (siehe Abb. 3).

[14] Bhaumik, Snig, Bootstrap, 2015, S. 7-11

# 3 Praxisteil

## 3.1 Anforderungsanalyse

## 3.2 Dokumentation

### 3.2.1 Grundfunktionalität

### 3.2.2 Erweiterungen

### 3.2.3 Graphische Gestaltung

# Literatur


Beasley, Robert E., [.Net Basics] Essential ASP.NET Web Forms Development, Berkeley, CA 2020

Augsten, Stephan, [.Net Core] Definition „Microsoft .NET Core Platform“, Was ist .NET Core?, 09.04.2020, https://www.dev-insider.de/was-ist-net-core-a-914978/ (24.05.2020)

Rouse, Margaret, [MVC] Model View Controller (MVC), 10.2016, https://www.computerweekly.com/de/definition/Model-View-Controller-MVC (26.05.2020)

Gutsch, Jürgen, [ASP.Net] Microsofts Web-Frameworks im Vergleich, Die Qual der Wahl, 15.06.2017, https://www.dotnetpro.de/frontend/qual-wahl-1226135.html (26.05.2020)

Augsten, Stephan, [C#] Definition „C-Sharp“, Was ist C#?, 09.08.2019, https://www.dev-insider.de/was-ist-c-a-846162/ (26.05.2020)

o. V., [MVVM] The Model-View-ViewModel Pattern, 07.08.2017, https://docs.microsoft.com/de-de/xamarin/xamarin-forms/enterprise-application-patterns/mvvm (26.05.2020)

o. V., [Razor Pages] Welcome To Learn Razor Pages, 08.05.2019, https://www.learnrazorpages.com/ (26.05.2020)

o. V. [Aufbau Razor Pages] First look at Razor Pages, 08.01.2020 https://www.learnrazorpages.com/first-look (26.05.2020)

Jones, Matthew, [Razor vs. MVC] How Does Razor Pages Differ From MVC In ASP.NET Core?, 04.03.2019, https://exceptionnotfound.net/razor-pages-how-does-it-differ-from-mvc-in-asp-net-core/ (26.05.2020)

Anderson, Rick; Mullen, Taylor; Vicarel, Dan, [Razor Syntax] Razor syntax reference for ASP.NET Core, 12.02.2020, https://docs.microsoft.com/de-de/aspnet/core/mvc/views/razor?view=aspnetcore-3.1 (26.05.2020)

o. V., [ViewData] Working With ViewData in Razor Pages, o. D., https://www.learnrazorpages.com/razor-pages/viewdata (26.05.2020)

o. V., [Page Handler] Handler Methods in Razor Pages, 24.10.2018. https://www.learnrazorpages.com/razor-pages/handler-methods (26.05.2020)

# Selbstständigkeitserklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit mit dem Thema: „Portfolio Nr. 6 - Überprüfung und Verarbeitung eines String mit ASP.Net“ selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

**Ort, Datum** **Unterschrift**

Nürnberg, den 30.05.2020