

Dr. Holger Schwichtenberg

Blazor 6.0: Blazor WebAssembly, Blazor Server und Blazor Desktop

**Moderne Single-Page-Web- und hybride Cross-
Platform-Apps mit .NET, C# und Visual Studio**



Buchversion:	6.0.0 (16.09.2021)
Verlag:	www.IT-Visions.de , Fahrenberg 40b, D-45257 Essen
Sprachkorrektur:	Matthias Bloch, M.A.
ISBN:	978-3-934279-35-3
Bezugsquelle gedruckt:	www.amazon.de/exec/obidos/ASIN/3934279368/itvisions-21
Bezugsquelle Kindle:	www.amazon.de/exec/obidos/ASIN/B08J5L3WQ7/itvisions-21
Bezugsquelle PDF:	www.leanpub.com/Blazor60

Für Heidi, Felix und Maja

1 Inhaltsverzeichnis (Hauptkapitel)

1	Inhaltsverzeichnis (Hauptkapitel).....	4
2	Inhaltsverzeichnis (Details).....	6
3	Vorwort	19
4	Über den Autor.....	21
5	Über dieses Buch.....	22
6	Programmcodebeispiel zum Download.....	27
7	Was ist Blazor?	45
8	Projektaufbau	93
9	Übersetzung und Debugging	115
10	Komponenten (Razor Components).....	128
11	Routing und Navigation	148
12	Razor-Syntax.....	161
13	Ereignisse und Ereignisbehandlung	170
14	Komponenteneinbettung / Unterkomponenten.....	180
15	Zustandsverwaltung	209
16	Formulare/Eingabemasken.....	225
17	Klassenbibliotheken und Razor Class Libraries (RCL).....	263
18	Dependency Injection (DI).....	279
19	Interoperabilität zwischen .NET und JavaScript	288
20	Zugriff auf Webservices/WebAPIs	321
21	Benachrichtigungen mit ASP.NET Core SignalR in Blazor	341
22	Serversystem- und Browserinformationen	350
23	Authentifizierung und Benutzerverwaltung	360
24	Installation von Blazor-Anwendungen.....	389
25	Tipps, Tricks und Tools	398
26	Hybride Apps mit Blazor Desktop	477
27	Fallbeispiel "MiracleList"	493
28	Umstieg von ASP.NET Webforms zu ASP.NET Core Blazor	579
29	Mobile Apps mit Mobile Blazor Bindings	615
30	Quellen im Internet	623
31	Versionsgeschichte dieses Buchs	624
32	Stichwortverzeichnis (Index)	633
33	Werbung in eigener Sache ☺	644

2 Inhaltsverzeichnis (Details)

1	Inhaltsverzeichnis (Hauptkapitel).....	4
2	Inhaltsverzeichnis (Details).....	6
3	Vorwort	19
4	Über den Autor.....	21
5	Über dieses Buch.....	22
5.1	Versionsgeschichte dieses Buchs	22
5.2	Bezugsquelle für Aktualisierungen	22
5.3	Geplante Kapitel.....	22
5.4	Programmiersprache in diesem Buch.....	23
5.5	Notwendige Vorkenntnisse	23
5.6	Hinweis zur Erwähnung der Blazor-Varianten in diesem Buch.....	25
5.7	Ihre Belohnung, wenn Sie helfen, dieses Buch zu verbessern!	25
6	Programmcodebeispiel zum Download.....	27
6.1	Webadresse für Downloads.....	27
6.2	Übersicht über die Beispiele.....	27
6.3	Eingesetztes CSS-Framework	29
6.4	Das Fallbeispiel "MiracleList"	29
6.4.1	Szenario	29
6.4.2	Technische Basis für MiracleList	30
6.4.3	Webadressen.....	30
6.4.4	Projektmappenaufbau	30
6.4.5	MiracleList-Bildschirmmasken	31
6.5	Weitere Beispiele innerhalb der MiracleList-Webanwendung.....	36
6.6	Visual Studio 2022	37
6.7	.NET SDK	39
6.8	Testen Ihrer PC-Konfiguration.....	41
6.9	Util.Log()	43
6.10	Qualitätssicherung der Programmcodebeispiele.....	43
7	Was ist Blazor?	45
7.1	Blazor-Arten.....	45
7.2	Geschichte von Blazor.....	47
7.3	Technischer Support für Blazor.....	48
7.3.1	Support für Blazor Server.....	48

7.3.2	Support für Blazor WebAssembly	49
7.4	Blazor Server	50
7.4.1	Rendering und Interaktion bei Blazor Server	50
7.4.2	Softwarearchitektur	51
7.4.3	Netzwerkverkehr bei Blazor Server	51
7.4.4	Ressourcenbedarf und Skalierbarkeit	52
7.4.5	Vor- und Nachteile von Blazor Server	54
7.5	Blazor WebAssembly	55
7.5.1	Konzept von Blazor WebAssembly	55
7.5.2	WebAssembly (WASM)	56
7.5.3	Sandbox des Browsers / API-Einschränkungen	57
7.6	Blazor Desktop	59
7.6.1	Hybride Apps	59
7.6.2	Architektur von Blazor Desktop	60
7.7	NuGet-Pakete für Blazor	61
7.8	Neuerungen in Blazor 5.0	62
7.9	Neuerungen in Blazor 6.0	64
7.9.1	Blazor Desktop	65
7.9.2	Verbesserungen beim Komponentenmodell	65
7.9.3	Verbesserungen bei den Eingabesteuerelementen	66
7.9.4	Verbesserte Interoperabilität zwischen .NET und JavaScript	66
7.9.5	Verbesserte Werkzeuge	66
7.9.6	Verbesserungen beim Kompilieren und Deployment	66
7.9.7	Weitere Verbesserungen in Blazor 6.0	68
7.10	Fehlende Funktionen in Blazor 6.0	69
7.11	Vergleich zwischen Blazor WebAssembly, Blazor Server und Blazor Desktop	69
7.11.1	Die wichtigsten Unterschiede	69
7.11.2	Browserunterstützung für Blazor	76
7.11.3	Performance-Vergleich	78
7.11.4	Anwendungsgröße und Startzeiten	83
7.11.5	Speicherlimit	85
7.11.6	Fazit	87
7.12	Blazor im Vergleich zu anderen Spielarten von ASP.NET Core	88
7.13	Blazor im Vergleich zu anderen Webframeworks	89
7.14	Uno Platform	91

8	Projektaufbau	93
8.1	Voraussetzungen für Blazor	93
8.2	Die Webanwendung in der Standard-Projektvorlage	93
8.3	Blazor-Projekte anlegen	94
8.4	Aufbau eines Blazor Server-Projekts	95
8.4.1	Ordner und Dateien	95
8.4.2	Projektdatei.....	97
8.4.3	Start-Code Program.cs.....	98
8.4.4	Genauere Fehlermeldungen aktivieren (Detailed Errors)	99
8.4.5	Struktur des Seitenaufbaus	100
8.5	Aufbau eines Blazor WebAssembly-Projekts	102
8.5.1	Ordner und Dateien	107
8.5.2	Aufbau der index.html.....	107
8.5.3	Projektdatei.....	107
8.5.4	Start-Code im Client-Projekt.....	109
8.5.5	Start-Code im Server-Projekt	109
8.5.6	Struktur des Seitenaufbaus	110
8.6	Unterschiede zwischen Blazor Server und Blazor WebAssembly	111
8.7	Anlegen einer kompletten mehrschichtigen Projektstruktur mit der .NET CLI	111
9	Übersetzung und Debugging	115
9.1	Übersetzung.....	115
9.2	Start von Blazor-Projekten in Visual Studio	116
9.3	Hot Reloading (seit .NET 6).....	118
9.4	Visual Studio-Debugging in Blazor Server-Projekten	122
9.5	Visual Studio-Debugging in Blazor WebAssembly-Projekten	122
9.6	Ablaufverfolgung des ASP.NET Core Webservers.....	126
9.7	Browser-Debugging in Blazor WebAssembly	126
9.8	Veröffentlichen von Blazor-Anwendungen.....	127
10	Komponenten (Razor Components).....	128
10.1	Einsatz von Razor Components.....	128
10.2	Namen für Komponenten	128
10.3	Grundkonzepte am Beispiel einer "Hello World"-Komponente	128
10.4	Markup-Dateien mit Inline-Code	130
10.5	Komponenten mit Code-Behind-Datei.....	132
10.5.1	Code-Behind-Dateien auf Basis von Vererbung	133

10.5.2	Code-Behind-Dateien auf Basis von partiellen Klassen.....	136
10.6	Komponenten ohne Markup nur mit Programmcode	138
10.7	CSS-Isolation (ab Blazor 5.0)	141
10.7.1	Scoped Styles anlegen	141
10.7.2	Scoped Styles einbinden.....	141
10.7.3	CSS-Isolation mit Razor Class Libraries.....	142
10.8	Nicht-visuelle Komponenten.....	143
10.9	Layoutseiten (Masterpages)	146
11	Routing und Navigation	148
11.1	Routendefinitionen.....	148
11.2	Routen mit Parametern.....	149
11.2.1	Navigation per <a>-Tag	151
11.2.2	URL-Fragmente	151
11.2.3	Komponente <NavLink>.....	151
11.2.4	Navigation im Code.....	152
11.3	Auswerten der aktuellen URL	153
11.3.1	Fallunterscheidung bei Mehrfachrouten	154
11.3.2	Query Strings.....	154
11.3.3	Komplette URL-Auswertung	155
11.3.4	Routerkonfiguration	159
12	Razor-Syntax.....	161
12.1	Unterschiede zwischen Razor in Blazor und Razor in MVC und Razor Pages.....	161
12.2	Ausdrücken vs. Befehlsfolgen.....	165
12.3	Asynchrone Ausdrücke in der Razor-Syntax	166
12.4	Praxisaufgabe zur Razor-Syntax	167
12.4.1	Aufgabenstellung.....	167
12.4.2	Lösung	168
13	Ereignisse und Ereignisbehandlung	170
13.1	Komponentenlebenszyklusereignisse	170
13.2	Reaktion auf HTML-Ereignisse	171
13.3	Eigene Parameter bei HTML-Ereignissen.....	172
13.4	Standardparameter.....	173
13.5	Standardereignisbehandlung unterbinden.....	176
13.6	Ereignisweitergabe unterbinden	177
13.7	Komplexe Befehlsfolgen als Ereignisbehandlung.....	179

14 Komponenteneinbettung / Unterkomponenten.....	180
14.1 Einbetten von Razor Components	180
14.2 Parameter für eingebettete Razor Components	181
14.2.1 Parameter mit Value Type.....	181
14.2.2 Parameter mit Reference Type	185
14.3 Komponentenparameter aus Querystring (seit .NET 6)	187
14.4 Kaskadierende Parameter	189
14.5 Pflichtparameter für Komponenten (seit .NET 6)	193
14.6 Vorlagenbasierte Komponenten (Templated Components)	193
14.6.1 Einfache vorlagenbasierte Komponente	194
14.6.2 Vorlagenbasierte Komponente mit mehreren Fragmenten	195
14.6.3 Datenübergabe an Renderfragment via Kontext.....	196
14.6.4 Komplexe Objekte als Kontext	197
14.6.5 Vorlagenbasierte Komponente mit generischem Typparameter.....	198
14.6.6 Verschachtelung von Renderfragmenten.....	199
14.6.7 Standardwerte für Renderfragmente.....	201
14.6.8 Praxislösung: Repeater	202
14.7 Dynamic Component (seit .NET 6).....	204
14.8 Error Boundaries (seit .NET 6)	206
15 Zustandsverwaltung	209
15.1 Komponentenzustand	209
15.2 Testanwendung: Counter.....	209
15.3 Verlust des Zustandes.....	210
15.4 Bewahrung des Zustandes	212
15.5 Sitzungszustand.....	212
15.6 Generischer Sitzungszustand.....	213
15.7 Nutzung des Webbrowserspeichers.....	216
15.7.1 Verschlüsselter Browserspeicher (Protected Browser Storage) für Blazor Server	
216	
15.7.2 NuGet-Pakete die Nutzung des Browserspeichers in Blazor WebAssembly und	
Blazor Desktop.....	218
15.7.3 Einsatz von Blazored.LocalStorage.....	219
15.8 Cookies.....	222
15.9 Persistierung in Datenbanken oder anderen persistenten Speichern auf dem Server ...	224
16 Formulare/Eingabemasken.....	225
16.1 Formulare auf Basis der Blazor-Eingabekomponenten.....	225

16.1.1	Verfügbare Blazor-Eingabekomponenten	225
16.1.2	Komponente <EditForm>.....	226
16.1.3	Datenbindung	227
16.1.4	Datenannotationen.....	227
16.1.5	Validierungskomponenten in Blazor	229
16.1.6	Eigene Validierungsannotationen.....	229
16.1.7	Praxisbeispiel	230
16.1.8	Optionsfelder (<InputRadio> und <InputRadioGroup>)	236
16.1.9	Dateien hochladen (<InputFile>)	238
16.1.10	Praxisbeispiel: Dateien-Upload bei Blazor Server	239
16.2	Formulare auf Basis von Standard-HTML-Tags.....	245
16.2.1	Datenbindung	245
16.2.2	Reaktion auf einzelne Buchstabeneingaben	245
16.2.3	Auswahlfelder (<select>)	250
16.2.4	Datenbindung mit Datumsangaben	253
16.2.5	Praxisbeispiel	255
16.3	Tipps & Tricks zu Formularen	260
16.3.1	Direkter Objektverweis	260
16.3.2	Fokus setzen	260
17	Klassenbibliotheken und Razor Class Libraries (RCL).....	263
17.1	.NET Standard.....	263
17.2	Referenzierbarkeit	264
17.3	Anlegen der Klassenbibliotheken.....	265
17.4	Einsatzgebiete der verschiedenen Klassenbibliotheksarten.....	267
17.5	Razor Class Libraries (RCL)	267
17.5.1	Erstellen einer Razor Class Library.....	267
17.5.2	Inhalte einer Razor Class Library	269
17.5.3	Referenzierung einer Razor Class Library	269
17.5.4	Einbettung von Razor Components aus einer Razor Class Library.....	271
17.5.5	Routing zu Razor Components aus einer Razor Class Library	271
17.5.6	Nutzung von statischen Ressourcen aus einer Razor Class Library	272
17.6	Lazy Loading von DLLs (ab Blazor 5.0)	273
17.6.1	Integration von Lazy Loading in die Standardprojektvorlage	274
17.6.2	Integration von Lazy Loading in das MiracleList-Fallbeispiel.....	276
17.7	Code-Sharing zwischen Blazor Server, Blazor WebAssembly und Blazor Desktop ...	277

18	Dependency Injection (DI).....	279
18.1	Microsoft.Extensions.DependencyInjection.....	279
18.2	Implementierung eines Dienstes.....	279
18.3	Lebensdauer von Instanzen	280
18.4	Registrierung von Diensten	281
18.5	Automatisch registrierte Dienste	282
18.6	Injektion im Template mit @inject	285
18.7	Injektion in Properties mit [Inject]	285
18.8	Injektion in einen Konstruktor (Konstruktorinjection).....	285
18.9	Manuelle Beschaffung von Instanzen	286
19	Interoperabilität zwischen .NET und JavaScript	288
19.1	Motivation	288
19.2	Einschränkung der JavaScript-Interoperabilität in Blazor Server	288
19.3	Aufruf von C# zu JavaScript	289
19.4	Eigene JavaScript-Dateien verwenden	290
19.4.1	Globale Skripteinbindung.....	290
19.4.2	JavaScript-Isolation / IJSObjectReference (ab Blazor 5.0)	293
19.5	Aufruf von JavaScript zu C#	294
19.5.1	Aufruf von JavaScript zu C# über statische Methoden	294
19.5.2	Praxisbeispiel	295
19.5.3	Aufruf von JavaScript zu C# (Instanzmethoden)	298
19.6	Praxisbeispiel: Nutzung einer vorhandenen JavaScript-Bibliothek am Beispiel Chartist.js 299	
19.7	Praxisbeispiel: Integration mit Angular über Web Components	303
19.7.1	Implementierung einer Web Component mit Angular Elements.....	304
19.7.2	Nutzung der Web Component in JavaScript	306
19.7.3	Einbindung von Web Components in Blazor	307
19.8	Praxisbeispiel: Angular-Datagrid in Blazor	310
19.8.1	Implementierung der Web Component in Angular	310
19.8.2	Nutzung der Web Component in JavaScript	313
19.8.3	Einbindung der Web Component <angular-grid> in Blazor	313
19.9	Streaming zwischen .NET und JavaScript (seit .NET 6).....	318
19.10	JavaScript-Initializer (seit .NET 6).....	320
20	Zugriff auf Webservices/WebAPIs	321
20.1	Daten- und Ressourcenzugriffe in Blazor WebAssembly	321
20.2	Daten- und Ressourcenzugriffe in Blazor Server und Blazor Desktop	322

20.3	Webservice-Arten.....	322
20.4	Szenario.....	323
20.5	REST-Dienste (WebAPIs)	324
20.5.1	WebAPIs erstellen mit ASP.NET Core	324
20.5.2	WebAPIs testen mit Postman	325
20.5.3	Blazor-Zugriff auf Web APIs.....	326
20.5.4	Cross-Origin Resource Sharing für das WebAPI	328
20.5.5	Metadaten mit Open API Specification.....	329
20.5.6	Client-Generierung aus Metadaten mit Visual Studio	330
20.5.7	Client-Generierung aus Metadaten mit NSwagStudio	332
20.6	Google RPC-Dienste	333
20.6.1	gRPC-Dienst erstellen	333
20.6.2	Blazor-Zugriff auf gRPC-Dienste	337
20.7	Leistungsvergleich zwischen WebAPI und gRPC	338
20.8	SOAP und die Windows Communication Foundation (WCF).....	339
21	Benachrichtigungen mit ASP.NET Core SignalR in Blazor	341
21.1	Implementierung im Webserver.....	342
21.2	Implementierung im Blazor-Client.....	343
21.3	Aktivieren der Websockets-Unterstützung.....	347
21.4	Diagnose von ASP.NET Core SignalR	349
22	Serversystem- und Browserinformationen	350
22.1	HttpContext.....	350
22.2	Allgemeine Systeminformationen	351
22.3	Host Environment-Informationen	356
22.4	Verwenden von Environmentnamen	357
23	Authentifizierung und Benutzerverwaltung	360
23.1	Verwenden von ASP.NET Core Identity in Blazor Server	360
23.1.1	Aktivieren der Authentifizierung in Blazor Server	360
23.1.2	Individuelle Benutzerkontenverwaltung.....	361
23.1.3	Anpassung der Benutzerverwaltungsseiten	363
23.1.4	Weitere Anpassung von ASP.NET Identity	366
23.2	Authentifizierung in Blazor WebAssembly	366
23.2.1	OIDC-Authentifizierung bei Projekten ohne "ASP.NET Core Hosted".....	367
23.2.2	Weitere OIDC-Optionen	371
23.2.3	Authentifizierung bei Projekten mit "ASP.NET Core Hosted"	371

23.3	Eigene Authentifizierungsprovider (AuthenticationStateProvider).....	375
23.3.1	AuthenticationStateProvider für Debugging-Zwecke	376
23.3.2	AuthenticationStateProvider in MiracleList	376
23.3.3	Nutzung des eigenen AuthenticationStateProvider	383
23.4	Autorisierung.....	384
23.4.1	Zugriff auf den angemeldeten Benutzer	385
23.4.2	Autorisierung von Komponenten	386
23.4.3	Inhalte für angemeldete Benutzer.....	386
23.4.4	Umleitung für nicht-autorisierte Benutzer.....	386
24	Installation von Blazor-Anwendungen.....	389
24.1	Deployment	389
24.2	Installation auf Azure-Diensten.....	390
24.2.1	WebSockets	390
24.2.2	Brotli-Komprimierung.....	391
24.3	Installationsvoraussetzungen für eigene Windows-Webserver (IIS).....	392
24.4	Installation auf einem eigenen Windows-Webserver (IIS).....	393
24.5	Self-Hosting (Kestrel Webserver)	397
25	Tipps, Tricks und Tools	398
25.1	Ermitteln der Blazor-Art und -Versionsnummer	398
25.2	Ausgaben in die Browerkonsole	400
25.3	HTML-Kopf beeinflussen	404
25.3.1	Lösung in Blazor 5.0	404
25.3.2	Lösung In Blazor 6.0	405
25.4	BlazorFiddle	406
25.5	Render-Modi bei Blazor Server	407
25.5.1	Server Prerendered	407
25.5.2	Server	408
25.5.3	Static	409
25.5.4	Static mit Parametern	409
25.5.5	Mehrere Blazor Server-Komponenten in einer Seite.....	410
25.6	Render-Modi bei Blazor WebAssembly.....	410
25.7	Ahead-of-Time-Kompilierung (AOT) (seit .NET 6).....	412
25.8	Anwendungskonfigurationsdateien	413
25.8.1	AppSettings.json	413
25.8.2	Listen in Konfigurationseinträgen	415

25.8.3	Andere Konfigurationsformate.....	415
25.9	Steuerung des Rendering.....	416
25.10	Leistungsoptimierung beim Rendern großer Datenmenge durch Virtualisierung	420
25.10.1	Szenario	422
25.10.2	Suksessives Rendern mit <Virtualize>	423
25.10.3	Suksessives Laden per ItemsProvider-Funktion.....	425
25.11	Überwachung von Blazor Server-Anwendungen	426
25.11.1	Überwachung der Netzwerklatenz bei Blazor Server.....	426
25.11.2	Überwachung der Sitzungen/Circuits bei Blazor Server	428
25.12	Hintergrundaufgaben und Fortschrittsanzeige.....	433
25.13	Zeitgesteuerte Aufgaben via Timer	436
25.13.1	Praxisbeispiel: Zeitgesteuerte Seitenaktualisierung.....	437
25.13.2	Praxisbeispiel: Countdown.....	438
25.13.3	Eigene Timer-Komponente für Blazor	439
25.14	Meldungskomponente	440
25.15	Modale Dialoge mit Bootstrap	445
25.16	Benachrichtigungen.....	448
25.16.1	Desktop-Benachrichtigungen	448
25.16.2	Benachrichtigungen im Browserfenster (Toasts)	451
25.17	Zugriff auf die lokale Zeit des Webbrowsers	452
25.18	Custom Boot Resource Loading.....	452
25.19	Progressive Web Applications (PWA) mit Blazor WebAssembly	453
25.20	Verzögerter Start einer Blazor WebAssembly-Anwendung.....	458
25.21	Integration von Blazor-Anwendungen in andere Webanwendungen	459
25.22	Mehrsprachigkeit (Lokalisierung/Globalisierung/ Internationalisierung)	460
25.22.1	Festlegung der aktuellen Sprache	460
25.22.2	Texte aus Ressourcendateien.....	460
25.22.3	Beispiel.....	460
25.22.4	Notwendige Infrastruktur bei Blazor WebAssembly.....	464
25.22.5	Notwendige Infrastruktur bei Blazor Server.....	465
25.22.6	Lokalisierung bei Blazor Desktop	467
25.22.7	Darstellung von Datums- und Zahleingabefeldern.....	468
25.23	Optimierung der Download-Größe bei Blazor WebAssembly	469
25.24	Micro-Apps mit Blazor	470
25.24.1	Projektaufbau.....	471

25.24.2	Kommunikation.....	473
26	Hybride Apps mit Blazor Desktop	477
26.1	Blazor Desktop auf Windows mit WPF und Windows Forms	477
26.2	Beispiel: Blazor-Anwendung in Windows Forms.....	478
26.3	Implementierungsdetails	481
26.4	Integration in WPF	489
26.5	Verteilungsoptionen	489
26.6	Blazor in Cross-Platform-Apps mit .NET MAUI.....	489
27	Fallbeispiel "MiracleList"	493
27.1	Das MiracleList-Backend.....	493
27.1.1	Softwarearchitektur	496
27.1.2	Entitätsklassen/Geschäftsobjekte/Datentransferobjekte	499
27.1.3	Entity Framework Core-Kontextklasse	501
27.1.4	Lebensdauer der Kontextklasse in ASP.NET Core-Anwendungen.....	502
27.1.5	Geschäftslogik.....	503
27.1.6	Web API.....	511
27.2	MiracleList-Frontend mit Blazor Server	521
27.2.1	Softwarearchitektur	521
27.2.2	Projektdatei.....	524
27.2.3	Startcode (Program.cs und Startup.cs)	525
27.2.4	Startseite (_Host.cshtml)	528
27.2.5	Anmeldeansicht (Login.razor).....	529
27.2.6	Authentication State Provider (MLAuthenticationStateProvider)	533
27.2.7	Hauptansicht (Index.razor).....	536
27.2.8	Bearbeitungsformular (TaskEdit.razor).....	545
27.3	MiracleList-Frontend mit Blazor WebAssembly	548
27.3.1	Softwarearchitektur	548
27.3.2	Projektdatei.....	550
27.3.3	Startseite mit Ladeanimation (index.html)	552
27.3.4	Blazor WebAssembly-Startcode (Program.cs).....	553
27.3.5	Anmeldeansicht (Login.razor).....	554
27.3.6	Authentication State Provider (MLAuthenticationStateProvider)	558
27.3.7	Hauptansicht (Index.razor).....	561
27.3.8	Bearbeitungsformular (TaskEdit.razor).....	568
27.4	MiracleList-Hybrid-Frontend mit Blazor Desktop in einer WPF-Anwendung	570

27.4.1	Softwarearchitektur	570
27.4.2	Projektdatei.....	571
27.4.3	WPF-Hostfenster	573
27.4.4	AppState	576
27.4.5	App.razor.....	577
27.4.6	Weitere Dateien	578
28	Umstieg von ASP.NET Webforms zu ASP.NET Core Blazor	579
28.1	Migrationspfade und Herausforderungen.....	579
28.2	Grundsätzliche Vorgehensweise bei der Migration.....	581
28.3	Umstieg von Webforms auf Blazor an einem Beispiel	582
28.3.1	Umsetzung der Datentabelle mit Webforms und Blazor	583
28.3.2	Umstellung der Benutzerschnittstellenbeschreibung.....	589
28.3.3	Umstellung der Benutzerschnittstellensteuerung	589
28.4	Einsatz von Telerik UI for Blazor	590
28.4.1	Komponenten in Telerik UI for Blazor	590
28.4.2	Bezugsquelle und Preise.....	591
28.4.3	Installation von Telerik UI for Blazor	592
28.4.4	Erstellen eines Projekts mit Telerik UI for Blazor	594
28.4.5	Integration von Telerik UI for Blazor in bestehende Blazor-Projekte.....	597
28.4.6	Umsetzung der Flugdatentabelle mit Telerik UI for Blazor	598
28.5	Einsatz von DevExpress UI for Blazor	604
28.5.1	Komponenten in DevExpress UI for Blazor.....	604
28.5.2	Bezugsquelle und Preise.....	604
28.5.3	Installation von DevExpress UI for Blazor	605
28.5.4	Erstellen eines Projekts mit DevExpress UI for Blazor.....	606
28.5.5	Umsetzung der Flugdatentabelle mit DevExpress UI for Blazor	608
29	Mobile Apps mit Mobile Blazor Bindings	615
29.1	Architektur der Blazor Mobile Bindings	615
29.2	Versionsgeschichte.....	616
29.3	Projekt anlegen.....	616
29.4	Übersetzung und Debugging für Android	617
29.5	Aufbau der Projektvorlage	619
29.6	Verwenden von HTML-Rendering	620
29.7	Verfügbare Steuerelemente	621
29.8	HTML-renderndes Blazor in Mobile Blazor Bindings.....	622

30	Quellen im Internet	623
31	Versionsgeschichte dieses Buchs	624
32	Stichwortverzeichnis (Index)	633
33	Werbung in eigener Sache ☺	644
33.1	Dienstleistungen	644
33.2	Aktion "Buch für Buchrezension"	645
33.3	Aktion "Buch-Abo"	646

3 Vorwort

Liebe Leserinnen und Leser,

ich entwickle Webanwendungen seit Mitte der 1990er Jahre. Zunächst habe ich damals mit dem Common Gateway Interface (CGI) und Perl, danach mit dem Internet Database Connector (IDC) und Active Server Pages (ASP) sowie Visual Basic Webclasses programmiert. Nach einer Zwischenetappe in der Java-Welt mit Java Server Pages (JSP), Servlets und Applets folgte dann für mich ASP.NET in diversen Ausprägungen (Webforms, AJAX, Dynamic Data und MVC), schließlich Silverlight, danach ganz viel JavaScript/TypeScript (mit jQuery, AngularJS, Angular, vue.js und diversen anderen Bibliotheken und Frameworks). Seit dem Jahr 2016 verwende ich in manchen Softwareentwicklungsprojekten ASP.NET Core und seit 2019 auch Blazor, während andere weiterhin JavaScript-basierte Frameworks verwenden.

Der Grund dieser Vorrede: Ich habe wirklich schon eine Menge Webframeworks gesehen und verspüre dennoch (oder auch gerade deswegen) einige Begeisterung für die "neuste Sau", die durch das Webdorf getrieben wird; diese "Sau" namens Blazor.

Mit .NET 6.0 sind alle drei Blazor-Varianten (Blazor Server, Blazor WebAssembly und Blazor Desktop) in einer stabilen Version mit einem Support für drei Jahre (was Microsoft heutzutage unter "Long-Term-Support" versteht 😊) verfügbar. Dieses Buch behandelt die Version **Blazor 6.0**. Blazor 6.0 umfasst die vierte Version von Blazor Server, die dritte Version von Blazor WebAssembly und die erste Version von Blazor Desktop.

Alle Ausführungen im Buch und den abgedruckten Beispielen gelten – sofern nicht ausdrücklich anders erwähnt – **sowohl für Blazor WebAssembly als auch Blazor Server und Blazor Desktop**. Es gibt nur wenige Unterschiede zwischen drei Architekturen, und auf die wird an entsprechender Stelle im Buch hingewiesen (Falls ich irgendwo einmal eine der drei Varianten in meinem Text vergessen habe sollte, zu erwähnen, fragen Sie gerne bei mir nach).

Mit Blazor konnten wir schon einige SPA-Webanwendungen in **Rekordzeit entwickeln**, denn man ist als Entwickler damit sehr produktiv. Wenn man lange Erfahrung mit den verschiedenen Microsoft-Webframeworks einerseits und JavaScript-basierten Frameworks andererseits hat, ist man mit Blazor wirklich sehr schnell. Ich denke aber auch, dass Entwickler mit weniger Vorerfahrungen in Blazor eine gute Technik finden werden.

In den letzten Jahren haben wir bei www.IT-Visions.de bei unseren Kunden im ganzen deutschsprachigen Raum immer mehr klassische Windows-Desktop-Entwickler, die bisher mit Windows Forms oder WPF gearbeitet haben, auf Webtechniken umgeschult; getrieben von Chefs oder externen Beratern, die eine "Alles ins Web"- und "Cross-Platform"-Strategie verkündet haben. Mangels technischer Alternativen zur Entwicklung von Single-Page-Web-Applications (SPAs) war die Technikentscheidung für den Browser immer JavaScript bzw. TypeScript, mit Angular, React, vue.js oder anderen Bibliotheken. Auf dem Server konnten die Entwickler oft in der .NET-Welt bleiben mit ASP.NET Core WebAPI. Manchmal stand aber auch hier JavaScript via node.js auf dem Plan. Es waren nicht wenige .NET-Entwickler, die auch nach intensiver Motivation und Schulung keinerlei Gefallen an der JavaScript-Welt finden konnten.

Blazor ist die **Hoffnung für alle JavaScript-Hasser** – wenngleich man hier zumindest in der ersten Zeit noch nicht ohne etwas JavaScript auskommen wird. Aber JavaScript ist wieder – wie früher – nur gelegentlich eingestreut und nicht mehr das Zentrum der Webanwendung.

Natürlich gibt es auch bei Blazor noch einige Hürden zu überwinden, wie immer bei den ersten Versionen einer neuen Technik.

Ich plane, in Zukunft weitere Versionen dieses Buchs zu veröffentlichen mit weiteren Aspekten zum Thema Blazor und auch zu den Basistechniken aus ASP.NET Core und .NET. Auch die Weiterentwicklungen von Blazor werde ich mit diesem Buch dokumentieren.

Das Buch setzt aber voraus, dass Sie **.NET, C# und Visual Studio schon kennen**. Ich biete dazu aber auch weitere Bücher an, siehe Kapitel "Notwendige Vorkenntnisse".

Dieses Fachbuch wird vertrieben auf folgenden Wegen (Ich nenne neben dem Verkaufspreis auch, wie viel – bzw. wenig – ich als Autor von den jeweiligen Händlern erhalte. Der Rest ist Gewinn der Händler):

- Gedruckt bei Amazon.de für 49,99 Euro (der Autor erhält 22,05 Euro):
www.amazon.de/exec/obidos/ASIN/3934279368/itvisions-21
- Kindle-E-Book bei Amazon.de für 44,99 Euro (der Autor erhält 14,99 Euro):
www.amazon.de/exec/obidos/ASIN/B08J5L3WQ7/itvisions-21
- PDF inkl. Updates bei Leanpub.com ab 44,99 Dollar (der Autor erhält ca. 28,75 Euro):
www.leanpub.com/Blazor60

Tipp: Sie können jederzeit Aktualisierungen des PDF-Buchs (gleiche Hauptversion) kostenfrei bei Leanpub.com beziehen. Amazon erlaubt dies leider nicht! ☹ Ich biete daher Käufer bei Amazon die PDF-Version zum Sonderpreis von 19,99 Dollar an:

www.leanpub.com/Blazor60/c/StrangeNewWorlds

Ich habe mich für den Vertriebsweg des gedruckten Buchs über Amazon entschieden, weil ich dort ständig Updates zu dem Buch einreichen kann. Per Print-on-Demand erhalten Leser dann immer das topaktuelle Buch. Oft liefert Amazon dennoch am Tag nach der Bestellung das Buch schon aus. Der Vertrieb dieses Buch über klassische IT-Verlage, die leider heutzutage immer noch größere Auflagen vorproduzieren, ist für ein sehr agiles Softwareprodukt wie Blazor keine Alternative mehr.

Da diese Preise in Anbetracht der vielen Stunden Arbeit an diesem Werk leider nicht nennenswert dazu beitragen können, den Lebensunterhalt meiner Familie zu bestreiten, ist dieses Projekt ein Hobby. Dementsprechend kann ich nicht garantieren, wann es Updates zu diesem Buch geben wird. Ich werde dann an diesem Buch arbeiten, wenn ich neben meinem Beruf als Softwarearchitekt, Berater und Dozent und meinen sportlichen Betätigungen noch etwas Zeit für das Fachbuchautorenhobby übrig habe.

Falls mir in diesem Buch oder den zugehörigen Downloads menschliche Fehler passiert sind, möchte ich mich dafür schon jetzt in aller Form bei Ihnen entschuldigen. Bitte geben Sie mir einen freundlichen, genau beschriebenen Hinweis auf meine Fehler. Ich freue mich immer über konstruktives Feedback und Verbesserungsvorschläge. Bitte verwenden Sie dazu das Kontaktformular: www.dotnet-doktor.de/Leserfeedback

Tipp: Ich belohne Sie mit E-Books für gemeldete Fehler, siehe Kapitel "Über dieses Buch/ Ihre Belohnung, wenn Sie helfen, dieses Buch zu verbessern".

Ich helfe Ihnen gerne, Ihren eigenen Programmcode zu schreiben, aber ich hoffe, Sie verstehen, dass ich dies nicht ehrenamtlich tun kann. Wenn Sie **technische Hilfe** zu Blazor und .NET oder anderen Themen rund um die Entwicklung und den Betrieb von Anwendungen (Desktop, Web und Mobile) sowie Server und Cloud benötigen, stehe ich Ihnen im Rahmen meiner beruflichen Tätigkeit für die Firmen www.IT-Visions.de (Beratung, Schulung, Support) und MAXIMAGO GmbH (Softwareentwicklung, siehe www.MAXIMAGO.de) gerne zur Verfügung. Bitte wenden Sie sich für ein Angebot an das jeweilige Kundenteam. Bitte kontaktieren Sie die Firmen aber nicht für Feedback und Verbesserungsvorschläge zu diesem Buch, da dieses Buch reine Privatsache ist.

Die Beispiele zu diesem Buch können Sie herunterladen auf der von mir ehrenamtlich betriebenen **Leser-Website** unter: www.dotnet-doktor.de/Leser

Dort müssen Sie sich zunächst registrieren. Bei der Registrierung wird ein Lösungswort abgefragt. Bitte geben Sie dort **StrangeNewWorlds** ein (siehe auch Kapitel "Programmcodebeispiele zum Download").

Herzliche Grüße aus Essen, dem Herzen der Metropole Ruhrgebiet

Holger Schwichtenberg

4 Über den Autor

- Studienabschluss Diplom-Wirtschaftsinformatik an der Universität Essen
- Promotion an der Universität Essen im Fachgebiet komponentenbasierter Softwareentwicklung
- Seit 1996 selbstständig als unabhängiger Berater, Dozent, Softwarearchitekt und Fachjournalist
- Fachlicher Leiter des Expertenteams bei www.IT-Visions.de in Essen
- Chief Technology Expert (CTE) der Softwareentwicklung bei der MAXIMAGO GmbH in Dortmund (www.MAXIMAGO.de)
- Über 80 Fachbücher beim Carl Hanser Verlag, bei O'Reilly, Microsoft Press, APress und Addison-Wesley sowie mehr als 1300 Beiträge in Fachzeitschriften
- Gutachter in den Wettbewerbsverfahren der EU gegen Microsoft (2006-2009)
- Ständiger Mitarbeiter der Zeitschriften iX (seit 1999), dotnetpro (seit 2000) und Windows Developer (seit 2010) sowie beim Online-Portal heise.de (seit 2008)
- Regelmäßiger Sprecher auf nationalen und internationalen Fachkonferenzen (z.B. Microsoft TechEd, Microsoft Summit, Microsoft IT Forum, BASTA, BASTA-on-Tour, enterJS, .NET Architecture Camp, Advanced Developers Conference, Developer Week, OOP, DOTNET Cologne, MD DevDays, Community in Motion, DOTNET-Konferenz, VS One, NRW.Conf, Net.Object Days, Windows Forum, Container Conf)
- Zertifikate und Auszeichnungen von Microsoft:
 - Microsoft Most Valuable Professional (MVP), kontinuierlich ausgezeichnet seit 2004
 - Microsoft Certified Solution Developer (MCSD)
- Thematische Schwerpunkte:
 - Softwarearchitektur, mehrschichtige Softwareentwicklung, Softwarekomponenten, SOA
 - Visual Studio, Continous Integration (CI) und Continous Delivery (CD) mit Azure DevOps
 - Microsoft .NET (.NET Framework, .NET Core), C#, Visual Basic
 - .NET-Architektur/Auswahl von .NET-Techniken
 - Einführung von .NET und Visual Studio/Migration auf .NET
 - Webanwendungsentwicklung und Cross-Plattform-Anwendungen mit HTML, ASP.NET (Core), JavaScript/TypeScript und Webframeworks wie Angular und Blazor
 - Enterprise .NET, verteilte Systeme/Webservices mit .NET, insbesondere Windows Communication Foundation, WebAPI und gRPC
 - Relationale Datenbanken, XML, Datenzugriffsstrategien
 - Objekt-Relationales Mapping (ORM), insbesondere ADO.NET Entity Framework und Entity Framework Core
 - PowerShell
 - Architektur- und Code-Reviews
 - Performance-Analysen und -Optimierung
 - Entwicklungsrichtlinien
- Ehrenamtliche Community-Tätigkeiten:
 - Vortragender für die International .NET Association (INETA)
 - Betrieb diverser Community-Sites: www.dotnet-lexikon.de, www.dotnetframework.de, www.windows-scripting.de, www.aspnetdev.de u. a.
- Firmenwebsites: www.IT-Visions.de und www.MAXIMAGO.de
- Weblog: www.dotnet-doktor.de



www.IT-Visions.de
Dr. Holger Schwichtenberg



M A X I M A G O

- | |
|--|
| <ul style="list-style-type: none"> ▪ Kontakt für Anfragen zu Schulung und Beratung: kundenteam@IT-Visions.de Telefon 0201 / 64 95 90 - 50 ▪ Kontakt für Anfragen für Softwareentwicklungsprojekte: hsc@MAXIMAGO.de Telefon 0231 / 58 69 67 - 12 ▪ Kontakt für Feedback zu diesem Buch: www.dotnet-doktor.de/Leserfeedback |
|--|

5 Über dieses Buch

5.1 Versionsgeschichte dieses Buchs

Die Versionsgeschichte dieses Buch finden Sie in einem eigenen Kapitel am Ende des Buchs.

Hinweis: Die Versionsgeschichte ist eine wichtige Referenz für die Leser, die sich aktuelle Versionen des Buchs beschaffen (z.B. über Leanpub.com) und wissen wollen, was sich geändert hat. Wenn Sie das Buch erstmalig lesen, müssen Sie die Versionsgeschichte nicht lesen.

5.2 Bezugsquelle für Aktualisierungen

Leanpub-Kunden können jederzeit Aktualisierungen des PDF-Buchs (gleiche Hauptversion!) kostenfrei bei Leanpub.com beziehen.

Käufer der Kindle- oder Druck-Version können die aktuelle PDF-Version zum Preis von 19,99 Dollar (zzgl. 7% Mehrwertsteuer) unter folgender Webadresse beziehen:

<https://leanpub.com/ASPNETBlazor/c/StrangeNewWorlds>

Hinweise: Leider erlauben Amazon u.a. Buchhändler aufgrund der Buchpreisbindungsgesetze in Deutschland den Autoren grundsätzlich nicht, dass Leser eine Aktualisierung im Kindle-Format oder in gedruckter Form vergünstigt erhalten.

Bitte beachten Sie auch, dass die ISBN-Regularien erfordern, dass bei einer Titeländerung bei neuer Produktversion eine neue ISBN vergeben und damit auch ein neues Buchprojekt bei Amazon und Leanpub erstellt werden muss.

5.3 Geplante Kapitel

Die Reihenfolge der für folgende Buchversionen geplanten Kapitel ist hier zunächst alphabetisch angeordnet und entspricht nicht der Reihenfolge, in der die Kapitel erscheinen werden.

- AdditionalAttributes und @attributes für Komponenten
- Autorisierung mit Rollen
- Anpassbarer Ladevorgang bei Blazor WebAssembly (ab Blazor 6.0)
- Blazor WebForms Components (A collection of Blazor components that emulate the web forms components of the same name) →
<https://github.com/FritzAndFriends/BlazorWebFormsComponents>
- Custom Event Arguments (ab Blazor 6.0)
- Cross-Platform-HTML-Apps mit Blazor Electron
- DotNet.createJSObjectReference()
- Eigene Basisklassen für Razor Components
- Fehlerbehandlung / blazor-error-ui → **Gehört zum Standardlieferumgang der Projektvorlagen und wird bereits verwendet in MiracleList.**
- Fluent Validator (<https://github.com/Blazored/FluentValidation>)
- Host Environments in Blazor Server verwenden
- IComponentActivator zur Kontrolle der Komponenteninstanzierung (ab Blazor 5.0)
- Installation von Blazor-Anwendungen auf Linux-Servern

- Integration von Blazor in JavaScript-Anwendungen (ab Blazor 6.0)
- IL-Linker-Konfigurationsdatei (LinkerConfig.xml)
- JavaScript-Interop ohne Marshalling mit IJSInProcessRuntime und IJSUnmarshalledRuntime (in Blazor WebAssembly)
- Razen Blazor Components (kostenfrei) → <https://razor.radzen.com>
- Radzen-RAD-Werkzeuge (kommerziell) → <https://www.radzen.com>
- Blazorise Components (kostenfrei) → <https://efroliddemo.blazorise.com>
- Synchrone JavaScript-Interop mit IJSInProcessRuntime (in Blazor WebAssembly ab Blazor 5.0)
- LESS und SCSS mit Blazor
- Leistungsoptimierung (@key, Inlining, ShouldRender u.a.)
- Token-basierte Authentifizierung für Blazor WebAssembly mit Microsoft Authentication Library (MSAL) in den NuGet-Paketen Microsoft.Authentication.WebAssembly.Msal und Microsoft.AspNetCore.Authentication.AzureAD.UI gegen Azure Active Directory (AAD) (seit 3.2 Preview 2)
- preserve-component-state (ab Blazor 6.0)
- PreseveWhiteSpace
- Relinkung (ab Blazor 6.0)
- Routing-Anpassungen (<https://chrissainty.com/building-a-custom-router-for-blazor/amp/>)
- Testen von Blazor-Anwendungen (Microsoft.AspNetCore.Components.Testing und BUnit)
→ Beispiele zu BUnit finden Sie bereits im **MiracleList-Beispiel**, das Sie herunterladen können!
- TypeScript in Blazor Apps
- Validierung: Anpassung der CSS-Klassen (ab Blazor 5.0)
- Visual Studio Code als alternatives Werkzeug zu Visual Studio

Hinweis: Mit dieser Liste kommender Themen möchte ich gleichzeitig klarstellen, welche Themen Sie derzeit nicht im Buch finden, damit Sie als Leser keine falschen Erwartungen haben.

5.4 Programmiersprache in diesem Buch

Als Programmiersprache kommt in diesem Buch C# zum Einsatz, weil C# die einzige offiziell von Microsoft unterstützte Programmiersprache für Blazor ist.

Als zweite Programmiersprache für Blazor gibt es nur F# über eine Erweiterung [<https://github.com/fsboler/Bolero>], die außerhalb von Microsoft entwickelt wird und daher kein offizielles Produkt von Microsoft ist.

5.5 Notwendige Vorkenntnisse

Webprogrammierung ist ein großes, komplexes Gebiet. In diesem Buch konzentriere ich mich auf ASP.NET Blazor im engeren Sinne.

Als Vorkenntnisse sollten Sie zumindest ein gutes Grundlagenwissen in folgenden Gebieten mitbringen:

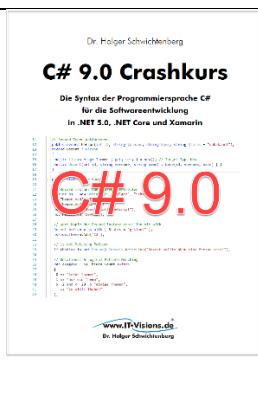
- HTTP
- HTML
- CSS
- JavaScript
- .NET Core
- ASP.NET Core
- C#
- Entity Framework Core
- Visual Studio

Ich habe drei Bücher geschrieben, durch die Sie dieses Grundlagenwissen erlangen können.

Für die ersten fünf Punkte ist dies:

	<p>Moderne Webanwendungen für .NET-Entwickler: Server-Anwendungen, Web APIs, SPAs & HTML-Cross-Platform-Anwendungen mit ASP.NET, ASP.NET Core, JavaScript/TypeScript und Angular</p> <p>ISBN 3960090153</p> <p>https://www.amazon.de/exec/obidos/ASIN/3960090153/itvisions21</p> <ul style="list-style-type: none"> ▪ Gedruckt: 49,90 Euro ▪ Kindle: 39,99 Euro
--	--

Wenn Sie C# lernen wollen, möchte ich Ihnen mein Buch "C# 9.0 Crashkurs" empfehlen:

	<p>C# 9.0 Crashkurs</p> <ul style="list-style-type: none"> ▪ Gedruckt (Print-on-Demand) bei Amazon.de für 24,99 Euro (der Autor erhält 10,52 Euro): www.amazon.de/exec/obidos/ASIN/3934279406/itvisions-21 ▪ Kindle-E-Book bei Amazon.de für 9,99 Euro (der Autor erhält 6,29 Euro): www.amazon.de/exec/obidos/ASIN/B08KPLNBZS/itvisions-21 ▪ PDF-E-Book bei Leanpub.com ab 11,99 Dollar (der Autor erhält ca. 14,00 Euro): www.leanpub.com/CSharp9
---	---

Um Entity Framework Core zu lernen, möchte ich Ihnen mein Buch "Moderne Datenzugriffslösungen mit Entity Framework Core 5.0" empfehlen:

	<h3>Moderne Datenzugriffslösungen mit Entity Framework Core 5.0</h3> <ul style="list-style-type: none"> ▪ Gedruckt bei Amazon.de für 69,99 Euro (der Autor erhält 29,90 Euro): www.amazon.de/exec/obidos/ASIN/3934279252/itvisions-21 ▪ Kindle-E-Book bei Amazon.de für 59,99 Euro (der Autor erhält 18,52 Euro): www.amazon.de/exec/obidos/ASIN/B087QSSKW1/itvisions-21 ▪ PDF-E-Book bei Leanpub.com ab 44,99 Dollar (der Autor erhält 35,99 Dollar): www.leanpub.com/EntityFrameworkCore5
---	--

5.6 Hinweis zur Erwähnung der Blazor-Varianten in diesem Buch

Vor .NET 6 gab es nur zwei Blazor-Arten: Blazor Server und Blazor WebAssembly. Mit .NET 6 gibt es eine dritte Blazor-Art: Blazor Desktop, die man auch noch mal differenzieren könnte in hybride Apps auf Desktop- und Mobilbetriebssystemen.

Es wird im Buch vorerst noch einige Stellen geben, wo ich nur von zwei Blazor-Arten (Blazor Server und Blazor WebAssembly) spreche. Es ist mir als Autor aus zeitlichen Gründen vorerst nicht möglich, das ganze Buch nach diesen Stellen zu durchkämen. Was per Volltextsuche auffindbar war, habe ich erledigt.

Geben Sie mir gerne einen freundlichen Hinweis auf Stellen, wo die Erwähnung von Blazor Desktop fehlt, siehe nächstes Unterkapitel

5.7 Ihre Belohnung, wenn Sie helfen, dieses Buch zu verbessern!

Wenn Sie Fehler in diesem Buch finden, bin ich Ihnen nicht nur wirklich sehr dankbar, sondern Sie bekommen auch eine Belohnung in Form von aktualisierten oder weiteren E-Books.

Fehlerart	E-Book-Guthaben
Inhaltlicher Fehler	Pro Fehler 20 Euro
Sprachlicher Fehler	Pro Fehler 4 Euro

Ein Beispiel: Wenn Sie einen inhaltlichen Fehler und fünf Rechtschreibfehler in diesem Buch finden, dann haben Sie bei mir 40 Euro gut. Dafür können Sie dann eins meiner selbstverlegten Bücher als E-Book bekommen.

Die selbstverlegten Bücher finden Sie unter www.IT-Visions.de/Verlag

Melden Sie die Fehler unter www.dotnet-doktor.de/Leserfeedback

Schreiben Sie dabei, welches E-Book Sie wünschen. Das Buch schicke ich Ihnen dann per E-Mail zu.

Tipp: Auch Fehler auf meiner persönlichen Website www.dotnet-doktor.de und der Firmenwebsite www.IT-Visions.de zählen mit!

Ich freue mich auf Ihre Fehlermeldung!

Holger Schwichtenberg

P.S. Falls Sie Ihre Fehlermeldung sich auf eine Ausgabe des Buchs bezieht, die älter als ein Jahr ist und der Fehler in der aktuellsten Ausgabe schon behoben ist, dann zählt das leider nicht.

6 Programmcodebeispiel zum Download

Die Beispiele zu diesem Buch können Sie als Visual Studio-Projekte herunterladen.

Hinweis: Bitte beachten Sie, dass nicht jede einzelne Zeile Programmcode, die Sie in diesem Buch finden, in den herunterladbaren Projekten enthalten sein kann. Die Projekte bilden funktionierende Lösungen. In diesem Buch werden auch alternative Lösungen für Einzelfälle diskutiert, die nicht unbedingt zu einer Gesamtlösung passen.

6.1 Webadresse für Downloads

Den Download der Beispiele zu diesem Buch finden Sie auf der Leser-Website unter der Webadresse:

www.dotnet-doktor.de/Leser

Dort müssen Sie sich bitte einmalig registrieren. Bei der Registrierung wird ein **Losungswort** abgefragt, das Sie als Käufer dieses Buchs ausweist. Bitte geben Sie dort **StrangeNewWorlds** ein.

Durch die Registrierung erhalten Sie dann ein persönliches **Kennwort** per E-Mail zugesendet, das Sie danach immer wieder für die Anmeldung zum Leserportal nutzen können.

Bei Problemen mit der Registrierung nutzen Sie bitte das auf der o.g. Webseite verlinkte FAQ.

6.2 Übersicht über die Beispiele

Die folgende Tabelle zeigt, wo Sie in dem herunterladbaren ZIP-Paket die Beispiele der einzelnen Kapitel des Fachbuchs finden.

 Blazor50_Neuheiten
 BlazorMicroApps
 BlazorWebservices
 MiracleList_Bazor
 MiracleListClientSeleniumTests
 WebformToBlazor
 Readme!!! Copyright Haftungsausschluss Support.pdf
 readme.txt

Abbildung: Inhalt des Download-Pakets zu diesem Buch

Es war eine bewusste Entscheidung, in diesem Buch nicht für jedes einzelne Beispiel ein eigenes Blazor-Projekt zu schaffen, sondern die meisten Beispiele in zwei Blazor-Projekte (eins für Blazor WebAssembly und eines für Blazor Server) zu integrieren. Dies hat folgende Vorteile:

- Auf diese Weise sind komplexere, praxisnahe Beispiele möglich.
- Die Aktualisierung der Beispiele auf neue NuGet-Paket-Versionen und API-Änderungen geht schneller (dies ist wichtig in der heutigen Zeit, in der es ständig neue Versionen, auch mit Breaking Changes, gibt).

Ordner	Programmcode aus Kapitel(n)	Hinweise
/src/MiracleList_SSB	Alle	Blazor Server-Beispiele einschließlich der Blazor Server-Implementierung des Fallbeispiels "MiracleList"

/src/MiracleList_CSBlazor	Alle	Blazor WebAssembly-Beispiele einschließlich der Blazor WebAssembly-Implementierung des Fallbeispiels "MiracleList"
/src/MiracleList_BD	Alle	Blazor WPF/Desktop-Beispiele einschließlich der Blazor WebAssembly-Implementierung des Fallbeispiels "MiracleList"
/src/MiracleListAPI_Proxy	Fallbeispiel	Generierte Proxy-Klasse für MiracleList-Backend-WebAPI
/src/ITVBlazorRCL	JavaScript-Interoperabilität Razor Class Library Fallbeispiel	Razor Class Library mit Code für JavaScript-Interoperabilität; wird im Fallbeispiel von Blazor Server, Blazor WebAssembly und Blazor Desktop verwendet. Realisiert die vielfach in diesem Buch verwendete Hilfsklasse BlazorUtil.cs mit zugehöriger JavaScript-Datei BlazorUtil.js
/src/MLBlazorRCL	Alle	Diese Razor Class Library stellt gemeinsam genutzte Inhalte (Klassen, Razor Components sowie statische Webelemente wie Grafiken und eine CSS-Datei) für das MiracleList-Fallbeispiel und ergänzende Beispiele aus diesem Buch bereit. Dieses Projekt wird im Fallbeispiel von Blazor Server, Blazor WebAssembly und Blazor Desktop gemeinsam verwendet.
/src/SamplesRCL	Alle	Ergänzende Beispiele aus dem Buch, die nicht in das MiracleList-Szenario fallen. Dieses Projekt wird im Fallbeispiel von Blazor Server, Blazor WebAssembly und Blazor Desktop gemeinsam verwendet. Diese Beispiele sind im Menü "Blazor-Beispiele außerhalb der MiracleList" aufrufbar.
/src/MiracleList_Backend	Fallbeispiel	Backend (ASP.NET Core WebAPI und ASP.NET Razor Pages)
/Blazor60_Neuheiten	Alle	Für neue Funktionen in Blazor 6.0 gibt es derzeit ein eigenes Beispielprojekt NET6BW.csproj mit neuen Funktionen in Blazor 6.0 (z.B. DynamicComponent und ErrorBoundaries).
/BlazorWebservices	Zugriff auf Webservices/Web APIs	Die Projektmappe "BlazorWebservices" umfasst die Implementierung von WebAPIs und Google RPC-Diensten und den Aufruf aus Blazor-WebAssembly.

		Eigenständiges Beispiel, das die Datenbank der fiktiven Fluggesellschaft World Wide Wings (www.world-wide-wings.de) verwendet.
/WebformsToBlazor	Umstieg von Webforms auf Blazor	Eigenständiges Beispiel für den Umstieg von ASP.NET Webforms zu ASP.NET Core Blazor, das die Datenbank der fiktiven Fluggesellschaft World Wide Wings (www.world-wide-wings.de) verwendet.
/BlazorMicroApps	Integration von Blazor-Apps in eine Webforms-Anwendung	Eigenständiges Beispiel, das keine Datenbank verwendet, sondern auf den Standardprojektvorlagen mit Daten im RAM basiert.

6.3 Eingesetztes CSS-Framework

Alle Beispiele verwenden die CSS-Klassen des CSS-Frameworks Bootstrap [<https://getbootstrap.com>] zur Formatierung. Dies ist natürlich nur eine von vielen Gestaltungsoptionen in Blazor. Sie können auch andere Design Systeme wie Tailwind, Bulma oder Microsoft Fluent UI einsetzen.

Beispiele für CSS-basierte Design-Systeme:

- Twitter Bootstrap [<https://getbootstrap.com>]
- Google Material Design (Angular Material <https://material.angular.io/>, Vuetify <https://vuetifyjs.com/en/>)
- Microsoft Fluent Design System / Fluent UI [<https://developer.microsoft.com/de-de/fluentui#/>]
- Tailwind CSS [<https://tailwindcss.com/>]
- Ant Design [<https://ant.design/>]
- Bulma [<https://bulma.io/>]
- eFrolicss [<https://efrolic.github.io/css/>]
- ORGENIC UI [<https://github.com/MAXIMAGO/organic-ui>]

6.4 Das Fallbeispiel "MiracleList"

Dieses Kapitel führt in das zentrale Blazor-Fallbeispiel "MiracleList" ein, das an vielen Stellen in diesem Buch verwendet wird.

6.4.1 Szenario

Das Fallbeispiel ist sehr praxisnah, weil das Anwendungsszenario ein Nachbau einer existierenden, sehr erfolgreichen Anwendung ist.

Im Jahr 2015 zahlte Microsoft für die Übernahme des Berliner App-Herstellers Wunderlist mehr als 100 Millionen US-Dollar [www.heise.de/newsticker/meldung/Microsoft-uebernimmt-Berliner-Startup-6Wunderkinder-2678017.html].

"MiracleList" ist eine Nachprogrammierung dieser Aufgabenverwaltung als Webanwendung und Cross-Plattform-Anwendung. Es gibt vier Implementierungen: eine für Blazor Server, eine für Blazor WebAssembly, eine für Blazor Desktop und eine mit Angular und TypeScript. In diesem Buch werden nur die ersten drei Varianten besprochen.



Abbildung: Das MiracleList-Logo

Hinweis: Microsoft hat mittlerweile Wunderlist neu implementiert als "Microsoft To-Do". Wunderlist wird am 6. Mai 2020 eingestellt. Das Fallbeispiel MiracleList existiert dessen ungeachtet weiterhin.

6.4.2 Technische Basis für MiracleList

Das MiracleList-Backend ist mit ASP.NET Core realisiert.

Die ursprüngliche Version des MiracleList-Frontends ist mit TypeScript und Angular realisiert. Die drei in diesem Buch beschriebenen Frontend-Versionen basieren auf Blazor Server und Blazor WebAssembly sowie Blazor Desktop.

6.4.3 Webadressen

Die fertige Webanwendung läuft öffentlich im Internet in der Microsoft-Cloud "Azure":

- Webanwendung "MiracleList"-Frontend mit Blazor Server und C#:
<https://miracelist-bs.azurewebsites.net>
- Webanwendung "MiracleList"-Frontend mit Blazor WebAssembly und C#:
<https://miracelist-bw.azurewebsites.net>
- Frontend mit Angular und TypeScript (Webanwendung und Cross-Platform-Desktop-Clients für Windows, Linux und macOS):
<https://miracelist.azurewebsites.net>
- Backend mit ASP.NET Core für Blazor WebAssembly- und Angular-Clients:
<https://miracelistbackend.azurewebsites.net>

6.4.4 Projektmappenaufbau

Die MiracleList-Projektmappe (MiracleList.sln) besteht aus zahlreichen Projekten, die logisch in fünf Ordnern sortiert sind:

- **Backend:** Alle Projekte für das MiracleList-Backend
- **Blazor-Frontends:** Die Blazor WebAssembly- und Blazor Server-Implementierung des Frontends mit Hilfsbibliotheken
- **Solution Items:** Hilfsskript für DevOps-Prozesse
- **Tests:** Unit Tests und Integrationstests
- **Tools:** Hilfsprojekte für das Anlegen der Datenbank im DevOps-Prozess und ein Diagramm der Geschäftsobjektklassen

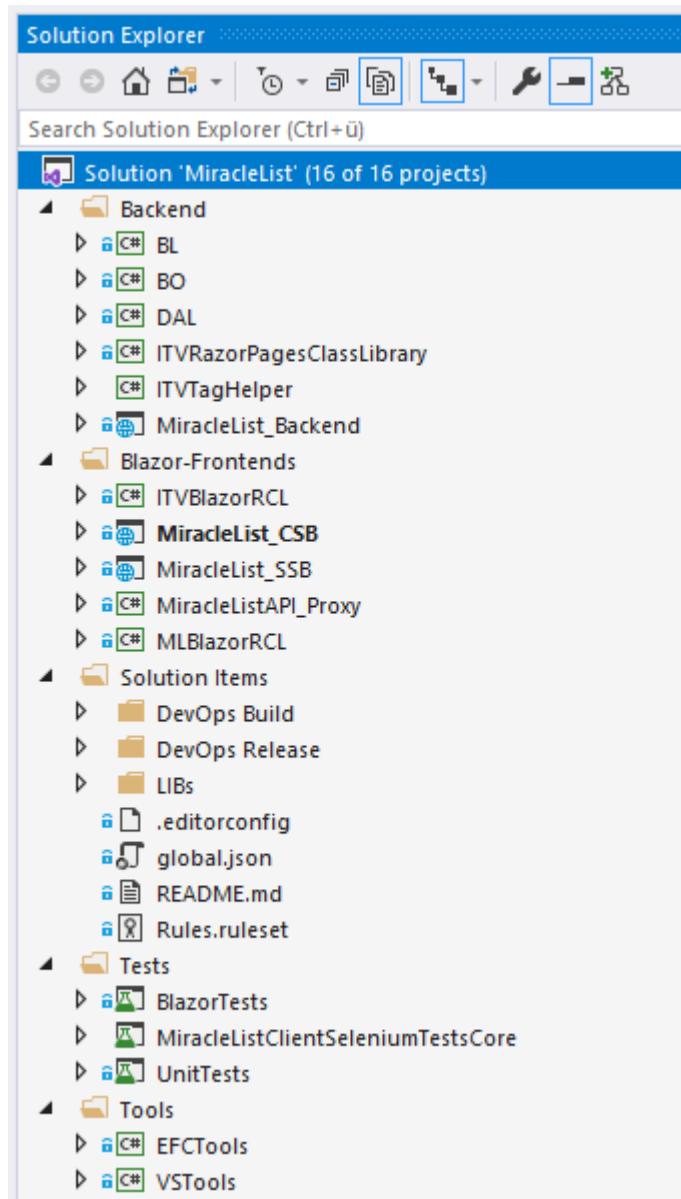


Abbildung: Aufbau der Projektmappe

6.4.5 MiracleList-Bildschirmmasken

Die Anmeldeseite fragt E-Mail-Adresse und Kennwort. Damit es leicht ist, MiracleList als Demonstrations-Anwendung zu nutzen, ist eine explizite Benutzerregistrierung ausdrücklich nicht vorgesehen. Wenn es für diese E-Mail-Adresse noch kein Benutzerkonto gibt, wird automatisch ein neues Benutzerkonto angelegt mit einigen Beispielaufgaben.

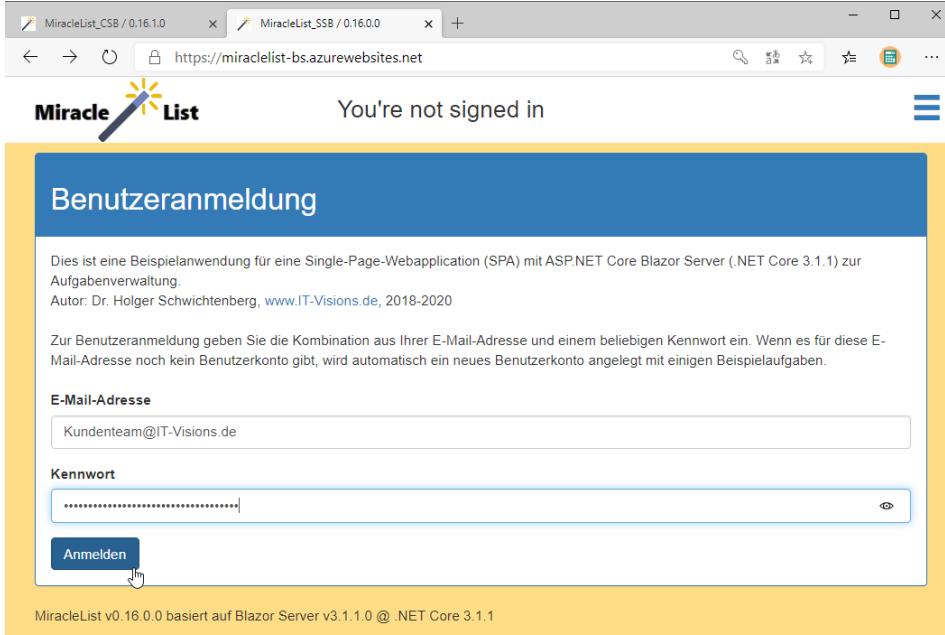


Abbildung: Anmeldeseite bei der Blazor Server-Implementierung des MiracleList-Frontends

Bei der Blazor WebAssembly-Implementierung wird zusätzlich eine Auswahl für den Backend-Server gegeben. Wenn die Blazor WebAssembly-Anwendung in Visual Studio gestartet wird, wird als Backend zusätzlich "localhost" zur Auswahl gestellt, um sich mit einer lokalen Version des Backends, statt dem in Azure gehosteten Backend zu verbinden. Unter der Backend-Server sieht man eine Zeile mit Zustands-Abzeichen ("Badges") für die Server. Ein grünes Abzeichen bedeutet, dass der Server verfügbar ist. Grau bedeutet, der Zustand wird gerade ermittelt. Bei einem roten Abzeichen ist der Server nicht verfügbar.

Dies ist eine Beispielanwendung für eine Single-Page-Webapplication (SPA) mit ASP.NET Core Blazor Webassembly (Mono 6.11.0 (explicit/c32414966c7)) zur Aufgabenverwaltung.
Autor: Dr. Holger Schwichtenberg, www.IT-Visions.de, 2018-2020

Zur Benutzeranmeldung geben Sie die Kombination aus Ihrer E-Mail-Adresse und einem beliebigen Kennwort ein. Wenn es für diese E-Mail-Adresse noch kein Benutzerkonto gibt, wird automatisch ein neues Benutzerkonto angelegt mit einigen Beispieldaten.

E-Mail-Adresse
Kundenteam@IT-Visions.de

Kennwort
.....

Backend-Server
https://miraclelistbackend.azurewebsites.net/

Status: <https://miraclelistbackend-staging8f46.azurewebsites.net> <https://miraclelistbackend.azurewebsites.net>

Anmelden

MiracleList v0.16.1.0 basiert auf Blazor WebAssembly v3.1.0.0 @ Mono 6.11.0 (explicit/c32414966c7)

Abbildung: Anmeldeseite bei der Blazor WebAssembly-Implementierung des MiracleList-Frontends

Hinweis: Wenn Sie localhost als Backend-Server nutzen wollen, müssen Sie auf Ihrem Rechner das Backend (Projekt MiracleList_Backend.csproj) zuvor gestartet und auch die Datenbank angelegt haben (siehe dazu Readme.md)! Wenn Sie das Backend auf einem anderen Port laufen lassen, müssen Sie in AppSettings.cs den Port ändern!

Dies ist eine Beispielanwendung für eine Single-Page-Webapplication (SPA) mit ASP.NET Core Blazor Webassembly (Mono 6.11.0 (explicit/c32414966c7)) zur Aufgabenverwaltung.
Autor: Dr. Holger Schwichtenberg, www.IT-Visions.de, 2018-2020

Zur Benutzeranmeldung geben Sie die Kombination aus Ihrer E-Mail-Adresse und einem beliebigen Kennwort ein. Wenn es für diese E-Mail-Adresse noch kein Benutzerkonto gibt, wird automatisch ein neues Benutzerkonto angelegt mit einigen Beispieldaten.

E-Mail-Adresse
test@ihredomain.de

Kennwort
.....

Backend-Server
http://localhost:8889/

Status: <https://miraclelistbackend.azurewebsites.net> <https://miraclelistbackend-staging8f46.azurewebsites.net> <http://localhost:8889>

Anmelden

MiracleList v0.16.2.0 basiert auf Blazor WebAssembly v3.1.0.0 @ Mono 6.11.0 (explicit/c32414966c7)

Abbildung: Beim Start der Blazor WebAssembly-Anwendung auf "localhost" wird auch das lokale Backend angeboten

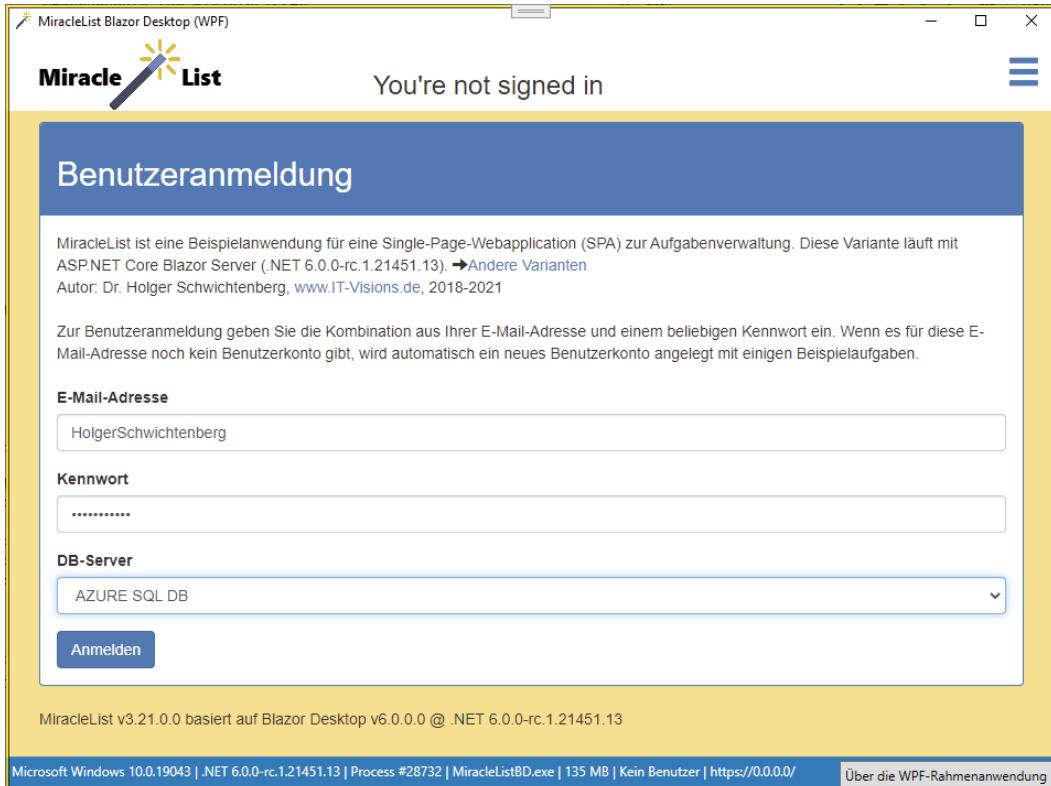


Abbildung: MiracleList in der Blazor Desktop-Variante, gehostet in einer Windows-Desktop-Anwendung mit Windows Presentation Foundation mit direktem Datenbankzugriff wie bei Blazor Server

Der angemeldete Benutzer kann eine Liste von Aufgabenkategorien erstellen und in jeder Kategorie eine Liste von Aufgaben anlegen.

Eine Aufgabe besteht aus einem Titel, einer Notiz, einem Eintragsdatum sowie einem Fälligkeitsdatum und kann als erledigt markiert werden. Über die Funktionen von Wunderlist hinaus kann in MiracleList eine Aufgabe drei (A, B oder C), statt nur zwei Wichtigkeitsgrade (Wichtig ja/nein) sowie einen Aufwand (Zahl) besitzen. Bewusst besitzt der Aufwand keine Maßeinheit; der Benutzer kann selbst entscheiden, ob er den Aufwand in Stunden, Tagen oder nur in relativen Werten, wie z.B. "1" (für niedrig) bis "10" (für hoch), vergeben will.

Wie bei Wunderlist kann eine Aufgabe Teilaufgaben besitzen, wobei eine Teilaufgabe nur einen Titel und einen Status besitzt. Einige Details aus dem Original fehlen aber in MiracleList, z.B. das Hochladen von Dateien zu Aufgaben, das Verschieben von Aufgaben zwischen Kategorien, die Suche nach Hashtags, das Duplizieren und Drucken von Listen sowie der Aufgabenaustausch zwischen Benutzern. Einige Funktionen wie anklickbare Hyperlinks in Aufgabentexten sind nicht realisiert, um einen Missbrauch der für alle Nutzer offenen Website zu vermeiden.

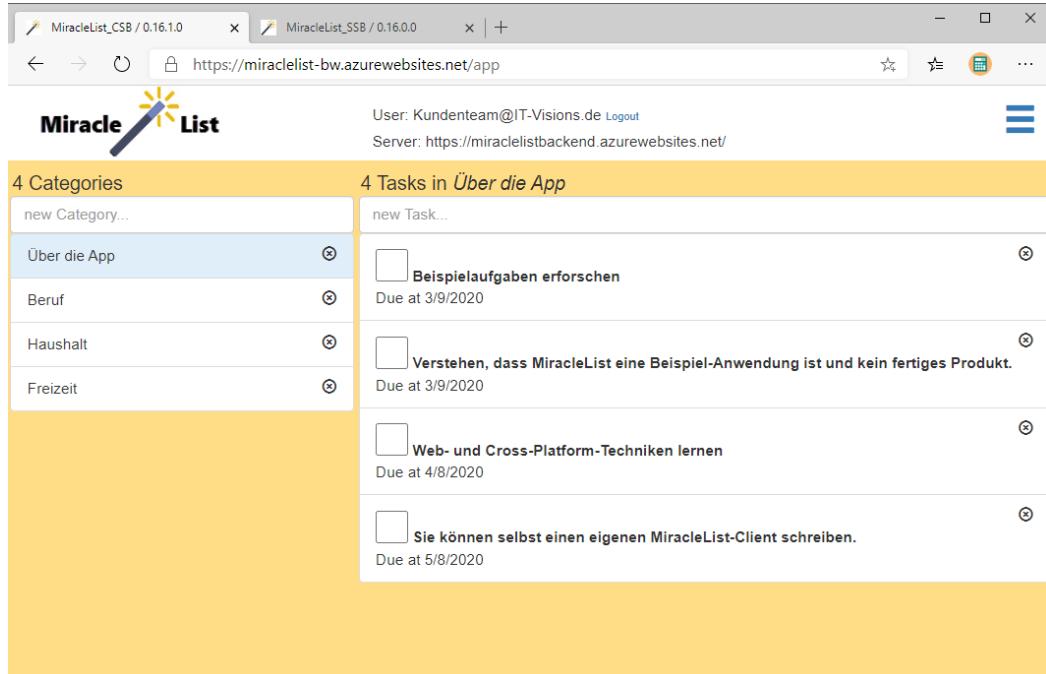


Abbildung: Hauptansicht der MiracleList-Anwendung

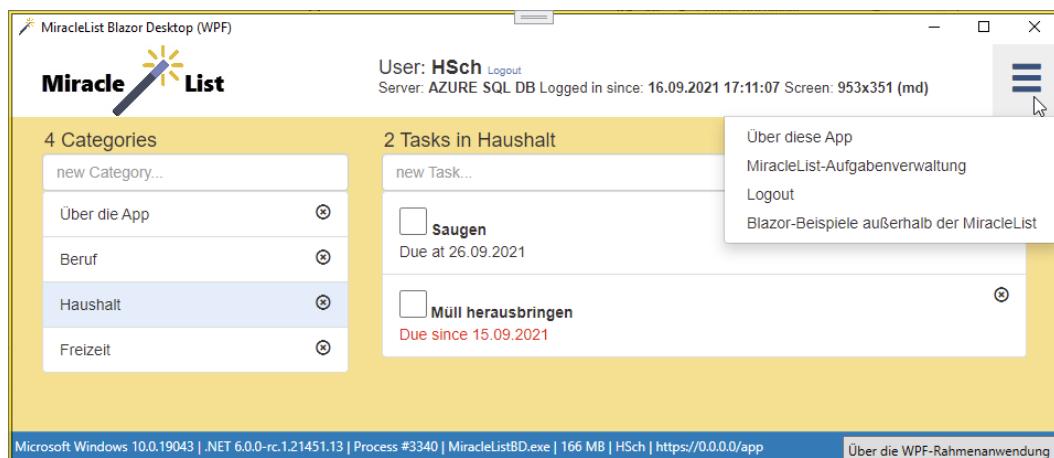


Abbildung: Hauptansicht der MiracleList-Anwendung in der Variante "Blazor Desktop"

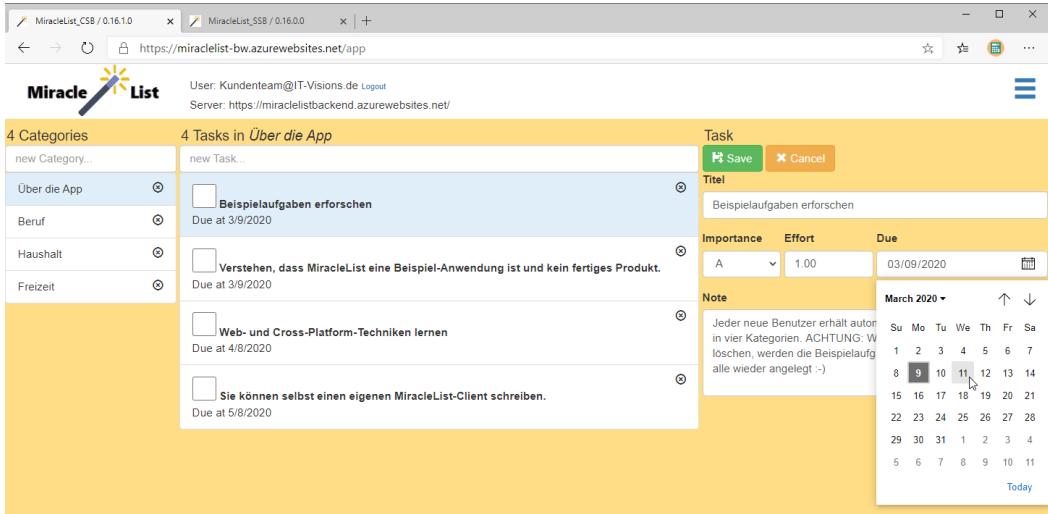


Abbildung: Bearbeitungsansicht der MiracleList-Anwendung

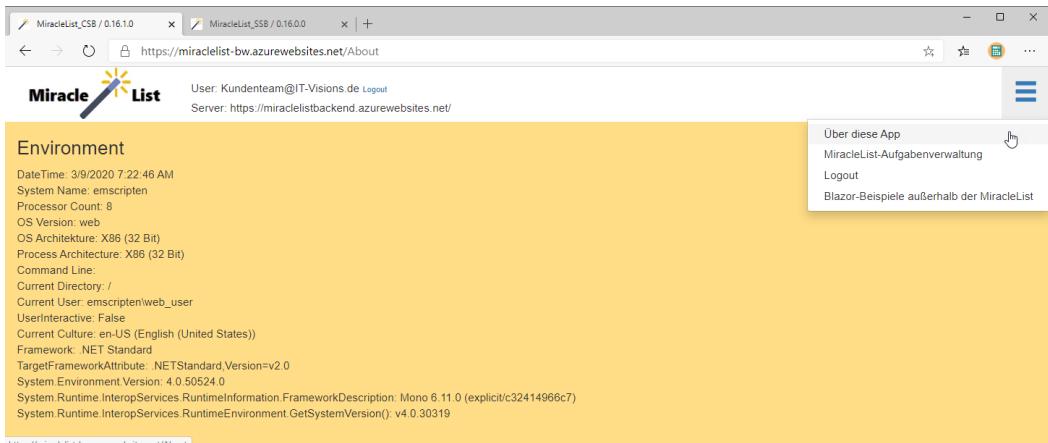


Abbildung: Technische Daten sieht man im Menü "Über diese App"

6.5 Weitere Beispiele innerhalb der MiracleList-Webanwendung

Viele Beispiele in diesem Buch stammen direkt aus dem Anwendungsszenario der Aufgabenverwaltung "MiracleList". Allerdings stammen nicht alle Beispiele dieses Buchs aus dem Szenario der Aufgabenverwaltung selbst. Einige Beispiele greifen nicht auf die Geschäftslogik und die Datenbank von MiracleList zurück. Dies hat zwei Gründe:

- Einige Beispiele passen inhaltlich nicht in das Szenario "MiracleList".
- Aus didaktischen Gründen ist es an einigen Stellen geboten, einfachere Beispiele zu machen.

Alle Beispiele, die nicht Teil der Aufgabenverwaltung sind, finden Sie dennoch in den beiden Blazor-Projekten der MiracleList. Sowohl in der Blazor WebAssembly- als auch der Blazor Server-Variante rufen Sie diese unter der relativen URL /Demos oder den Menüpunkt "Blazor-Beispiele außerhalb der MiracleList" auf.



Abbildung: Die Blazor-Beispiele außerhalb der MiracleList sind in das Menü der MiracleList integriert.

Hinweis: Den Quellcode der meisten Beispiele in diesem Buch, deren URL mit "/Demos" beginnt, finden Sie in dem Projekt /src/Samples. Nur einige wenige Beispiele, die spezifisch sind für eine bestimmte Blazor-Variante, finden Sie in den spezifischen Projekten, also /src/MiracleList_CS8/Demos und /src/MiracleList_SSB/Demos sowie /src/MiracleList_BD/Demos.

6.6 Visual Studio 2022

Für die Entwicklung von Blazor-Anwendungen sollten Sie die Microsoft-Entwicklungsumgebung Visual Studio 2022 verwenden in der jeweils aktuellen Update-Version.

Bezugsquelle: <https://visualstudio.microsoft.com/de/vs>

Es gibt Visual Studio 2022 in drei Varianten: Community, Professional und Enterprise. Für die Entwicklung von Blazor-Anwendungen ist (für Einzelpersonen und Organisation bis fünf Entwickler) die kostenfreie Community-Variante ausreichend. Einige fortgeschrittene Werkzeugfunktionen (z.B. einige Analyse- und Testfunktionen wie IntelliTrace, Live Unit Testing und Code Coverage) erhalten Sie nur mit der Enterprise-Version. Einen Vergleich der Fähigkeiten der Versionen finden Sie unter:

<https://visualstudio.microsoft.com/de/vs/compare>

Unterstützte Features	Visual Studio Community	Visual Studio Professional	Visual Studio Enterprise
	Kostenloser Download	Kaufen	Kaufen
⊕ Unterstützte Nutzungsszenarien	●●●○	●●●●	●●●●
Unterstützung für Entwicklungsplattformen ²	●●●●	●●●●	●●●●
⊕ Integrierte Entwicklungsumgebung	●●●○	●●●○	●●●●
⊕ Erweitertes Debuggen und Diagnose	●●○○	●●○○	●●●●
⊖ Testtools	●○○○	●○○○	●●●●
Live Unit Testing			●
IntelliTest			●
Microsoft Fakes (Isolation von Komponententests)			●
Code Coverage			●
Komponententests	●	●	●

Abbildung: Ausschnitt aus dem Vergleich der Visual Studio-Varianten

Die aktuellen Preise finden Sie unter <https://visualstudio.microsoft.com/de/vs/pricing>.

Tipp: Kostenfrei können Sie die jeweils aktuelle Preview-Version verwenden, die es hier gibt: <https://visualstudio.microsoft.com/de/vs/preview>. Diese ist aber jeweils in einigen Punkten noch nicht ausgereift und der Einsatz für die professionelle Softwareentwicklung daher nicht empfohlen.

Man kann eine stabile Version und eine Preview-Version parallel auf einem Windows-System installieren.

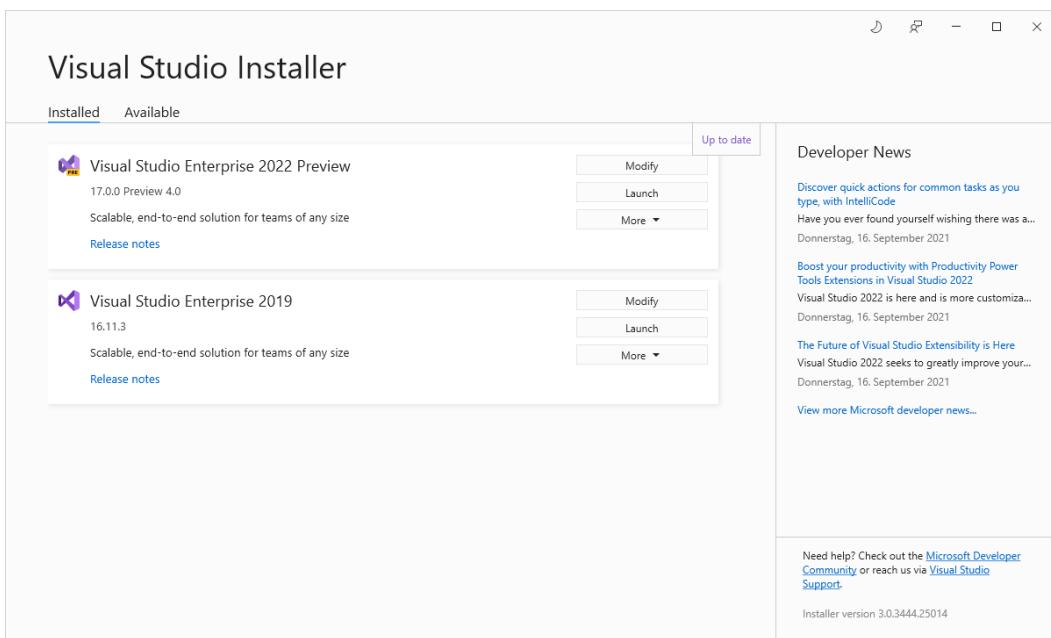


Abbildung: Parallelinstallation der stabilen Version 16.7 und der Preview 1 von 16.8

Hinweis: Visual Studio 2022 entspricht der Version 17. Seit einigen Jahren aktualisiert Microsoft die Entwicklungsumgebung im Abstand von wenigen Wochen und vergibt dafür neue Nummern an der zweiten Stelle: 17.1, 17.2, 17.3 usw.

Für die Softwareentwicklung mit Blazor müssen Sie den Workload "ASP.NET and web development" installieren.

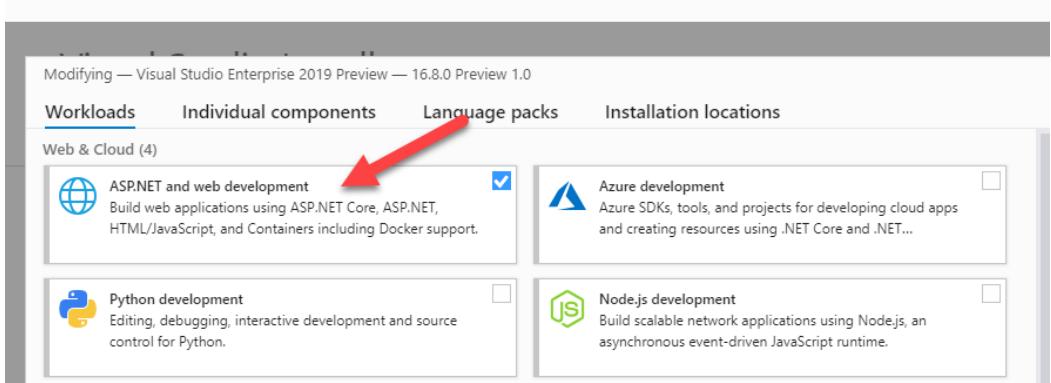


Abbildung: Notwendige Workloads für ASP.NET Core Blazor

Praxistipp: Verwenden Sie die englische Version von Visual Studio, nicht die deutsche Übersetzung. Diese hat einige Übersetzungsschwächen. Zudem ist das "googeln" nach Hilfe im Internet eingeschränkt, wenn Sie nach deutschen Begriffen suchen, weil sie die genaue englische Bezeichnung nicht kennen.

6.7 .NET SDK

Ein typisches Problem, das Sie nicht nur mit den Programmcodebeispielen in diesem Buch, sondern auch mit Ihren Projekten in der Praxis haben können, ist das Fehlen der notwendigen Installation

des .NET 6 Software Development Kits auf Ihrem Entwicklungssystem. Dann kann Visual Studio die Projekte nicht laden, siehe folgende Abbildung.

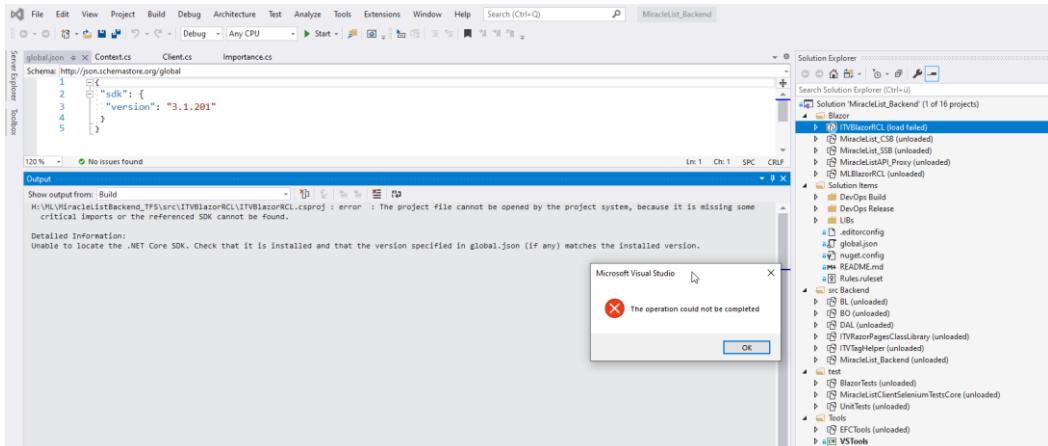


Abbildung: Die notwendige SDK-Version fehlt

Es gibt grundsätzlich zwei Verfahren, wie die .NET SDK-Version gewählt wird:

- Im Standard wird die aktuellste .NET SDK-Version verwendet (einschließlich Preview-Versionen). Dies kann zu unerwünschten Effekten beim Kompilieren führen, z.B. aufgrund von Veränderungen (Breaking Changes) in neueren SDK-Versionen.
- Wenn es eine Datei global.json im Projektunterordner gibt, wird die dort festgelegte SDK-Version für diesen Ordner und alle Unterordner verwendet.

Hinweis: Der Einsatz einer global.json-Datei ist empfohlen. Sie erhalten z.B. in Azure DevOps folgenden Hinweis: "Unless you have locked down a SDK version for your project(s), a newer SDK might be picked up which might have breaking behavior as compared to previous versions."

Wenn Visual Studio die in der global.json festgelegte .NET SDK-Version nicht finden kann, haben Sie zwei Optionen:

- Installieren Sie die in der global.json-Datei verlangte .NET SDK-Version. Sie bekommen diese kostenfrei unter der Webadresse <https://dotnet.microsoft.com/download>.
- Sie ändern die SDK-Version in der global.json auf eins der bei Ihnen installierten .NET SDKs. Welche SDKs Sie installiert haben, erfahren Sie mit dem Kommandozeilenbefehl `dotnet --list-sdks`

```

PS T:\> dotnet --version
6.0.100-rc.1.21458.32
PS T:\> dotnet --list-sdks
2.1.818 [C:\Program Files\dotnet\sdk]
3.1.413 [C:\Program Files\dotnet\sdk]
5.0.101 [C:\Program Files\dotnet\sdk]
5.0.103 [C:\Program Files\dotnet\sdk]
5.0.104 [C:\Program Files\dotnet\sdk]
5.0.203 [C:\Program Files\dotnet\sdk]
5.0.207 [C:\Program Files\dotnet\sdk]
5.0.302 [C:\Program Files\dotnet\sdk]
5.0.303 [C:\Program Files\dotnet\sdk]
5.0.401 [C:\Program Files\dotnet\sdk]
6.0.100-preview.6.21355.2 [C:\Program Files\dotnet\sdk]
6.0.100-rc.1.21458.32 [C:\Program Files\dotnet\sdk]
PS T:\>

```

Abbildung: Liste der auf einem System installierten .NET SDKs

Hinweise: Wenn die Eingabeaufforderung den Befehl "dotnet" nicht finden kann, haben Sie gar kein .NET SDK installiert. Nicht jede .NET SDK-Version ist mit jeder Version von Visual Studio kompatibel.

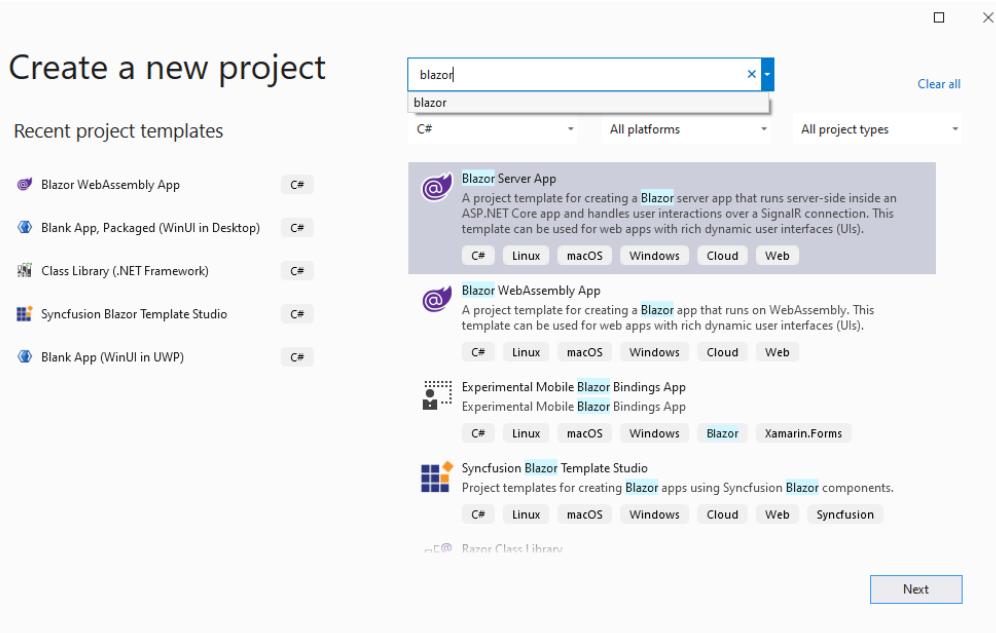
6.8 Testen Ihrer PC-Konfiguration

In diesem Kapitel wird ein kurzer Test Ihrer Systemkonfiguration durchgeführt mit dem Ziel, festzustellen, ob Ihr System korrekt funktioniert. Es wird ein Blazor WebAssembly-Projekt erstellt.

Ein Projekt startet man mit Verwendung der Projektvorlage "Blazor Server" oder "Blazor WebAssembly" im Dialog "File/New Project" in Visual Studio.

Tipp: Geben Sie im Suchdialog "Blazor" ein, siehe Abbildung.

Wählen Sie hier zum Test "Blazor WebAssembly". Nach der Auswahl des Namens (geben Sie ein "MiracleListBW" für das Projekt und "MiracleList" für die Projektmappe) und des von Ihnen wählbaren Zielpfades (wie immer sollte man einen Dateisystempfad ohne Leerzeichen wählen, das macht die Verwendung von Kommandozeilenwerkzeugen einfacher), kommt ein weiterer Dialog. Hier ist ".NET 6.0", "No Authentication", "Configure for HTTPS" und "Progressive Web App" zu wählen.

*Abbildung: Wahl der Projektvorlage*

Configure your new project

Blazor WebAssembly App C# Linux macOS Windows Cloud Web

Project name
MiracleListBW

Location
T:\VS

Solution
Create new solution

Solution name ⓘ
MiracleListBW

Place solution and project in the same directory

Back Next

Abbildung: Einstellungen für die neue Blazor WebAssembly-Anwendung

Additional information

Blazor WebAssembly App C# Linux macOS Windows Cloud Web

Framework ⓘ
.NET 6.0 (Preview)

Authentication type ⓘ
None

Configure for HTTPS ⓘ
 ASP.NET Core hosted ⓘ
 Progressive Web Application ⓘ

Back Create

Abbildung: Weitere Einstellungen für die neue Blazor WebAssembly-Anwendung

Nach dem Anlegen des Projekts sehen Sie in Visual Studio eine Projektmappe mit einem Projekt (siehe rechts in der nächsten Abbildung). Darin gibt es einen Ordner "wwwroot" mit CSS- und Grafik-Dateien (u.a. Twitter Bootstrap und Open Iconic) sowie einer statischen index.html. Die

Razor Components (.razor-Dateien) der einfachen Beispielanwendung von Microsoft finden Sie unter /Pages und /Shared. Der Startcode der Anwendung liegt in Program.cs und App.Razor.

_Imports.Razor enthält die Einbindungen von Namensräumen mit @using, die für alle Razor Component-Dateien gelten.

Nun sollten Sie die Anwendung übersetzen ("Build/Build Solution") und dann starten, entweder mit "Debug/Start Debugging" (Taste F5) oder "View in Browser" im Kontextmenü des Projekts. Wenn alles auf Ihrem PC korrekt arbeitet, sehen Sie die Webanwendung mit drei Menüpunkten (links in der Abbildung).

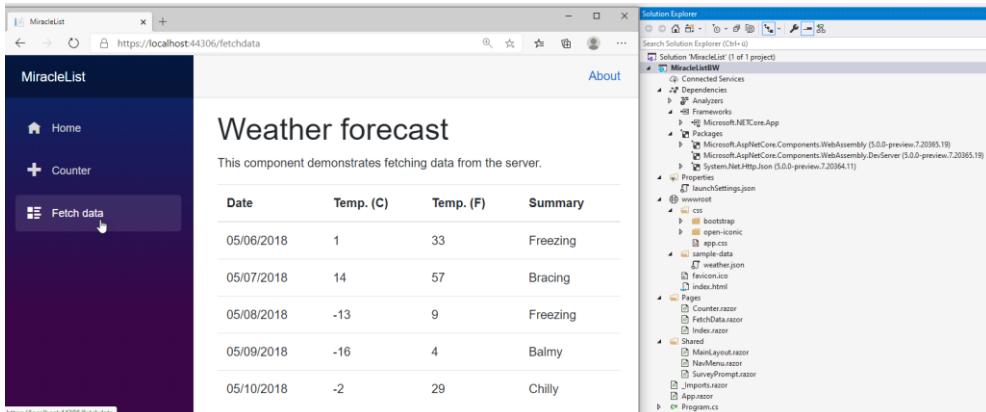


Abbildung: Beispielanwendung von Microsoft

Hinweis: Es gibt leider mannigfaltige Gründe, warum es auf Ihrem System zu einem Fehler kommt, den der Autor dieses Buch nicht hatte. Microsoft hat leider in jeder Version von Visual Studio immer neue Fehler eingebaut. Falls Sie einen Fehler bekommen, recherchieren Sie im Internet oder wenden Sie sich an den Support von Microsoft. Falls Sie Hilfe durch mich als Autor wünschen, nutzen Sie bitte den Support von www.IT-Visions.de/Support.

6.9 Util.Log()

In den Beispielen in diesem Buch kommt immer wieder der Aufruf Util.Log() zur Ausgabe in das Konsolenfenster des Webbrowsers vor. Dies ist keine vorgefertigte Blazor-Funktion, sondern eine vom Autor dieses Buchs selbstdefinierte Hilfsroutine, die anders als das eingebaute Console.WriteLine() sowohl in Blazor WebAssembly als auch Blazor Server funktioniert. Die Implementierung finden Sie im Kapitel "Tipps & Tricks".

6.10 Qualitätssicherung der Programmcodebeispiele

Ich versichere Ihnen, dass die Programmcodebeispiele auf zwei meiner Entwicklungssysteme kompilierten und liefen, bevor ich sie per Kopieren & Einfügen in das Manuskript zu diesem Buch übernommen habe und auf der Leser-Website zum Download veröffentlicht habe.

```
Output
Show output from: Build [x] [y] [z] [a] [b] [c] [d] [e]
1>UnitTests -> H:\ML\MiracleListBackend_TFS\test\UnitTests\bin\Debug\netcoreapp2.0\UnitTests.dll
14>MiracleList_Backend -> H:\ML\MiracleListBackend_TFS\src\MiracleList_Backend\bin\Debug\netcoreapp3.1\Miraclelist_WebAPI.dll
10>MiracleList_CS8 -> H:\ML\MiracleListBackend_TFS\src\MiracleList_CS8\bin\Debug\netstandard2.1\MiracleList_CS8.dll
14>MiracleList_Backend -> H:\ML\MiracleListBackend_TFS\src\MiracleList_Backend\bin\Debug\netcoreapp3.1\Miraclelist_WebAPI.Views.dll
14>Done building project "Miraclelist_Backend.csproj".
13>MiracleList_SS8 -> H:\ML\MiracleListBackend_TFS\src\MiracleList_SS8\bin\Debug\netcoreapp3.1\Miraclelist_SS8.dll
13>MiracleList_SS8 -> H:\ML\MiracleListBackend_TFS\src\MiracleList_SS8\bin\Debug\netcoreapp3.1\MiracleList_SS8.Views.dll
13>Done building project "Miraclelist_SS8.csproj".
11>EFTTools -> H:\ML\MiracleListBackend_TFS\src\EFTTools\bin\Debug\netcoreapp2.0\EFTTools.dll
10>MiracleList_CS8 (Blazor output) -> H:\ML\MiracleListBackend_TFS\src\MiracleList_CS8\bin\Debug\netstandard2.1\dist
10>Done building project "Miraclelist_CS8.csproj".
===== Rebuild All: 14 succeeded, 0 failed, 0 skipped ======
```

Abbildung: Beweis, dass alle Projekte in Visual Studio fehlerfrei übersetzen

Dennoch gibt es leider Gründe, warum die Beispiele bei Ihnen als Leser dieses Fachbuchs nicht laufen:

- Eine abweichende Systemkonfiguration (in der heutigen komplexen Welt der vielen Varianten und Versionen von Betriebssystemen und Anwendungen nicht unwahrscheinlich). Es ist einem Autor nicht möglich, alle Konfigurationen durchzutesten.
- Änderungen, die sich seit der Erstellung der Beispiele ergeben haben (von den vielen Breaking Changes, die ASP.NET Core immer wieder durch Microsoft erhält, können auch Beispiele betroffen sein, was nicht immer leicht zu entdecken ist).
- Schließlich sind auch menschliche Fehler des Autors möglich. Bitte bedenken Sie, dass das Fachbuchschreiben – wie im Vorwort erwähnt – nur ein Hobby ist. Es gibt nur sehr wenige Menschen in Deutschland, die hauptberuflich als Fachbuchautor arbeiten und so professionell Programmcodebeispiele erstellen und testen können wie kommerziellen (bezahlten) Programmcode.

Falls dennoch Beispiele bei Ihnen nicht laufen, kontaktieren Sie mich bitte unter

www.dotnet-doktor.de/Leserfeedback

mit einer sehr genauen Fehlerbeschreibung und Ihrer genauen Systemkonfiguration. Ich bemühe mich, Ihnen binnen zwei Wochen zu antworten. Im Einzelfall kann es wegen dienstlicher oder privater Abwesenheit aber auch länger dauern.

7 Was ist Blazor?

ASP.NET Core Blazor (kurz: Blazor) ist ein Webentwicklungsframework von Microsoft zur Entwicklung von Single-Page-Web-Applications (SPAs). Es basiert auf Oberflächen, die mit Hypertext Markup Language (HTML) und Cascading Style Sheets (CSS) gestaltet werden.

7.1 Blazor-Arten

Es gibt derzeit drei Arten von Blazor mit verschiedenen Softwarearchitekturmodellen:

- **Blazor WebAssembly** (alias: Client-Side Blazor): Hier läuft der .NET-Programmcode und das HTML-Rendering im Webbrower in dessen WebAssembly-Laufzeitumgebung. Eine von Microsoft entwickelte JavaScript-Bibliothek kommt hier dennoch auch zum Einsatz, um zwischen der WebAssembly-VM des Browsers und dem Document Object Model des Webbrowsers (DOM) zu synchronisieren, denn die WebAssembly-VM des Browsers hat laut W3C-Standard keinen Zugriff auf dessen DOM. Konkret bedeutet dies: Jedes Ereignis im Browser sendet die JavaScript-Bibliothek an die Blazor WebAssembly-Anwendung. Diese kann auf Ereignisse mit der Ausführung von C#-Code oder sogenannten Razor-Templates (mit Platzhaltern in der Syntax @xy) reagieren. C#-Code und Templates manipulieren aber mangels Zugang nicht das echte DOM, sondern eine Kopie davon innerhalb der WebAssembly-VM, die "Virtual DOM" genannt wird. Der vordefinierte Code von Microsoft synchronisiert die Änderungen auf das echte DOM und ändert so die sichtbare Oberfläche im Webbrower.
- **Blazor Server** (alias: Server-Side Blazor): Hier läuft der .NET-Programmcode auf dem Webserver. Eine JavaScript-Bibliothek agiert im Browser als Gegenpart. Auch hier gibt es ein Virtual DOM, das aber auf dem Webserver liegt. Die Synchronisierung zwischen DOM im Webbrower und Virtual DOM auf dem Webserver erfolgt hier in einzelnen Datenpaketen mit ASP.NET Core SignalR via Netzwerk über eine kontinuierliche Websocket-Verbindung. Dafür kommt ein kompaktes Synchronisierungsverfahren zum Einsatz. Wie bei Blazor WebAssembly aktualisiert auch bei Blazor Server die JavaScript-Bibliothek die Oberfläche und sendet Ereignisse zurück an Blazor (in diesem Fall über das Netzwerk).
- **Blazor Desktop:** Hier läuft die Blazor-Anwendung in einer nativen Anwendung direkt auf dem Betriebssystem. Microsoft unterstützt hier als Host-Frameworks Windows Forms (für Windows), Windows Presentation Foundation (für Windows) sowie .NET MAUI (für Windows, Android, iOS und macOS). Die Blazor-Anwendung wird ihrer HTML-/CSS-basierten Oberfläche im WebView-Steuerelement der jeweiligen Platform gehostet und hat alle Zugänge zu Ressourcen, Netzwerk und APIs wie eine native Anwendung.

Hinweis: Manche Entwickler denken, "Blazor Server" wäre der Server zu einer Blazor WebAssembly-Anwendung. Dies ist falsch. Blazor Server ist ein eigenes Softwarearchitekturmodell für die Entwicklung von Single-Page-Web-Applications.

Den wesentlichen Unterschied zwischen Blazor WebAssembly und Blazor Server veranschaulicht die nachstehende Abbildung von Microsoft: Während bei Blazor Server der .NET-Programmcode auf dem Server läuft und eine kontinuierliche Netzwerübertragung aller Ereignisse und DOM-Änderungen zwischen Webserver und Webbrower stattfindet, läuft bei Blazor WebAssembly der .NET-Code im Webbrower. Details zu den Unterschieden beider Modelle werden Sie in weiteren Unterkapiteln erfahren.