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

Abbildungsverzeichnis

Tabellenverzeichnis

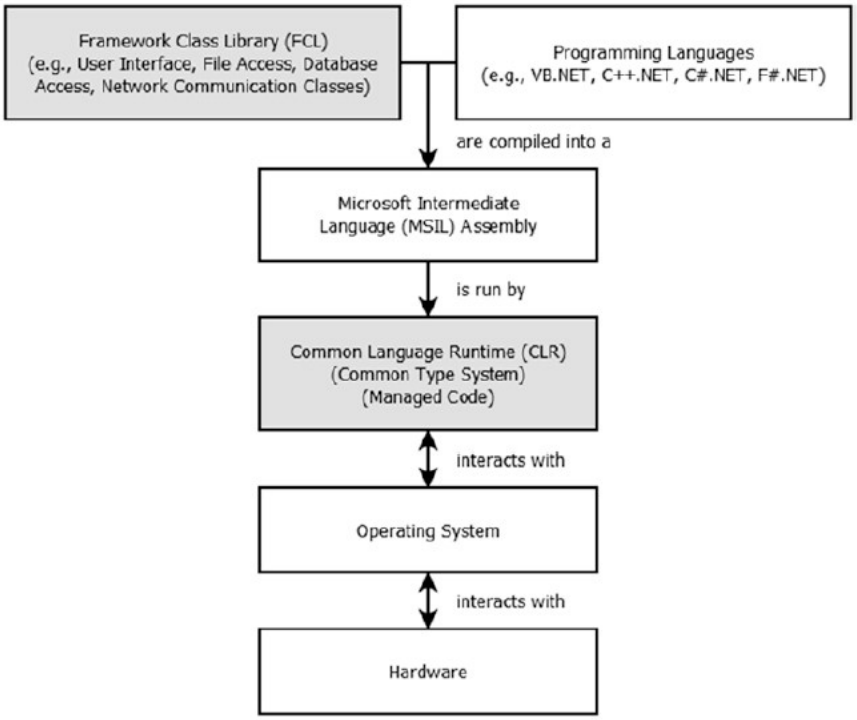
1 Einleitung

2 Theoretische Grundlagen

2.1 ASP.Net

Bei ASP.Net handelt es sich um ein 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

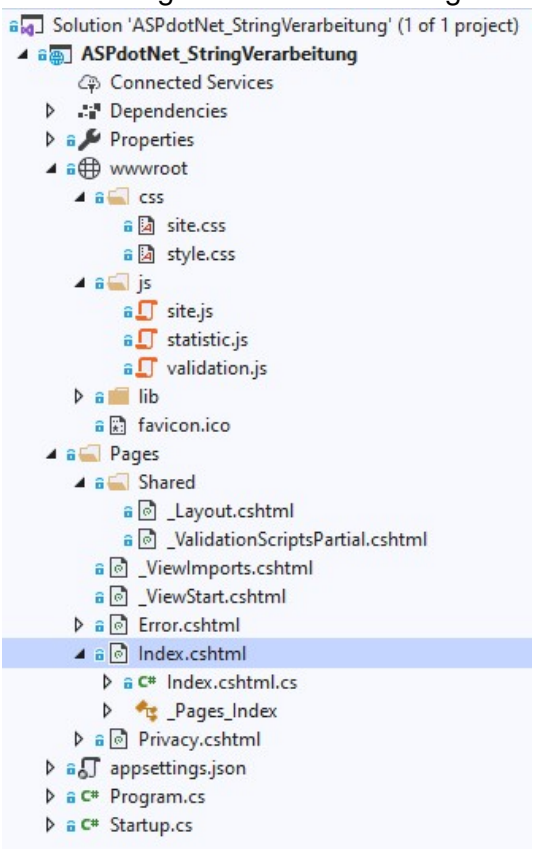
Razor Pages basieren auf ASP.Net MVC und zeichnen sich durch die Razor Syntax aus. Mit dieser können statische HTML Webseiten mit C# dynamisch gemacht werden. Dies zeichnet sich dadurch aus, dass eine Webseite durch eine .cshtml Datei erstellt wird, also eine Kombination aus C#, mittels der Razor Syntax und HTML. 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 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] Nichts desto trotz kann auch in einer Razor Page mit der MVC-Architektur gearbeitet werden falls dies gewünscht wird. [7]

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

2.2.2 Aufbau einer Razor Page

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, welche zusammen die eigentlichen Razor Pages darstellen. Die .cshtml Dateien sind die Views der Razor Page, allerdings könnte mit Hilfe der Razor Syntax 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. 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 Projekt implementiert werden. Diese sind, genau wie die PageModels, C# Dateien. Der Startpunkt der Website bildet die Index.cshtml. Eine weitere wichtige Page-Datei ist die _Layout.cshtml, welche ein Content Template für jede andere Page darstellt. In ihr kann z. B. der Header der Website für alle Razor Pages festgelegt werden.

Abbildung 2: Aufbau Razor Pages



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, die das ViewModel bzw. auch PageModel genannt darstellen und jeweils einer View zugeordnet sind.

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.

2.3 Bootstrap

3 Praxisteil: Dokumentation der Webanwendung

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 https://www.learnrazorpages.com/ (26.05.2020)

Selbstständigkeitserklärung