

Dr. Holger Schwichtenberg

Blazor 9.0

**Moderne Webanwendungen und
hybride Cross-Platform-Apps mit
.NET 9.0, C# 13.0 und Visual Studio 2022**



Buchversion: 9.0.0 vom 01.11.2024
Verlag: www.IT-Visions.de, Fahrenberg 40b, D-45257 Essen
Sprachliche Korrektur: Dorothea Fleischer, Heike Rickert, Matthias Bloch (M.A.)
ISBN: 978-3-934-27939-1
Bezugsquellen: www.IT-Visions.de/Buch/CS13

Für Heidi, Maja und Felix

1 Inhaltsverzeichnis (Hauptkapitel)

1	Inhaltsverzeichnis (Hauptkapitel)	4
2	Inhaltsverzeichnis (Details)	5
3	Vorwort.....	19
4	Über den Autor	21
5	Über dieses Fachbuch	23
6	Programmcodebeispiel zum Download	29
7	Überblick über Blazor.....	54
8	Blazor-Arten im Detail	72
9	Projektaufbau von Blazor-Webanwendungen.....	113
10	Übersetzung und Debugging.....	149
11	Komponenten (Razor Components).....	166
12	Routing und Navigation.....	185
13	Razor-Syntax	206
14	Ereignisse und Ereignisbehandlung	216
15	Komponenteneinbettung / Unterkomponenten	228
16	Zustandsverwaltung	260
17	Formulare/Eingabemasken und Datenbindung	279
18	Klassenbibliotheken und Razor Class Libraries (RCL)	317
19	Dependency Injection (DI)	335
20	Interoperabilität zwischen .NET und JavaScript.....	346
21	Zugriff auf Webservices/WebAPIs	379
22	Benachrichtigungen (Push-Nachrichten) mit ASP.NET Core SignalR in Blazor-Apps	398
23	System- und Browserinformationen	413
24	Authentifizierung und Benutzerverwaltung	434
25	Komponentenbibliotheken von Drittanbietern	465
26	Installation von Blazor-Anwendungen	482
27	Tipps, Tricks und Tools	498
28	Static Server-Side-Rendering (SSR).....	590
29	Hybride Apps mit Blazor Desktop.....	617
30	Hybride Apps mit Blazor MAUI	634
31	Fallbeispiel "MiracleList"	642
32	Umstieg von ASP.NET Webforms zu Blazor	780
33	Razor Components außerhalb von Webanwendungen.....	799
34	WebAssembly ohne Blazor.....	801
35	Quellen im Internet	806
36	Versionsgeschichte dieses Buchs.....	807
37	Stichwortverzeichnis (Index)	808
38	Werbung in eigener Sache ☺	822

2 Inhaltsverzeichnis (Details)

1	Inhaltsverzeichnis (Hauptkapitel).....	4
2	Inhaltsverzeichnis (Details).....	5
3	Vorwort	19
4	Über den Autor.....	21
5	Über dieses Fachbuch.....	23
5.1	Versionsgeschichte dieses Buchs	23
5.2	Hinweis zu den Vertriebswegen	23
5.3	Bezugsquelle des PDF-E-Books für Amazon-Kunden	23
5.4	Bezugsquelle für Aktualisierungen	24
5.5	Hinweise zur Breite und Tiefe dieses Buchs – Sie haben Einfluss!	24
5.6	Geplante Kapitel	24
5.7	Programmiersprache in diesem Buch.....	26
5.8	Notwendige Vorkenntnisse	26
5.9	Vermeidung des Einsatzes kostenpflichtiger Software	27
5.10	Hinweis zur Erwähnung der Blazor-Varianten in diesem Buch.....	28
5.11	Ihre Belohnung, wenn Sie helfen, dieses Buch zu verbessern!	28
6	Programmcodebeispiel zum Download.....	29
6.1	Webadresse für Downloads	29
6.2	Übersicht über die Beispiele	29
6.3	Eingesetztes CSS-Framework.....	32
6.4	Das Fallbeispiel "MiracleList"	32
6.4.1	Szenario	32
6.4.2	MiracleList-Client-Features.....	33
6.4.3	Technische Basis für MiracleList	34
6.4.4	Webadressen.....	35
6.4.5	Projektmappenaufbau	35
6.4.6	MiracleList-Bildschirmmasken	37
6.5	Weitere Beispiele innerhalb der MiracleList-Webanwendung	43
6.6	Visual Studio 2022.....	44
6.6.1	Varianten von Visual Studio.....	44
6.6.2	Stabile Version und Preview-Version	45
6.6.3	Workloads	46
6.6.4	Sprachversionen	46
6.7	.NET SDK.....	47
6.8	.NET WebAssembly Build Tools	49
6.9	Testen Ihrer PC-Konfiguration	50
6.10	Util.Log().....	52
6.11	Qualitätssicherung der Programmcodebeispiele	53
7	Überblick über Blazor	54
7.1	.NET 9.0.....	54
7.2	ASP.NET Core.....	54
7.3	ASP.NET Core Blazor oder einfach Blazor?	55

7.4	Überblick über die Blazor-Arten	55
7.5	Blazor United.....	59
7.6	Geschichte von Blazor.....	60
7.7	Marktverbreitung von Blazor	62
7.8	Technischer Support für Blazor.....	63
7.8.1	Allgemeine .NET-Support-Richtlinie	64
7.8.2	Aktuelle Blazor-Versionen mit Support.....	65
7.9	Betriebssysteme/Plattformen	65
7.10	Browserunterstützung für Blazor.....	66
7.10.1	Browserunterstützung für Blazor Web-Anwendungen	66
7.10.2	Browserunterstützung für Blazor Static SSR	67
7.10.3	Browserunterstützung für Blazor Hybrid	67
7.10.4	Einsatz von JavaScript in Blazor.....	67
7.11	Webserverunterstützung für Blazor	67
7.12	Voraussetzungen für Blazor auf dem Entwicklungsrechner	68
7.13	Voraussetzungen für Blazor auf dem Zielsystemen	68
7.14	NuGet-Pakete für Blazor	69
7.15	Neuerungen in Blazor 9.0	70
7.15.1	Projekteinstellungen für Blazor 9.0.....	70
7.15.2	Neue Funktionen in Blazor 9.0	70
7.16	Verschobene Features.....	70
8	Blazor-Arten im Detail	72
8.1	Blazor Server (Interactive Server-Side Rendering).....	72
8.1.1	Rendering und Interaktion bei Blazor Server.....	72
8.1.2	Softwarearchitektur.....	73
8.1.3	Netzwerkverkehr bei Blazor Server	73
8.1.4	Ressourcenbedarf und Skalierbarkeit	74
8.1.5	Vor- und Nachteile von Blazor Server	75
8.2	Blazor WebAssembly (Client-Side-Rendering).....	77
8.2.1	Konzept von Blazor WebAssembly	77
8.2.2	WebAssembly (WASM)	78
8.2.3	JITerpreter und AOT.....	79
8.2.4	SIMD für Blazor WebAssembly	80
8.2.5	WebCIL-Format für Assemblies in Blazor WebAssembly	80
8.2.6	Sandbox des Browsers / API-Einschränkungen	81
8.2.7	Anwendungsgröße und Startzeiten.....	84
8.2.8	Speicherlimit	88
8.3	Auto-Render-Modus (Blazor United).....	89
8.4	Blazor Static Server-Side-Rendering (Blazor Static SSR).....	92
8.4.1	Vergleich mit ASP.NET Core MVC und ASP.NET Razor Pages	93
8.4.2	Warum Blazor Static-SSR?	93
8.5	Blazor Hybrid (Blazor Desktop / Blazor MAUI).....	96
8.5.1	Hybride Apps	96
8.5.2	Blazor Desktop	97
8.5.3	Blazor MAUI	99

8.6	Vergleich der Blazor-Arten.....	99
8.7	Performance-Vergleich	103
8.8	Fehlende Funktionen in Blazor 9.0	106
8.9	Blazor im Vergleich zu anderen Weboberflächen mit ASP.NET Core.....	107
8.10	Blazor im Vergleich zu anderen Webframeworks	108
8.11	Exkurs: Uno Platform	111
9	Projektaufbau von Blazor-Webanwendungen	113
9.1	Übersicht über die Blazor-Projektvorlagen.....	113
9.2	Die Webanwendung in der Standard-Projektvorlage	115
9.3	Neue Projektvorlage "Blazor Web App"	116
9.4	Interaktivitätstyp "None" (Static Server-Side-Rendering)	119
9.4.1	Inhalt der Projektvorlage	119
9.4.2	Struktur des Seitenaufbaus	121
9.4.3	Startcode in Program.cs.....	122
9.4.4	Darstellung der Projektvorlage	123
9.4.5	Anpassung der Vorlage: Dynamische Fußzeile.....	124
9.4.6	Streaming beim Static Server-Side-Rendering	124
9.4.7	Anpassung der Vorlage: Komponenten einbetten	129
9.5	Interaktivitätstyp "Server" (Blazor Server)	129
9.5.1	Interaktivitätstyp "Server" mit Location "Per Page/Component"	130
9.5.2	Interaktiven Inseln in Seiten einbetten.....	130
9.5.3	Interaktivitätstyp "Server" mit Location "Global"	131
9.5.4	Tipp: Genauere Fehlermeldungen aktivieren (Detailed Errors).....	131
9.6	Interaktivitätstyp "WebAssembly" (Blazor WebAssembly)	132
9.7	Interaktivitätstyp "Auto" (Wechsel von Blazor Server zu Blazor WebAssembly)	135
9.8	Mischung der Render-Modi	136
9.9	Veraltete Projektvorlage "Blazor Server"	138
9.9.1	Migration eines bestehenden Blazor Server-Projekts nach Blazor 9.0	138
9.9.2	Migration von einer Blazor Server-Anwendung zur Blazor Web App.....	138
9.10	Projektvorlage "Blazor WebAssembly Standalone App"	139
9.10.1	Projekte.....	140
9.10.2	Projektdatei.....	141
9.10.3	Ordner und Dateien	142
9.10.4	Aufbau der index.html.....	143
9.10.5	Start-Code.....	143
9.10.6	Inhaltsseiten.....	144
9.11	Veraltete Projektvorlage "Blazor WebAssembly"	145
9.12	Projektvorlagen für Blazor Hybrid.....	146
9.13	Anlegen einer kompletten mehrschichtigen Projektstruktur mit der .NET CLI	146
10	Übersetzung und Debugging	149
10.1	Übersetzung	149
10.2	Start von Blazor-Projekten in Visual Studio und per Kommandozeile	150
10.2.1	Aktualisierungen im Webbrowser	150
10.2.2	Webserver und Endpunkte festlegen	151
10.2.3	Webbrowser festlegen	152

10.2.4	HTTPS	153
10.3	Hot Reloading (seit .NET 6.0)	153
10.4	Visual Studio-Debugging in Blazor Server- und Blazor Hybrid-Projekten	158
10.5	Visual Studio-Debugging in Blazor WebAssembly-Projekten	159
10.6	Ablaufverfolgung des ASP.NET Core Webservers	164
10.7	Browser-Debugging in Blazor WebAssembly	164
10.8	Veröffentlichen von Blazor-Anwendungen	165
11	Komponenten (Razor Components)	166
11.1	Einsatz von Razor Components	166
11.2	Namen für Komponenten	166
11.3	Grundkonzepte am Beispiel einer "Hello World"-Komponente	166
11.4	Markup-Dateien mit Inline-Code	168
11.5	Komponenten mit Code-Behind-Datei	170
11.5.1	Code-Behind-Dateien auf Basis von Vererbung	171
11.5.2	Code-Behind-Dateien auf Basis von partiellen Klassen	173
11.6	Komponenten ohne Markup nur mit Programmcode	175
11.7	CSS-Isolation (seit Blazor 5.0)	177
11.7.1	Scoped Styles anlegen	178
11.7.2	Scoped Styles einbinden	178
11.7.3	CSS-Isolation mit Razor Class Libraries	179
11.8	Nicht-visuelle Komponenten	179
11.9	Layoutseiten (Masterpages)	181
11.10	Sektionen in Layoutseiten	184
12	Routing und Navigation	185
12.1	Routendefinitionen	185
12.2	Keine Reaktion auf HTML-Ereignisse bei Blazor Static SSR	186
12.3	Routen mit Parametern	187
12.4	Navigation	190
12.4.1	Navigation per <a>-Tag	190
12.4.2	URL-Fragmente	191
12.4.3	Komponente <NavLink>	191
12.4.4	Navigation im Code	192
12.5	Übergabe von Werten per Browser History Stack	194
12.6	Auswerten der aktuellen URL	195
12.6.1	Praxisbeispiel: Fallunterscheidung bei Mehrfachrouten	196
12.6.2	Query Strings	196
12.6.3	Komplette URL-Auswertung	197
12.7	Verhindern der Navigation im Browser (seit Blazor 7.0)	200
12.8	Routerkonfiguration	204
13	Razor-Syntax	206
13.1	Unterschiede zwischen Razor in Blazor und Razor in MVC und Razor Pages	206
13.2	Ausdrücke vs. Befehlsfolgen	210
13.3	Asynchrone Ausdrücke in der Razor-Syntax	211
13.4	Praxisaufgabe zur Razor-Syntax	212
13.4.1	Aufgabenstellung	212

13.4.2	Lösung.....	213
13.5	Problemlösung bei der Änderungsfeststellung in Schleifen mit @key	214
14	Ereignisse und Ereignisbehandlung	216
14.1	Komponentenlebenszyklusereignisse	216
14.2	Reaktion auf HTML-Ereignisse	217
14.3	Eigene Parameter bei HTML-Ereignissen	218
14.4	Inline-Ereignisbehandlung (ohne Ereignisbehandlungsmethode).....	219
14.5	Standardparameter	220
14.6	Standardereignisbehandlung unterbinden	222
14.7	Asynchrone Ereignisbehandlung	223
14.8	Ereignisweitergabe unterbinden.....	223
14.9	Komplexe Befehlsfolgen als Ereignisbehandlung	225
14.10	Vermeiden Sie JavaScript-Ereignisbehandlungen!	226
14.11	Keine Reaktion auf HTML-Ereignisse bei Blazor Static SSR	227
15	Komponenteneinbettung / Unterkomponenten.....	228
15.1	Einbetten von Razor Components.....	228
15.2	Gefahr: Vergessene Namensraumimporte	228
15.3	Leistungseinbußen durch Komponentenbildung/Optimierung durch Inlining	230
15.4	Komponenten umbenennen.....	230
15.5	Parameter für eingebettete Razor Components	231
15.5.1	Setzen von Parameter in der Host-Komponente.....	231
15.5.2	Ereignisparameter	231
15.5.3	Parameter mit Value Type	232
15.5.4	Parameter mit Reference Type	235
15.5.5	Pflichtparameter für Komponenten (seit Blazor 6.0).....	237
15.6	Komponentenparameter aus Querystring (seit .NET 6.0)	237
15.7	Kaskadierende Parameter.....	239
15.8	Globale kaskadierende Blazor-Parameter (seit Blazor 8.0).....	242
15.9	Vorlagenbasierte Komponenten (Templated Components)	243
15.9.1	Einfache vorlagenbasierte Komponente	243
15.9.2	Vorlagenbasierte Komponente mit mehreren Fragmenten	244
15.9.3	Datenübergabe an Renderfragment via Kontext.....	245
15.9.4	Komplexe Objekte als Kontext.....	246
15.9.5	Vorlagenbasierte Komponente mit generischem Typparameter	247
15.9.6	Verschachtelung von Renderfragmenten	248
15.9.7	Standardwerte für Renderfragmente	249
15.9.8	Praxislösung: Repeater	250
15.10	Weitergabe von HTML-Attributen	252
15.11	Dynamic Component (seit .NET 6.0)	255
15.12	Error Boundaries (seit .NET 6.0)	257
16	Zustandsverwaltung	260
16.1	Inhalt des Komponentenzustands.....	260
16.2	Lebensdauer des Komponentenzustands.....	260
16.3	Testanwendung: Counter	260
16.4	Verlust des Zustandes	261

16.5	Bewahrung des Zustandes	263
16.6	Sitzungszustand	263
16.7	Generischer Sitzungszustand	265
16.8	Nutzung des Webbrowserspeichers	266
16.8.1	Verschlüsselter Browserspeicher (Protected Storage) für Blazor Server	267
16.8.2	NuGet-Pakete für den Browserspeicher in Blazor WebAssembly und Blazor Hybrid	269
16.8.3	Einsatz von Blazored.LocalStorage	269
16.9	Cookies	273
16.10	Persistierung in Datenbanken oder anderen persistenten Speichern auf dem Server	274
16.11	Persistent Component State	275
17	Formulare/Eingabemasken und Datenbindung	279
17.1	Formulare auf Basis der Blazor-Eingabekomponenten	279
17.1.1	Verfügbare Blazor-Eingabekomponenten	279
17.1.2	Komponente <EditForm>	280
17.1.3	Datenbindung	281
17.1.4	Type beim Steuerelement <InputNumber>	282
17.1.5	Datenannotationen	282
17.1.6	Validierungskomponenten in Blazor	284
17.1.7	Codebasierte Validierung	284
17.1.8	Eigene Validierungsannotationen	284
17.1.9	Fluent Validation	285
17.1.10	Praxisbeispiel	286
17.1.11	Optionsfelder (<InputRadio> und <InputRadioGroup>)	291
17.1.12	Dateien hochladen (<InputFile>)	293
17.1.13	Zwei-Wege-Datenbindung mit eigenen Komponenten	294
17.2	Formulare auf Basis von Standard-HTML-Steuerelementen	295
17.2.1	Datenbindung	295
17.2.2	Reaktion auf einzelne Buchstabeneingaben	296
17.2.3	Zusammengesetzte Eingaben (KeyboardEventArgs.IsComposing)	300
17.2.4	Auswahlfelder (<select>)	301
17.2.5	Datenbindung mit Datumsangaben	304
17.2.6	Praxisbeispiel	306
17.3	Tipps & Tricks zu Formularen	310
17.3.1	Direkter Objektverweis	310
17.3.2	Fokus setzen	310
17.3.3	Programmcode ausführen im Rahmen der Datenbindung	311
17.4	Formulare bei Blazor Static SSR	314
18	Klassenbibliotheken und Razor Class Libraries (RCL)	317
18.1	.NET Standard	317
18.2	Referenzierbarkeit	318
18.3	Anlegen von Klassenbibliotheken	319
18.4	Referenzieren von Klassenbibliotheken	320
18.5	Einsatzgebiete der verschiedenen Klassenbibliotheksarten	321
18.6	Razor Class Libraries (RCL)	322
18.6.1	Bedarf für Code-Sharing zwischen Blazor-Projekten	322

18.6.2 Softwarearchitektur mit Razor Class Library	322
18.6.3 Erstellen einer Razor Class Library	323
18.6.4 Inhalte einer Razor Class Library	324
18.6.5 Referenzierung einer Razor Class Library	325
18.6.6 Einbettung von Razor Components aus einer Razor Class Library	328
18.6.7 Routing zu Razor Components aus einer Razor Class Library	329
18.6.8 Nutzung von statischen Ressourcen aus einer Razor Class Library	330
18.7 Lazy Loading von Assemblies bei Blazor WebAssembly	331
18.7.1 Integration von Lazy Loading in die Standardprojektvorlage	332
18.7.2 Integration von Lazy Loading in das MiracleList-Fallbeispiel	333
19 Dependency Injection (DI)	335
19.1 Microsoft.Extensions.DependencyInjection	335
19.2 Implementierung eines Dienstes	335
19.3 Lebensdauer von Instanzen	336
19.4 Registrierung von Diensten	337
19.5 Automatisch registrierte Dienste	337
19.6 Injektion im Template mit <code>@inject</code>	342
19.7 Injektion in Properties mit <code>[Inject]</code>	342
19.8 Dependency Injection im Konstruktor von Razor Components	343
19.9 Injektion in normale Klasse	343
19.10 Manuelle Beschaffung von Instanzen	344
19.11 Geschlüsselte Dienste bei der Dependency Injection (seit .NET 8.0)	345
20 Interoperabilität zwischen .NET und JavaScript	346
20.1 Motivation	346
20.2 Einschränkung der JavaScript-Interoperabilität in Blazor Server	346
20.3 Aufruf von C# zu JavaScript	347
20.4 Eigene JavaScript-Dateien verwenden	348
20.4.1 Globale Skripteinbindung	348
20.4.2 JavaScript-Dateien nachladen (JavaScript-Isolation / <code>IJSObjectReference</code>)	351
20.5 Aufruf von JavaScript zu C#	353
20.5.1 Aufruf von JavaScript zu C# über statische Methoden	353
20.5.2 Praxisbeispiel	354
20.5.3 Aufruf von JavaScript zu C# (Instanzmethoden)	357
20.6 Praxisbeispiel: Nutzung einer JavaScript-Bibliothek am Beispiel Chartist.js	358
20.7 Praxisbeispiel: Integration mit Angular über Web Components	362
20.7.1 Implementierung einer Web Component mit Angular Elements	362
20.7.2 Nutzung der Web Component in JavaScript	364
20.7.3 Einbindung von Web Components in Blazor	365
20.8 Praxisbeispiel: Angular-Datagrid in Blazor	367
20.8.1 Implementierung der Web Component in Angular	367
20.8.2 Nutzung der Web Component in JavaScript	370
20.8.3 Einbindung der Web Component <code><angular-grid></code> in Blazor	370
20.9 Streaming zwischen .NET und JavaScript (seit .NET 6.0)	374
20.10 JavaScript-Initializer (seit .NET 6.0)	376
20.11 <code>[JSImport]</code> und <code>[JSExport]</code> für Blazor WebAssembly (seit Blazor 7.0)	376

21	Zugriff auf Webservices/WebAPIs	379
21.1	Daten- und Ressourenzugriffe in Blazor WebAssembly	379
21.2	Daten- und Ressourenzugriffe in Blazor Server und Blazor Hybrid.....	380
21.3	Webservice-Arten.....	380
21.4	Szenario	380
21.5	REST-Dienste (WebAPIs).....	381
21.5.1	WebAPIs erstellen mit ASP.NET Core.....	382
21.5.2	Neue Blazor-Projekte inklusive ASP.NET Core WebAPI beginnen.....	383
21.5.3	WebAPIs testen mit Postman.....	383
21.5.4	Zugriff auf Web APIs in Blazor	384
21.5.5	Cross-Origin Resource Sharing für das WebAPI	386
21.5.6	Metadaten mit Open API Specification.....	386
21.5.7	Client-Generierung aus Metadaten mit Visual Studio.....	387
21.5.8	Client-Generierung aus Metadaten mit NSwagStudio	389
21.6	gRPC-Dienste	391
21.6.1	gRPC-Dienst erstellen.....	391
21.6.2	gRPC-Dienste testen mit Postman	394
21.6.3	Blazor-Zugriff auf gRPC-Dienste	394
21.7	Leistungsvergleich zwischen WebAPI und gRPC.....	395
21.8	SOAP und die Windows Communication Foundation (WCF)	396
22	Benachrichtigungen (Push-Nachrichten) mit ASP.NET Core SignalR in Blazor-Apps	398
22.1	Transport- und Serialisierungsoptionen in SignalR	398
22.2	SignalR-Plattformen	398
22.3	Implementierung eines SignalR-Hubs im Webserver.....	400
22.4	Einbinden eines SignalR-Hubs	404
22.5	Typisierter SignalR-Hub.....	404
22.6	Aktivieren der Websockets-Unterstützung.....	405
22.7	Implementierung eines SignalR-Hub-Connection im Blazor-Client	407
22.8	Automatisches Wiederherstellen einer Verbindung	410
22.9	Nachrichten senden im SignalR-Client.....	410
22.10	Überwachung und Diagnose von SignalR-Verbindungen	411
23	System- und Browserinformationen	413
23.1	Ermitteln der Blazor-Versionsnummer.....	413
23.2	Allgemeine Systeminformationen	413
23.3	Ermittlung des aktuellen Blazor-Render-Modus	423
23.4	Ermittlung von Webserver-Informationen bei Blazor Server	427
23.5	HttpContext in Blazor Server	429
23.6	HttpContext in Blazor Static SSR.....	431
23.7	Host Environment-Informationen.....	431
23.8	Verwenden von Umgebungsnamen	432
24	Authentifizierung und Benutzerverwaltung	434
24.1	Authentifizierung in Blazor Web Apps mit ASP.NET Core Identity	434
24.1.1	Inhalt der Projektvorlage.....	435
24.1.2	Anlegen der Datenbank.....	438
24.1.3	Konfiguration des E-Mail-Versandes	440

24.1.4 Webbasierte Oberfläche von ASP.NET Core Identity.....	441
24.1.5 Anpassung der Benutzerverwaltung	444
24.1.6 Übergabe des Authentifizierungszustandes	445
24.2 Authentifizierung in Blazor WebAssembly Standalone Apps	445
24.2.1 Individuelle Konten (OIDC-Authentifizierung)	445
24.2.2 Microsoft Identity Platform	448
24.2.3 Weitere OIDC-Optionen	449
24.2.4 Individual Accounts.....	450
24.3 Eigene Authentifizierungsprovider (AuthenticationStateProvider)	453
24.3.1 AuthenticationStateProvider für Debugging-Zwecke.....	453
24.3.2 AuthenticationStateProvider in MiracleList	454
24.3.3 Nutzung des eigenen AuthenticationStateProvider	460
24.4 Autorisierung	461
24.4.1 Zugriff auf den angemeldeten Benutzer	461
24.4.2 Autorisierung von Komponenten	462
24.4.3 Inhalte für angemeldete Benutzer	462
24.4.4 Umleitung für nicht-autorisierte Benutzer.....	462
25 Komponentenbibliotheken von Drittanbietern	465
25.1 Überblick über die Anbieter (Auswahl)	465
25.2 Einsatz der Radzen Blazor Components	465
25.2.1 Komponenten in den Radzen Blazor Components	465
25.2.2 Bezugsquelle	466
25.2.3 Erstellen eines Projekts mit den Radzen-Komponenten	466
25.2.4 Beispiel 1: Artikelverwaltung.....	467
25.2.5 Beispiel 2: MiracleList	471
25.3 Einsatz von Telerik UI for Blazor	471
25.3.1 Komponenten in Telerik UI for Blazor	471
25.3.2 Bezugsquelle und Preise.....	472
25.3.3 Installation von Telerik UI for Blazor	472
25.3.4 Erstellen eines Projekts mit Telerik UI for Blazor.....	473
25.3.5 Integration von Telerik UI for Blazor in bestehende Blazor-Projekte	476
25.3.6 Beispiel.....	477
25.4 Einsatz von DevExpress UI for Blazor	477
25.4.1 Komponenten in DevExpress UI for Blazor.....	478
25.4.2 Bezugsquelle und Preise.....	478
25.4.3 Installation von DevExpress UI for Blazor.....	479
25.4.4 Erstellen eines Projekts mit DevExpress UI for Blazor	480
25.4.5 Beispiel.....	481
26 Installation von Blazor-Anwendungen.....	482
26.1 Deployment-Profile.....	482
26.2 Deploymentarten.....	483
26.3 Installation auf Azure-Diensten	485
26.3.1 Wahl der .NET-Version und Deployment-Methode.....	485
26.3.2 WebSockets.....	485
26.3.3 Brotli-Komprimierung.....	486

26.3.4	Problembehandlung für Fehler	487
26.4	Installationsvoraussetzungen für eigene Windows-Webserver (IIS)	488
26.5	Installation auf einem eigenen Windows-Webserver (IIS)	489
26.6	Self-Hosting (Kestrel Webserver)	493
26.7	Deployment in Docker-Container.....	494
26.8	Ahead-of-Time-Kompilierung (AOT) in Blazor WebAssembly (seit .NET 6.0)	495
27	Tipps, Tricks und Tools	498
27.1	Ausgaben in die Browserkonsole	498
27.2	HTML-Kopfdaten beeinflussen.....	501
27.3	Optimierungen für statische Webressourcen	503
27.4	BlazorFiddle	505
27.5	Radzen	506
27.6	Integration von ASP.NET Core MVC und ASP.NET Razor Pages zu Blazor	511
27.7	Anwendungskonfigurationsdateien	511
27.7.1	AppSettings.json	512
27.7.2	Listen in Konfigurationseinträgen.....	513
27.7.3	Andere Konfigurationsformate	514
27.8	Steuerung des Rendering	515
27.9	Leistungsoptimierung beim Rendern großer Datenmenge durch Virtualisierung.....	518
27.9.1	Szenario.....	520
27.9.2	Sukzessives Rendern mit <Virtualize>	521
27.9.3	Sukzessives Laden per ItemsProvider-Funktion	523
27.10	QuickGrid und Codegenerierung (Scaffolding).....	523
27.10.1	Das QuickGrid-Steuerelement	524
27.10.2	QuickGrid erstellen per Codegenerierung in Visual Studio	524
27.10.3	QuickGrid anpassen	527
27.10.4	Virtualisierung beim QuickGrid.....	529
27.10.5	Weitere generierte Dateien	530
27.11	Verbesserte Wiederherstellung von Blazor Server-Verbindungen	531
27.12	Überwachung von Blazor Server-Anwendungen	532
27.12.1	Überwachung der Netzwerklatenz bei Blazor Server.....	532
27.12.2	Überwachung der Sitzungen/Circuits bei Blazor Server	534
27.13	Konfiguration der WebSocket-Komprimierung und Content Security Policy (CSP).....	540
27.14	Hintergrundaufgaben und Fortschrittsanzeige	541
27.15	Multi-Threading in Blazor WebAssembly.....	544
27.16	Zeitgesteuerte Aufgaben via Timer	545
27.16.1	Praxisbeispiel: Zeitgesteuerte Seitenaktualisierung	545
27.16.2	Praxisbeispiel: Countdown.....	546
27.16.3	Eigene Timer-Komponente für Blazor	547
27.17	Meldungskomponente	548
27.18	Modale Dialoge mit Bootstrap.....	551
27.19	Kontextmenüs.....	553
27.19.1	Installation der Komponente "BlazorContextMenu".....	554
27.19.2	Aufbau eines Kontextmenüs	554
27.19.3	Kontextmenü in MiracleList	555

27.19.4	Alternative Kontextmenükomponenten	555
27.20	Benachrichtigungen	556
27.20.1	Benachrichtigungen im Browserfenster (Toasts).....	556
27.20.2	Desktop-Benachrichtigungen	557
27.21	Drag & Drop	560
27.22	Zugriff auf die lokale Zeit des Webbrowsers.....	561
27.23	Custom Boot Resource Loading bei Blazor WebAssembly.....	562
27.24	Progressive Web Applications (PWA) mit Blazor WebAssembly	562
27.25	Verzögerter Start einer Blazor WebAssembly-Anwendung	567
27.26	Mehrsprachigkeit (Lokalisierung/Globalisierung/ Internationalisierung)	567
27.26.1	Festlegung der aktuellen Sprache	568
27.26.2	Texte aus Ressourcendateien.....	568
27.26.3	Beispiel.....	568
27.26.4	Notwendige Infrastruktur bei Blazor WebAssembly	572
27.26.5	Notwendige Infrastruktur bei Blazor Server.....	573
27.26.6	Lokalisierung bei Blazor Desktop	574
27.26.7	Darstellung von Datums- und Zahleingabefeldern	575
27.27	Optimierung der Download-Größe bei Blazor WebAssembly.....	575
27.28	Blazor WebAssembly und Firewall/Anti-Virus-Software	577
27.29	Native Abhängigkeiten/Nativen Code aufrufen	580
27.30	Obfuscation von Blazor-Anwendungen.....	583
27.31	Micro-Apps mit Blazor	584
27.31.1	Projektaufbau.....	584
27.31.2	Kommunikation.....	586
28	Static Server-Side-Rendering (SSR)	590
28.1	Vergleich zwischen Blazor Static SSR und Blazor-SPAs.....	590
28.2	Projektvorlage für Blazor Static SSR.....	591
28.3	Ein einfacher Zähler mit Blazor Static SSR.....	592
28.4	Anti-Forgery-Token zum Schutz gegen Cross-Site-Request-Forgery	594
28.5	Zustandsverwaltung per Cookies	595
28.6	Praxisnahes Registrierungsformular mit Blazor Static SSR	596
28.7	Streaming bei Blazor Static SSR	602
28.8	Enhanced Navigation.....	608
28.9	Konfiguration der Enhanced Navigation.....	609
28.10	Komponentenbildung bei Blazor Static SSR	610
28.11	Eigenes JavaScript bei Blazor Static SSR	614
28.12	Statische SSR-Inseln in Single-Page-Apps	615
29	Hybride Apps mit Blazor Desktop	617
29.1	Blazor Desktop auf Windows mit WPF und Windows Forms.....	617
29.2	Projektvorlage	618
29.3	Beispiel: Blazor-Anwendung in Windows Forms.....	618
29.3.1	Projektaufbau.....	621
29.3.2	Windows Forms-Teil der Anwendung	623
29.3.3	Gemeinsamer Zustand	624
29.3.4	Blazor-Teil der Anwendung	625

29.4	Hyperlinks in einer Blazor Desktop-Anwendungen	631
29.5	Integration in WPF	632
29.6	Verteilungsoptionen.....	632
30	Hybride Apps mit Blazor MAUI	634
30.1	Architektur einer Blazor MAUI-Apps	634
30.2	Projektvorlage ".NET MAUI Blazor Hybrid App".....	635
30.3	Projektvorlage ".NET MAUI Blazor Hybrid and Web App"	636
30.4	Anwendungsbeispiele.....	641
31	Fallbeispiel "MiracleList"	642
31.1	Das MiracleList-Backend	642
31.1.1	Softwarearchitektur	645
31.1.2	Entitätsklassen/Geschäftsobjekte/Datentransferobjekte	648
31.1.3	Entity Framework Core-Kontextklasse	650
31.1.4	Lebensdauer der Kontextklasse in ASP.NET Core-Anwendungen.....	651
31.1.5	Geschäftslogik.....	651
31.1.6	Web API.....	658
31.2	Architektur der MiracleList-Frontend-Implementierungen mit Blazor	667
31.2.1	Kopfprojekte	670
31.2.2	Abstraktionsschicht für 2-Tier/3-Tier	670
31.2.3	Abstraktion der Authentication State Provider (MLAuthenticationStateProvider.cs).....	674
31.2.4	Globaler Anwendungszustand (IAppState/AppState)	676
31.2.5	Projektmappe	676
31.3	Gemeinsame Razor-Komponenten (MLBlazorRCL)	678
31.3.1	Projektdatei	680
31.3.2	Layoutvorlage (MainLayout.razor).....	681
31.3.3	Anmeldeansicht (Login.razor)	683
31.3.4	Serverstatusanzeige auf der Anmeldeseite (ServerState.razor)	687
31.3.5	Verbindungsstatusanzeige im Kopfbereich (ConnectionState.razor)	690
31.3.6	Hauptansicht mit Aufgabenkategorieliste und Aufgabenliste (Main.razor)	691
31.3.7	EntiDarstellung einer Aufgabe in der Aufgabenliste (TaskElement.razor).....	700
31.3.8	Bearbeitungsformular (TaskEdit.razor).....	703
31.3.9	Unteraufgaben (SubTaskList.razor)	709
31.3.10	Bearbeitung der Aufgaben in einem Datagrid (TaskGrid.razor)	710
31.3.11	Einbettung anwendungsspezifischer Komponenten	712
31.3.12	Dateien im Dateisystem verwalten (FilesFromFilesystem.razor).....	713
31.3.13	Dateien per Webservice verwalten (FilesFromWebservice.razor)	719
31.4	MiracleList-Frontend mit Blazor Server (MiracleList_BS).....	723
31.4.1	Softwarearchitektur	723
31.4.2	Einrichten der Blazor Server-Anwendung	724
31.4.3	Projektaufbau	724
31.4.4	Projektdatei	727
31.4.5	Startcode (Program.cs und Startup.cs)	728
31.4.6	Startseite (_Layout.cshtml)	731
31.4.7	Authentication State Provider (MLAuthenticationStateProvider2Tier)	732
31.5	MiracleList-Frontend mit Blazor WebAssembly (MiracleList_BW)	735

31.5.1 Softwarearchitektur	735
31.5.2 Einrichten der Blazor WebAssembly-Anwendung.....	736
31.5.3 Projektaufbau.....	737
31.5.4 Projektdatei.....	738
31.5.5 HTML-Rahmendatei mit Ladeanimation	739
31.5.6 Blazor WebAssembly-Startcode (Program.cs)	741
31.5.7 Authentication State Provider (MLAuthenticationStateProvider3Tier)	743
31.6 MiracleList-Frontend als Blazor Web App / Blazor United (MiracleList_BU)	746
31.6.1 Softwarearchitektur	746
31.6.2 Einrichten der Blazor United-Anwendung	747
31.6.3 Projektaufbau.....	748
31.6.4 Projektdateien.....	749
31.6.5 Blazor United-Startcode (Program.cs).....	750
31.6.6 App.razor.....	753
31.6.7 Authentication State Provider (MLAuthenticationStateProvider3Tier)	753
31.7 MiracleList-Hybrid-Frontend mit Blazor Desktop in einer WPF-Anwendung (MiracleList_BD)	753
31.7.1 Softwarearchitektur	754
31.7.2 Einrichten der Blazor Desktop-Anwendung.....	755
31.7.3 Projektaufbau.....	755
31.7.4 Projektdatei.....	756
31.7.5 WPF-Hostfenster.....	758
31.7.6 Klasse HybridSharedState	761
31.7.7 App.razor.....	762
31.7.8 Exportfunktionen (Export.razor)	763
31.8 MiracleList-Cross-Platform-Frontend mit Blazor MAUI (MiracleList_BM)	765
31.8.1 Softwarearchitektur	767
31.8.2 Projektaufbau.....	767
31.8.3 Einrichten der Blazor MAUI-Anwendung	768
31.8.4 Projektdatei.....	769
31.8.5 Startdatei MauiProgram.cs	771
31.8.6 MAUI-Hostfenster.....	772
31.8.7 Klasse HybridSharedState	774
31.8.8 App.razor.....	774
31.8.9 Export.razor.....	774
31.9 DevOps mit MiracleList	777
31.9.1 Automatisierte Tests	777
31.9.2 Build- und Release-Pipelines.....	779
32 Umstieg von ASP.NET Webforms zu Blazor	780
32.1 Migrationspfade und Herausforderungen.....	780
32.2 Grundsätzliche Vorgehensweise bei der Migration	782
32.3 Umstieg von Webforms auf Blazor an einem Beispiel	783
32.3.1 Umsetzung der Datentabelle mit Webforms und Blazor	784
32.3.2 Umstellung der Benutzerschnittstellenbeschreibung	788
32.3.3 Umstellung der Benutzerschnittstellensteuerung.....	789

32.4	Umsetzung der Flugdatentabelle mit Telerik UI for Blazor	789
32.5	Umsetzung der Flugdatentabelle mit DevExpress UI for Blazor.....	793
33	Razor Components außerhalb von Webanwendungen.....	799
33.1	Voraussetzungen.....	799
33.2	Rendering von Razor-Komponenten	799
34	WebAssembly ohne Blazor.....	801
34.1	.NET-DLLs in WebAssembly ohne Blazor	801
34.2	WASI mit .NET (WASI SDK)	803
35	Quellen im Internet	806
36	Versionsgeschichte dieses Buchs.....	807
37	Stichwortverzeichnis (Index)	808
38	Werbung in eigener Sache ☺.....	822
38.1	Dienstleistungen	822
38.2	Aktion "Buch für Buchrezension"	823
38.3	Angebot "PDF-Buch-Abo".....	824

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, Svelte und diversen anderen Bibliotheken und Frameworks). Seit dem Jahr 2016 verwende ich in vielen 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 "neue Sau", die durch das Webdorf getrieben wird; diese "Sau" namens Blazor.

Dieses Buch behandelt die Version **Blazor 9.0**, die im Rahmen von **.NET 9.0 und ASP.NET Core 9.0** läuft. Diese erste Ausgabe des Buchs ist kurz vor der RTM-Version erschienen auf Basis von Release Candidate 2 und wird kurz nach dem Release von .NET 9.0 in einer aktualisierten Version veröffentlicht.

Alle Ausführungen im Buch und den abgedruckten Beispielen gelten – sofern nicht ausdrücklich anders erwähnt – sowohl für **Blazor Web Apps (Blazor Static Server-Side-Rendering, Blazor Server, Blazor WebAssembly)** sowie **Blazor Hybrid (Blazor Desktop und Blazor Blazor MAUI)**. Es gibt nur wenige Unterschiede zwischen 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).

Dieses Buch richtet sich **sowohl an Neueinsteiger in Blazor 9.0 als auch Umsteiger von älteren Versionen**. Es wäre nicht wirtschaftlich, zwei getrennter Bücher dafür zu produzieren. Daher bitte ich die Leser, die neu einsteigen um Verständnis, dass es im Buch Hinweise die Änderungen zu früheren Versionen gibt.

Mit Blazor konnten wir schon einige Single-Page-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 in der Praxis mit Blazor noch nicht ohne etwas JavaScript auskommt. 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 richtet sich sowohl an Neueinsteiger in Blazor als auch Umsteiger von älteren Blazor-Versionen, denn es gibt immer Hinweise darauf, ab welcher Version einzelne Features verfügbar sind.

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 69,99 Euro (der Autor erhält 25,30 Euro):
www.amazon.de/exec/obidos/ASIN/3934279392/itvisions-21
- Kindle-E-Book bei Amazon.de für 59,99 Euro (der Autor erhält 19,62 Euro):
www.amazon.de/exec/obidos/ASIN/B0CLKZJKT/itvisions-21
- PDF inkl. Updates bei Leanpub.com ab 49,99 Dollar (der Autor erhält 39,99 Dollar):
www.leanpub.com/Blazor90
- Als Teil des E-Book-Buch-Abos zusammen mit anderen aktuellen Fachbüchern ab 99,00 Euro im Jahr inkl. aller Updates (der Autor erhält den kompletten Preis):
www.IT-Visions.de/BuchAbo

Tipp: Käufer bei Leanpub.com können jederzeit Aktualisierungen des PDF-Buchs (gleiche Hauptversion) kostenfrei dort beziehen. Käufer bei Amazon erhalten die PDF-Ausgabe einmalig kostenfrei (siehe Kapitel "Über dieses Fachbuch"). E-Book-Abonnenten haben jederzeit Zugriff auf alle aktuellsten Ausgaben der Fachbücher von Dr. Holger Schwichtenberg.

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 dieses Kontaktformular:

www.dotnet-doktor.de/Leserfeedback

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

Ich helfe Ihnen gerne, Ihren eigenen Programmcode zu schreiben bzw. Fehler in Ihrem Code zu finden, aber ich hoffe, Sie verstehen, dass ich dies nicht ehrenamtlich tun kann. Wenn Sie **technische Hilfe** zu Blazor, .NET und JavaScript 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 Firma www.IT-Visions.de (Beratung, Schulung, Support, Softwareentwicklung) 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 meine 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 Losungswort abgefragt. Bitte geben Sie dort **LostInSpace** 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 in der IT tätig als Softwareentwickler, Softwarearchitekt, Berater, Dozent und Fachjournalist
- Fachlicher Leiter des Expertenteams bei www.IT-Visions.de in Essen
- Über 95 Fachbücher bei verschiedenen Verlagen, u.a. Carl Hanser Verlag, O'Reilly, APress, Microsoft Press und Addison Wesley sowie im Selbstverlag
- Mehr als 1500 Beiträge in Fachzeitschriften und Online-Portalen
- Gutachter in den Wettbewerbsverfahren der EU vs. 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. BASTA!, Developer Week, .NET Developer Conference, MD DevDays, Microsoft TechEd, Microsoft Summit, Microsoft IT Forum, OOP, .NET Architecture Camp, IT Tage, enterJS, Advanced Developers Conference, DOTNET Cologne, iterate=>ruhr, Community in Motion, DOTNET-Konferenz, VS One, NRW.Conf, Windows Forum, Container Conf)
- Auszeichnungen und Zertifikate von Microsoft:
 - Microsoft Most Valuable Professional (MVP), ununterbrochen ausgezeichnet seit 2004
 - Microsoft Certified Solution Developer (MCSD)
- Thematische Schwerpunkte:
 - Softwarearchitektur, mehrschichtige Softwareentwicklung, Softwarekomponenten
 - Visual Studio, Continous Integration (CI) und Continous Delivery (CD) mit Azure DevOps
 - Microsoft .NET (.NET Framework, .NET Core, modernes .NET), C#, Visual Basic
 - .NET-Architektur, Auswahl von .NET-Techniken
 - Einführung von .NET, Migration auf .NET
 - Webanwendungsentwicklung und Cross-Plattform-Anwendungen mit HTML/CSS, JavaScript/TypeScript und C# sowie Webframeworks wie Angular, Vue.js, Svelte, ASP.NET (Core) und Blazor
 - Verteilte Systeme/Webservices mit .NET, insbesondere WebAPI, gRPC und WCF/CoreWCF
 - 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) und .NET Foundation
 - Betrieb diverser Community-Websites:
www.dotnet-lexikon.de
www.dotnetframework.de
www.windows-scripting.de
www.aspnetdev.de
u.a.
- Firmenwebsite: www.IT-Visions.de
- Weblog: www.dotnet-doktor.de
- Kontakt für Anfragen zu Schulung und Beratung sowie Softwareentwicklungsprojekten:
E-Mail kundenteam@IT-Visions.de
Telefon 0201 / 64 95 90 – 50
- Kontakt für Feedback zu diesem Buch:
www.dotnet-doktor.de/Leserfeedback

5 Über dieses Fachbuch

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 Hinweis zu den Vertriebswegen

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 69,99 Euro (der Autor erhält 25,30 Euro):
www.amazon.de/exec/obidos/ASIN/3934279392/itvisions-21
- Kindle-E-Book bei Amazon.de für 59,99 Euro (der Autor erhält 19,62 Euro):
www.amazon.de/exec/obidos/ASIN/B0CLKZJKTX/itvisions-21
- PDF-E-Book **inkl. aller Buch-Updates** bei Leanpub.com ab 49,99 Dollar (der Autor erhält 39,99 Dollar):
www.leanpub.com/Blazor90
- Als Teil des E-Book-Buch-Abos zusammen mit anderen aktuellen Fachbüchern ab 99,00 Euro im Jahr **inkl. aller Buch-Updates** (der Autor erhält den kompletten Preis):
www.IT-Visions.de/BuchAbo

Hinweise: 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.

Ich nenne dabei auch den Erlös für den Autor, weil ich sehr häufig Leser treffe, die fälschlicherweise denken, der wesentliche Teil des Buchpreises komme dem Autor zu Gute. Das ist leider nicht so, außer bei Leanpub.com oder eigenen Vertriebswegen wie meinem Buchabo. Daher denke ich, dass es sinnvoll ist, dies transparent zu machen.

Tipp: Käufer bei Leanpub.com können jederzeit Aktualisierungen des PDF-Buchs (gleiche Hauptversion) kostenfrei dort beziehen. Käufer bei Amazon erhalten die PDF-Ausgabe einmalig kostenfrei (siehe Kapitel "Bezugsquelle des PDF-E-Books"). E-Book-Abonnenten haben jederzeit Zugriff auf alle aktuellsten Ausgaben der Fachbücher von Dr. Holger Schwichtenberg

5.3 Bezugsquelle des PDF-E-Books für Amazon-Kunden

Wenn Sie dieses Buch in gedruckter Form oder als Kindle-Ausgabe bei Amazon erworben haben, können Sie zusätzlich eine PDF-Version des Buchs **kostenfrei** erhalten.

Leiten Sie dazu Ihren Kaufbeleg von Amazon an folgende E-Mail-Adresse weiter:

PDFBuchZugabe@dotnet-doktor.de

Geben Sie dabei bitte Vorname, Name, Firma und E-Mail-Adresse an.

Sie erhalten dann binnen 1-2 Wochen das auf Sie personalisierte PDF-Dokument. Dieses Angebot gilt innerhalb von 6 Monaten nach dem Kauf des Buchs bei Amazon.

5.4 Bezugsquelle für Aktualisierungen

Leanpub-Kunden können jederzeit Aktualisierungen des PDF-Buchs (gleiche Hauptversion!) kostenfrei bei [Leanpub.com](https://leanpub.com) beziehen.

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

<https://leanpub.com/Blazor90/c/LostInSpace>

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 können.

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

5.5 Hinweise zur Breite und Tiefe dieses Buchs – Sie haben Einfluss!

Ein Fachbuch, das ein riesengroßes Themengebiet wie Blazor behandelt, kann nicht jedes Teilgebiet und jeden Aspekt der Technik behandeln, zumindest nicht in gleicher Tiefe. Dann würde solch ein Fachbuch über eintausend Seiten, in einigen Fällen sogar mehrere tausend Seiten umfassen.

Ich denke, dass ich nach aktuellem Stand der Technik und meinem Wissenstand etwa 2.500 Buchseiten zu Blazor schreiben könnte. Würden Sie so ein dickes (und entsprechend teures) Buch kaufen und lesen wollen?

Wie jeder Fachautor lese auch ich immer wieder Kritik, dass ein Leser ein bestimmtes Thema nicht oder nicht in ausreichender Tiefe behandelt sei in dem Buch. Das ist aus der Sicht des einzelnen Lesers sicherlich gerechtfertigt, aber wie jeder Fachautor muss ich eben zwingend eine Auswahl der Themen treffen. Gerne dokumentiere ich hier, wie ich persönlich diese Auswahl für meine Bücher treffe:

- Ich behandle im Buch die Themen, die wir in unserer Firma selbst in der Praxis in den Softwareentwicklungsprojekten brauchen.
- Ich behandle zusätzlich die Themen, die unsere Kunden in Schulungen behandelt haben möchten oder in Beratungsgesprächen ansprechen.
- Ich behandle Themen, die ich spannend finde.

Folglich sind die Themen, die ich im Buch nicht oder nur kurz behandle für uns und unsere Kunden nicht relevant bzw. so selbsterklärend, dass es keine Fragen dazu gibt.

Natürlich kann das für Sie anders sein. Sie können mir immer gerne schreiben, wenn Sie ein Thema im Buch behandelt haben möchten. Ich sammele diese Anregungen und wenn es mehrere Zuschriften zu einem Thema gibt, dann kommt das Thema weit oben auf die Prioritätenliste. Ich denke, das ist ein faires Verfahren.

5.6 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.

- Autorisierung mit Rollen
- Anpassbarer Ladevorgang bei Blazor WebAssembly (seit Blazor 6.0)
- Custom Event Arguments (seit Blazor 6.0)
- DotNet.createJSObjectReference()

- Eigene Basisklassen für Razor Components mit @inherits
- Datei-Upload bei Blazor WebAssembly und Blazor MAUI via Webservice (ist bereits im Code von MiracleList_BW enthalten, aber im Buch noch nicht beschrieben)
- IComponentActivator zur Kontrolle der Komponenteninstanziierung (seit Blazor 5.0)
- Installation von Blazor-Anwendungen auf Linux-Servern
- Integration von Blazor in JavaScript-Anwendungen (seit Blazor 6.0)
- Inkrementelle Migration auf Blazor mit .NET Upgrade Assistant und Microsoft.AspNetCore.SystemWebAdapters
- IL-Linker-Konfigurationsdatei (LinkerConfig.xml)
- JavaScript-Interop mit IJSInProcessRuntime als Alternative zu der im Buch behandelten IJSRuntime (in Blazor WebAssembly)
- JavaScript beim Static Server-Side-Rendering (SSR) mit <PageScript> (seit Blazor 8.0)
- Blazorise Components (kostenfrei) → <https://blazorise.com>
- Synchrone JavaScript-Interop mit IJSInProcessRuntime (in Blazor WebAssembly seit Blazor 5.0)
- LESS und SCSS mit Blazor
- NavigateToLogin() und NavigateToLogout() in NavigationManagerExtensions (seit Version 7.0)
- Open ID Connect-Authentifizierung: flexiblere Open ID Connect-Authentifizierung in Blazor WebAssembly (seit Version 7.0)
- PreserveWhiteSpace
- QR-Codes darstellen
- Razor-Rendering für Versand von HTML-E-Mail
- Routing mit @attribute [Route(Constants.CounterRoute)]
- Routing-Anpassungen (<https://chrissainty.com/building-a-custom-router-for-blazor/amp/>)
- SQLite in Blazor WebAssembly (möglich seit Blazor 6.0)
- SQLite in Blazor Hybrid
- Telerik REPL for Blazor <https://blazorrepl.telerik.com>
- Testen von Blazor-Anwendungen mit Razor-Komponententests über bUnit alias Razor.Components.Testing.Library → **Beispiele zu bUnit finden Sie bereits im MiracleList-Beispiel, das Sie herunterladen können!**
- Testen von Blazor-Anwendungen mit End-To-End-GUI-Tests im Webbrower über Microsoft Playwright → **Beispiele zu Playwright finden Sie bereits im MiracleList-Beispiel, das Sie herunterladen können!**
- TypeScript in Blazor Apps
- Validierung: Anpassung der CSS-Klassen (seit Blazor 5.0)
- Visual Studio Code als alternatives Werkzeug zu Visual Studio
- Web Crypto API / SubtleCrypto: Hashing-Algorithmen (seit .NET 7.0)
- Web Components Custom Elements mit RegisterAsCustomElement() (seit Blazor 6.0 im Preview, seit Blazor 7.0 einsatzreif)

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.

5.7 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/fbsolero/Bolero>], die außerhalb von Microsoft entwickelt wird und daher kein offizielles Produkt von Microsoft ist.

5.8 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 3. Auflage</p> <p>Holger Schwichtenberg und Jörg Krause mit Beiträgen von Dr. Joachim Fuchs, Manfred Steyer und Sebastian Kierschmager</p>	<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# Crashkurs" empfehlen:

	<p>C# 13.0 Crashkurs</p> <ul style="list-style-type: none"> PDF-E-Book bei Leanpub.com ab 19,99 Dollar gedruckt (Print-on-Demand) bei Amazon.de für 39,99 Euro Kindle-E-Book bei Amazon.de für 29,99 Euro www.IT-Visions.de/CSharpBuch
---	---

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

	<p>Moderne Datenzugriffslösungen mit Entity Framework Core 9.0</p> <ul style="list-style-type: none"> Gedruckt bei Amazon.de für 79,99 Euro Kindle-E-Book bei Amazon.de für 69,99 Euro PDF-E-Book bei Leanpub.com für 49,99 Dollar www.IT-Visions.de/EFCBuch
--	--

5.9 Vermeidung des Einsatzes kostenpflichtiger Software

Der Autor dieses Buch weiß, dass viele, aber nicht alle Leser dieses Buchs professionelle Softwareentwickler sind und hat daher darauf geachtet, dass Sie keine Software kaufen müssen, um die Inhalte dieses Buchs nachzuvollziehen.

Blazor selbst als Teil von .NET kostenfrei. Bei Visual Studio können Sie zum Lernen die kostenfreie Community-Edition einsetzen. Für den kommerziellen Einsatz brauchen Sie in Unternehmen, die die von Microsoft jeweils definierten Umsatzgrenze überschreiten, eine kostenpflichtige Version.

Darüberhinaus verzichtet das Buch im Wesentlichen auf die Beschreibung kostenpflichtiger Produkte. Eine Ausnahme bildet das Kapitel "Komponentenbibliotheken von Drittanbietern". **Das MiracleList-Fallbeispiel in diesem Buch ist komplett ohne kommerzielle Komponenten implementiert.**

5.10 Hinweis zur Erwähnung der Blazor-Varianten in diesem Buch

Vor .NET 6.0 gab es nur zwei Blazor-Arten: Blazor Server und Blazor WebAssembly. Mit .NET 6.0 erschien eine dritte Blazor-Art: Blazor Hybrid, die man auch noch mal differenzieren kann in hybride Apps in WPF und Windows Forms (Blazor Desktop) und in .NET MAUI (Blazor MAUI). Seit Blazor 8.0 gibt es auch noch Blazor Static Server-Side-Rendering (SSR) und Blazor United, wobei letzteres kein offizieller Begriff ist, sondern von Microsoft nur zu Beginn von .NET 8.0 verwendet wurde, um die Integration und den nahtlosen Übergang von Blazor Server und Blazor WebAssembly zu beschreiben.

Es wird im Buch vorerst noch einige Stellen geben, wo ich nicht alle Blazor-Arten explizit erwähne. Es ist mir als Hobby-Autor aus zeitlichen Gründen vorerst nicht möglich, das ganze Buch nach diesen Stellen zu durchkämmen. Was per Volltextsuche auffindbar war, habe ich erledigt.

Bitte: Geben Sie mir gerne einen freundlichen Hinweis auf Stellen, wo eine Erwähnung von Blazor MAUI, Blazor Desktop, Blazor Static SSR oder Blazor United fehlt, siehe nächstes Unterkapitel.

5.11 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 5 Euro
Sprachlicher Fehler	Pro Fehler 2 Euro

Ein Beispiel: Wenn Sie zwei inhaltliche Fehler und zehn Rechtschreibfehler in diesem Buch finden, dann haben Sie bei mir 30 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!

Die Fehlermeldung zählt nur, wenn nicht ein anderer Leser dies bereits gemeldet hat und es daher in der aktuellen Auflage schon korrigiert ist.

Ich freue mich auf Ihre Fehlermeldung!

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 **LostInSpace** 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

Das Downloadpaket zu diesem Buch besteht aus mehreren Ordnern (siehe Abbildung).

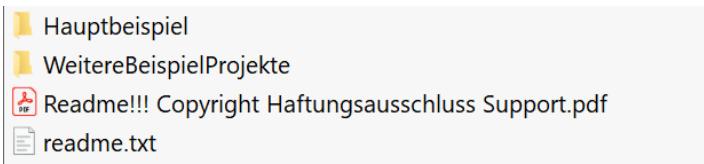


Abbildung: Inhalt des Download-Pakets zu diesem Buch

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

Es war eine bewusste Entscheidung, in diesem Buch nicht für jedes einzelne Kapitel ein eigenes Blazor-Projekt zu schaffen, sondern die meisten Beispiele in ein echtes Fallbeispiel "MiracleList" 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 unter "Hauptbeispiel/"	Programmcode aus Kapitel(n)	Hinweise
/MiracleList_Bazor /src/MiracleList_BU	Alle	Blazor United-Beispiele einschließlich der Blazor United-Implementierung des Fallbeispiels "MiracleList" auf Basis der Vorlage "Blazor Web App". Die Anwendung läuft im Auto-Modus, wechselt also von Blazor Server zu Blazor WebAssembly.
/MiracleList_Bazor /src/MiracleList_BS	Alle	Blazor Server-Beispiele einschließlich der Blazor Server-Implementierung des Fallbeispiels "MiracleList". Dieses Projekt

Ordner unter "Hauptbeispiel/"	Programmcode aus Kapitel(n)	Hinweise
		basiert noch auf der alten Projektvorlage "Blazor Server", läuft aber unter der aktuellen Blazor-Version.
/MiracleList_Bazor /src/MiracleList_BW	Alle	Blazor WebAssembly-Beispiele einschließlich der Blazor WebAssembly-Implementierung des Fallbeispiels "MiracleList". Dieses Projekt basiert noch auf der alten Projektvorlage "Blazor WebAssembly", läuft aber unter der aktuellen Blazor-Version..
/MiracleList_Bazor /src/MiracleList_BD	Alle	Blazor WPF-Desktop-Beispiele einschließlich der Blazor Desktop-Implementierung des Fallbeispiels "MiracleList"
/MiracleList_Bazor /src/MiracleList_BM	Alle	Blazor MAUI-Beispiele einschließlich der Blazor MAUI-Implementierung des Fallbeispiels "MiracleList"
/MiracleList_Bazor /src/MiracleListAPI_Proxy	Fallbeispiel	Generierte Proxy-Klasse für MiracleList-Backend-WebAPI, wird verwendet in MiracleList_BW und MiracleList_BM
/MiracleList_Bazor /src/ITVisions.Blazor	JavaScript-Interoperabilität Razor Class Library Fallbeispiel	Razor Class Library mit eigenen Blazor-Steuerelementen und Hilfscode für JavaScript-Interoperabilität; wird in allen vier Implementierung von MiracleList verwendet. Realisiert u.a. die vielfach in diesem Buch verwendete Hilfsklasse BlazorUtil.cs mit zugehöriger JavaScript-Datei BlazorUtil.js
/MiracleList_Bazor /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 in allen vier Implementierung von MiracleList verwendet.
/MiracleList_Bazor /src/SamplesRCL	Alle	Ergänzende Beispiele aus dem Buch, die nicht in das MiracleList-Szenario fallen. Dieses Projekt wird in allen vier Implementierung von MiracleList verwendet. Diese Beispiele sind im Menü "Blazor-Beispiele außerhalb der MiracleList" aufrufbar.
/MiracleList_Bazor /src/MiracleList_Backend	Fallbeispiel	Backend (ASP.NET Core WebAPI und ASP.NET Razor Pages)
/MiracleList_Bazor /Test/UnitTests	Bisher nicht im Buch behandelt	Automatisierte Tests (Unit Tests) der MiracleList-Geschäftslogik (BL), die wahlweise als isolierte Tests oder als

Ordner unter "Hauptbeispiel/"	Programmcode aus Kapitel(n)	Hinweise
		Integrationstests gegen die Datenbank laufen können
/MiracleList_Bazor /Test/WebserviceTests	Bisher nicht im Buch behandelt	Automatisierte Tests des MiracleList-Backend-Services (WebAPI und SignalR)
/MiracleList_Bazor /Test/BlazorTests	Bisher nicht im Buch behandelt	Automatisierte Tests der Blazor-Komponenten in MiracleList mit bUnit
/MiracleList_Bazor /Test/ SeleniumTests	Bisher nicht im Buch behandelt	Automatisierte End-to-End-GUI-Tests der MiracleList-Anwendung innerhalb von Webbrowern mit Selenium
/MiracleList_Bazor /Test/ PlaywrightTests	Bisher nicht im Buch behandelt	Automatisierte End-to-End-GUI-Tests der MiracleList-Anwendung innerhalb von Webbrowern mit Microsoft Playwright

Ordner unter "WeitereBeispiel Projekte"	Programmcode aus Kapitel(n)	Hinweise
/NET9_Demos	Allen Kapiteln, in denen es Neuerungen in Blazor 9.0 gibt	Beispiele zu neuen Funktionen in Blazor 9.0.
/NET8_Demos	Static Server-Side-Rendering und Tipps & Tricks/QuickGrid	Beispiel zu Blazor Static SSR und dem QuickGrid-Steuerelement, die inhaltlich nicht in das MiracleList-Beispiel passen.
/BlazorWebservices	Zugriff auf Webservices/WebAPIs	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.
/WebComponentsMitAngular_ITVElements	Interoperabilität zwischen .NET und JavaScript	TypeScript-Quellcode für die dort verwendeten Angular-basierten Komponenten

Ordner unter "WeitereBeispiel Projekte"	Programmcode aus Kapitel(n)	Hinweise
/WebAssemblyPrerendering	Tipps & Tricks/Render-Modi bei Blazor WebAssembly	Beispiel für das Prerendering einer Blazor WebAssembly-Anwendung

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/fluentui/>]
- Tailwind CSS [<https://tailwindcss.com/>]
- Ant Design [<https://ant.design/>]
- Bulma [<https://bulma.io/>]
- eFrolicss [<https://efrolic.github.io/css/>]
- Lion [<https://github.com/ing-bank/lion>]

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 der 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].



Abbildung: Das MiracleList-Logo

Hinweis: Mittlerweile hat Microsoft Wunderlist sterben lassen zugunsten einer Neuimplementierung unter dem Namen "Microsoft To Do" [<https://todo.microsoft.com/>]. Dieses Buch behandelt – als Erinnerung an das verblichene Wunderlist – eine Aufgabenverwaltung "MiracleList" mit ähnlicher Navigation.

Der angemeldete Benutzer in MiracleList 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, einem Fälligkeitsdatum und kann als erledigt markiert werden. Über die Funktionen von Wunderlist hinaus kann 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, die das Original noch zusätzlich beherrscht (z.B. die Suche nach Hashtags, das Duplizieren und Drucken von Listen sowie der Aufgabenaustausch zwischen Benutzern), sind in MiracleList noch nicht berücksichtigt. Wir können ja in einem kleinen Buch nicht eine 100-Millionen-Dollar-App nachprogrammieren – zumindest nicht komplett 😊.

Einige Funktionen wie anklickbare Hyperlinks in Aufgabentexten sind nicht realisiert, um einen Missbrauch der für alle Nutzer offenen Website zu vermeiden.

6.4.2 MiracleList-Client-Features

Nicht alle MiracleList-Funktionen sind in bereits in allen Clients implementiert. Grund dafür ist hauptsächlich Zeitmangel bei mir. **Bitte beachten Sie, dass MiracleList nur zu Test- und Weiterbildungszwecken (Schulungen, Fachbücher) dient!**

Hinweis: In der folgenden Tabelle bedeuten die zusammengelegten Zellen, dass die vier in diesem Buch vorgestellten Blazor-Varianten bei diesen Punkten eine gemeinsame Codebasis besitzen. Wie dies realisiert ist, erfahren Sie im Kapitel "Fallbeispiel MiracleList".

✓ vorhanden ⚡ noch nicht vorhanden ✕ nicht möglich wg. Sandbox bzw. Plattformbeschränkungen

Client	Angular	Angular	Angular	Vue.js	Blazor Server	Blazor Web-Assembly und Blazor United	Blazor Desktop	Blazor MAUI	Blazor Web-Assembly Tutorial
Umgebung	Browser	Electron	Cordova	Browser	Browser	Browser	WPF	MAUI	Browser
Codequelle	GitHub	GitHub	GitHub	GitHub	Buch	Buch	Buch	Buch	GitHub
Anmeldung	✓	✓	✓	✓			✓		✓
Anmeldung an verschiedenen Backends	⚡	⚡	⚡	⚡			✓		⚡
Statusanzeige für Backendsysteme auf der Anmeldeseite	⚡	⚡	⚡	⚡			✓		⚡
Permanenter Backend-Status in Hauptansicht	✓	✓	✓	⚡			✓		⚡
Permanenter Signal-Hub-Status in Hauptansicht	⚡	⚡	⚡	✓			✓		⚡
Aufgabenkategorien	✓	✓	✓	✓			✓		✓
Aufgaben	✓	✓	✓	✓			✓		✓
Unteraufgaben	✓	✓	✓	✓			✓		⚡
Aufgaben-Detailansicht ohne Bearbeitung	✓	✓	✓	⚡			⚡		⚡
Aufgaben bearbeiten mit Validierung	✓	✓	✓	✓			✓		✓
Aufgaben löschen	✓	✓	✓	✓			✓		✓
Aufgaben zuordnen per Drag&Drop	⚡	⚡	⚡	✓			✓		⚡
Aufgaben sortieren per Drag&Drop	✓	✓	✓	✓			⚡		⚡
Datei-Upload zu Aufgaben	⚡	⚡	⚡	⚡			✓		✓
Kontextmenü für Aufgaben	✓	✓	✓	⚡			✓		⚡
Ausblenden erledigter Aufgaben	⚡	⚡	⚡	⚡			✓		⚡
Liste der fälligen Aufgaben	✓	✓	✓	⚡			⚡		⚡
Aufgaben suchen	✓	✓	✓	⚡			✓		⚡
Aufgabenexport direkt ins Dateisystem	✗	✓	⚡	✗	✗	✗	✓	✓	✗
Generierung eines Word-Dokuments	✗	⚡	✗	✗	✗	✗	✓	✓ (nur Windows)	✗
Fenster-Synchronisation mit Push-Nachrichten via SignalR	⚡	⚡	⚡	⚡			✓		✓
Toast-Benachrichtigungen	⚡	⚡	⚡	✓			✓		✓
Ansicht und Bearbeitung der Aufgaben in einem DataGrid	⚡	⚡	⚡	⚡			✓		⚡
Mehrsprachigkeit	⚡	⚡	⚡	⚡			⚡		⚡
Anmeldung persistiert im Browserspeicher	⚡	⚡	⚡	✓			✓		✓
Aktuelle Kategorie persistiert im Browserspeicher	⚡	⚡	⚡	⚡			✓		✓
Progressive Web App	⚡	✗	✗	✗	✗	✓	✓	✗	⚡

Abbildung: MiracleList-Client-Features

6.4.3 Technische Basis für MiracleList

Das MiracleList-Backend ist mit ASP.NET Core auf Basis von .NET und C# realisiert.

Es gibt derzeit insgesamt sieben Implementierungen des Frontends:

- Vue.js mit TypeScript
- Angular mit TypeScript
- Blazor WebAssembly mit .NET/C#
- Blazor Server mit .NET/C#

- Blazor Web App mit Auto-Modus (Blazor United)
- Blazor Desktop mit .NET/C#
- Blazor MAUI mit .NET/C#

In diesem Buch werden nur die Blazor-Varianten besprochen.

6.4.4 Webadressen

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

- Backend mit ASP.NET Core für Blazor WebAssembly- und Angular-Clients:
<https://miraclelistbackend.azurewebsites.net>
- Webanwendung "MiracleList"-Frontend mit Blazor Server und C#:
<https://miraclelist-bs.azurewebsites.net>
- Webanwendung "MiracleList"-Frontend mit Blazor WebAssembly und C#:
<https://miraclelist-bw.azurewebsites.net>
- Webanwendung "MiracleList"-Frontend mit Blazor Web App (Blazor United) und C#:
<https://miraclelist-bu.azurewebsites.net>
- Webanwendung "MiracleList"-Frontend mit Vue.js und TypeScript:
<https://miraclelist-vue.azurewebsites.net>
- Frontend mit Angular und TypeScript (Webanwendung und Cross-Platform-Desktop-Clients für Windows, Linux und macOS):
<https://miraclelist.azurewebsites.net>

6.4.5 Projektmappenaufbau

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

- **Backend:** Alle Projekte für das MiracleList-Backend
- **Blazor Apps Head Projects:** Die verschiedenen Implementierungen des Frontends mit verschiedenen Blazor-Varianten
- **Blazor Apps Shared Code:** Gemeiner Code der Head-Projekte
- **Solution Items:** Konfigurationsdateien für Editor, Paketmanager und Build (.editorconfig, .gitignore, nuget.config, Directory.Build.props u.a.), Hilfspakete sowie Skripte für DevOps-Prozesse
- **Tests:** Unit Tests, Integrationstests und Browser-UI-Tests
- **Tools:** Hilfsprojekte für das Anlegen der Datenbank im DevOps-Prozess und ein Diagramm der Geschäftsobjektklassen

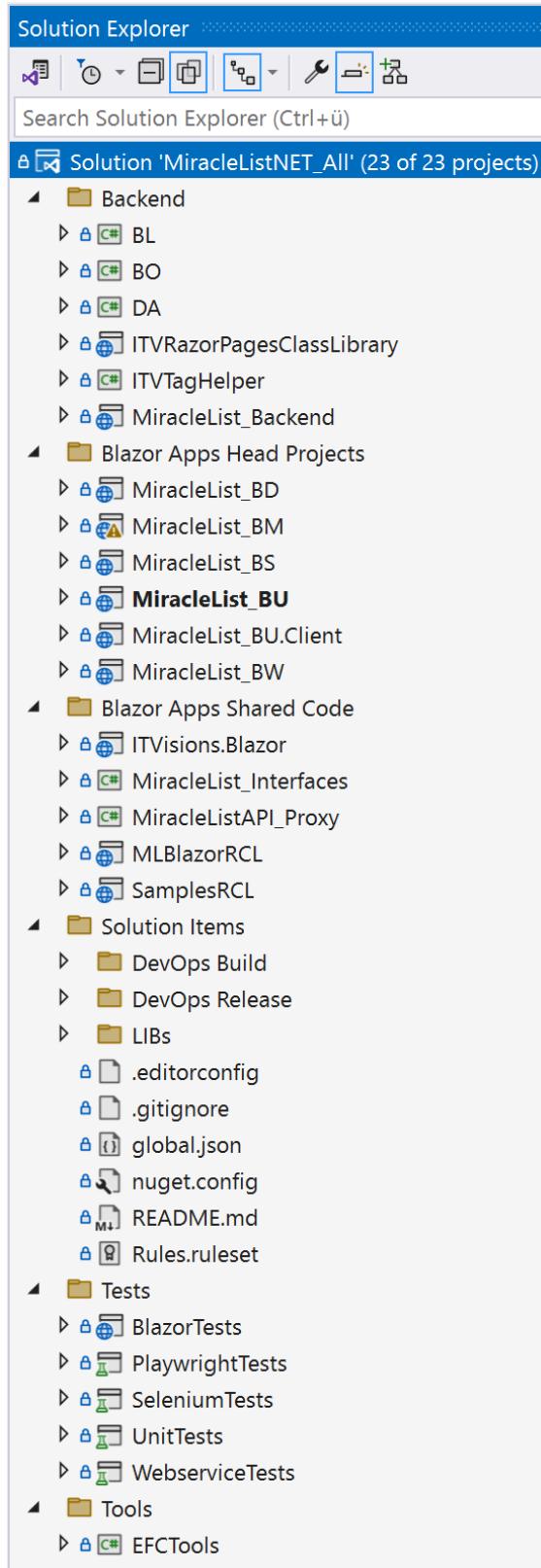


Abbildung: Aufbau der MiracleList-Projektmappe

6.4.6 MiracleList-Bildschirmmasken

Dieses Kapitel erläutern die wesentlichen Funktionen der MiracleList-Anwendung aus Benutzersicht, gibt Ihnen aber auch technische Hinweise.

Die Anmeldeseite fragt nach 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 mit einigen Beispielaufgaben angelegt.** Merken Sie sich das Kennwort für zukünftige Anmeldungen. Das Kennwort kann nicht geändert werden. Das Benutzeranmeldeverfahren ist bewusst vereinfacht, um die Hürde zum Einstieg in die Anwendung gering zu halten.

Die Blazor Server-Implementierung bietet bei "Server" Datenbankserver an. Diese können Sie in der appsettings.json-Datei unter "ConnectionString" konfigurieren.

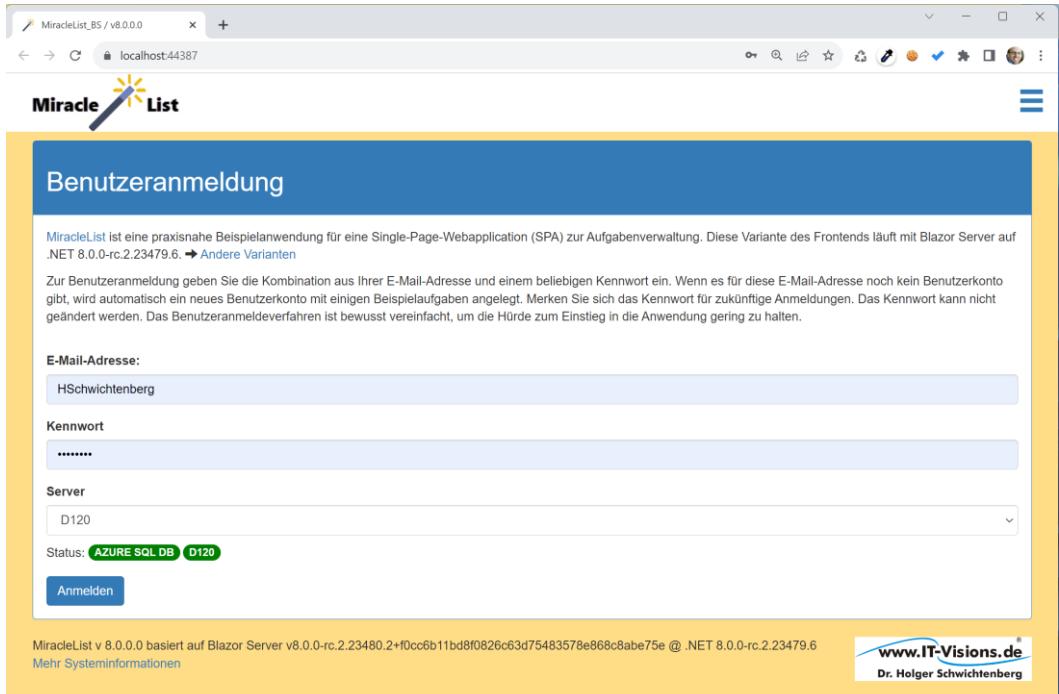
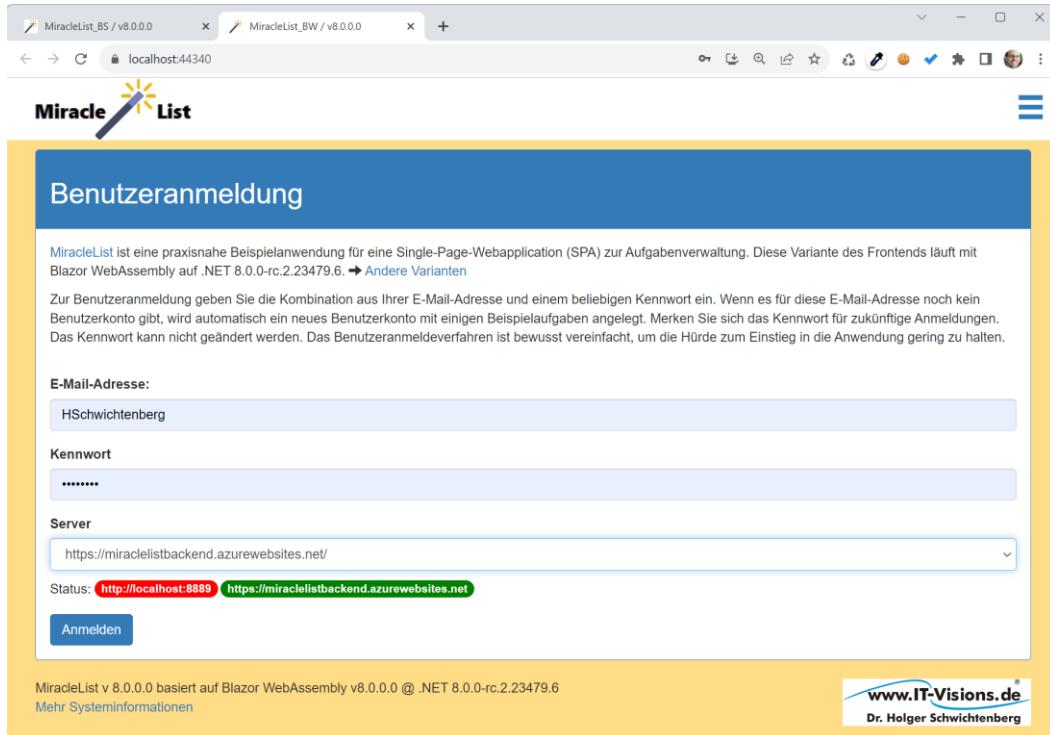


Abbildung: Anmeldeseite bei der Blazor Server-Implementierung des MiracleList-Frontends mit Auswahl zwischen verschiedenen Datenbankservern

Die Blazor WebAssembly-Implementierung bietet bei "Server" statt der Datenbankserver Application Server an.

Wenn die MiracleList-Anwendung in Visual Studio gestartet wird, wird als Server/Backend zusätzlich "localhost" zur Auswahl gestellt, um sich mit einer lokalen Version des WebAPI-Backends bzw. einer lokalen Version der Datenbank, statt dem in Azure gehosteten Varianten zu verbinden. Unter der Backend-Server-Auswahl sieht man eine Zeile mit Zustandskennzeichen ("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.

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 `/wwwroot/AppSettings.json` den Port ändern!



MiracleList ist eine praxisnahe Beispieldatenanwendung für eine Single-Page-Webapplication (SPA) zur Aufgabenverwaltung. Diese Variante des Frontends läuft mit Blazor WebAssembly auf .NET 8.0.0-rc.2.23479.6. ➔ [Andere Varianten](#)

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 mit einigen Beispielaufgaben angelegt. Merken Sie sich das Kennwort für zukünftige Anmeldungen. Das Kennwort kann nicht geändert werden. Das Benutzeranmeldeverfahren ist bewusst vereinfacht, um die Hürde zum Einstieg in die Anwendung gering zu halten.

E-Mail-Adresse:

HSchwichtenberg

Kennwort

.....

Server

https://miraclelistbackend.azurewebsites.net/

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

Anmelden

MiracleList v 8.0.0.0 basiert auf Blazor WebAssembly v8.0.0.0 @ .NET 8.0.0-rc.2.23479.6

Mehr Systeminformationen

www.IT-Visions.de
Dr. Holger Schwichtenberg

Abbildung: Anmeldeseite bei der Blazor WebAssembly-Implementierung des MiracleList-Frontends. Beim Start der Blazor WebAssembly-Anwendung auf "localhost" wird auch das lokale Backend angeboten, das aber hier nicht gestartet ist (daher rot!)

Wenn Sie die Blazor United-Implementierung starten, werden Sie diesen Effekt feststellen:

- Die Startseite zeigt beim allerersten Aufruf in der Fußzeile an, dass Sie mit Blazor Server läuft.
- Dennoch werden dort keine Datenbanken zur Auswahl angeboten, sondern Application Server, denn eine 2-Tier-Datenbankzugriff würde nach dem Wechsel zu Blazor WebAssembly nicht mehr funktionieren.
- Wenn Sie die Webanwendung aber dann ein zweites Mal aufrufen, steht in der Fußzeile nur noch ganz kurz "Blazor Server". Dann wechselt die Anzeige auf "Blazor WebAssembly" (zudem sehen Sie in der Titelleiste den Zusatz ".Client"), weil in der Zwischenzeit alle für Blazor WebAssembly notwendigen Dateien nachgeladen wurden.

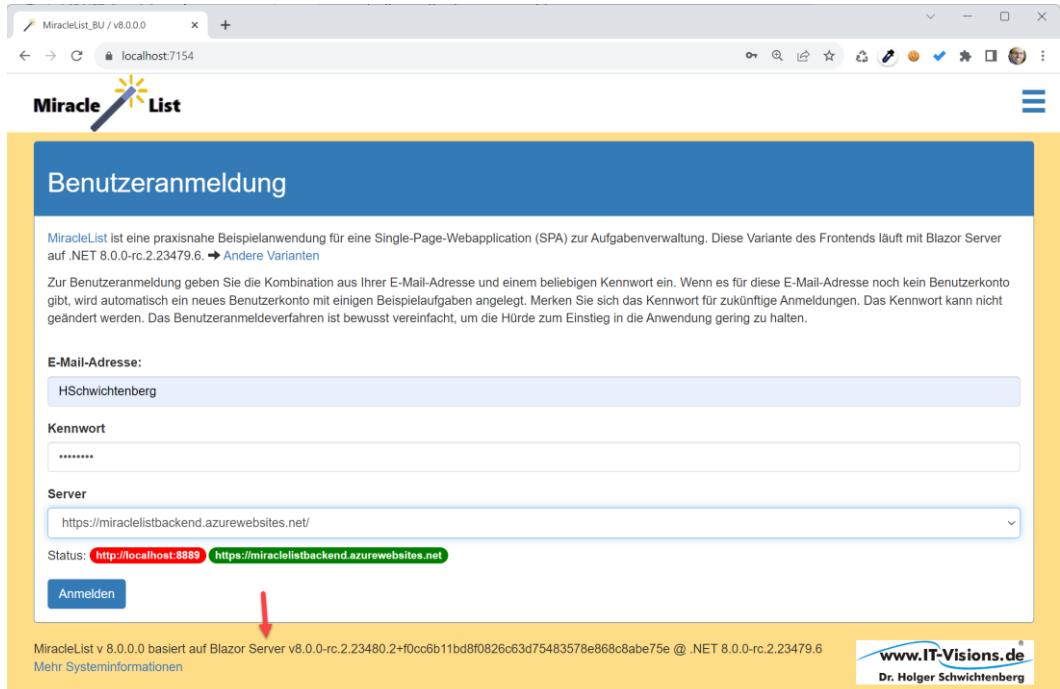


Abbildung: Anmeldeseite bei der Blazor United-Implementierung des MiracleList-Frontends beim ersten Aufruf

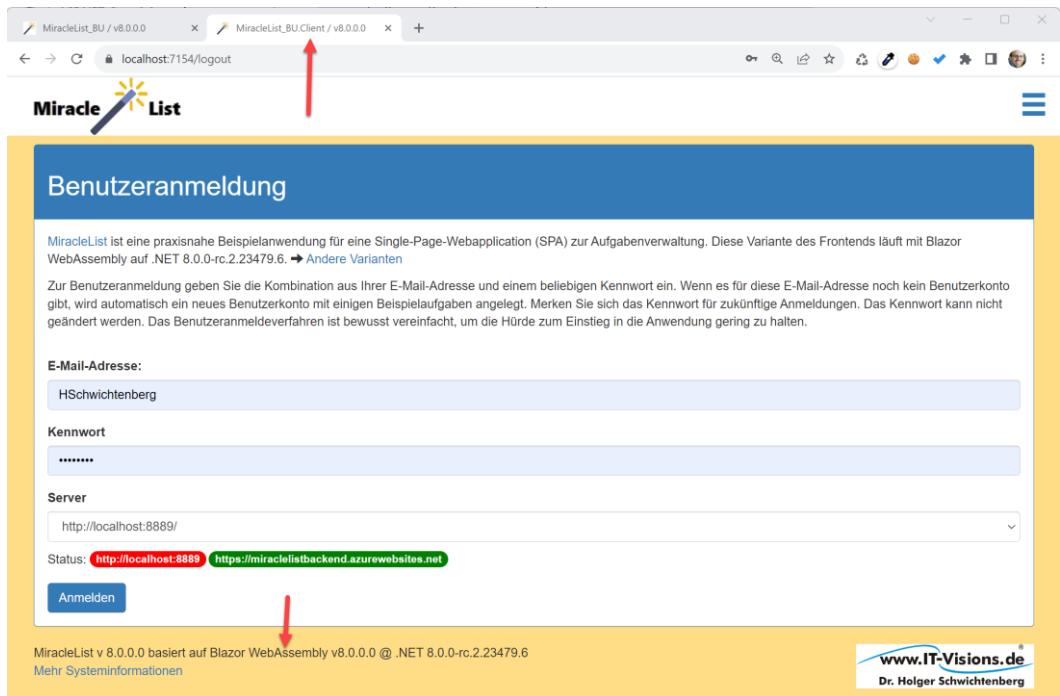


Abbildung: Anmeldeseite bei der Blazor United-Implementierung des MiracleList-Frontends beim zweiten Aufruf

Bei Blazor Desktop stehen ebenfalls Datenbankserver zur Auswahl, denn hier sind keine Webservices/Application Server als Zwischenschicht notwendig. Die Blazor MAUI-Implementierung

verwendet hingegen wieder das 3-Tier-Modell, denn nicht alle Plattformen können direkt Datenbankmanagementsystem aufrufen.

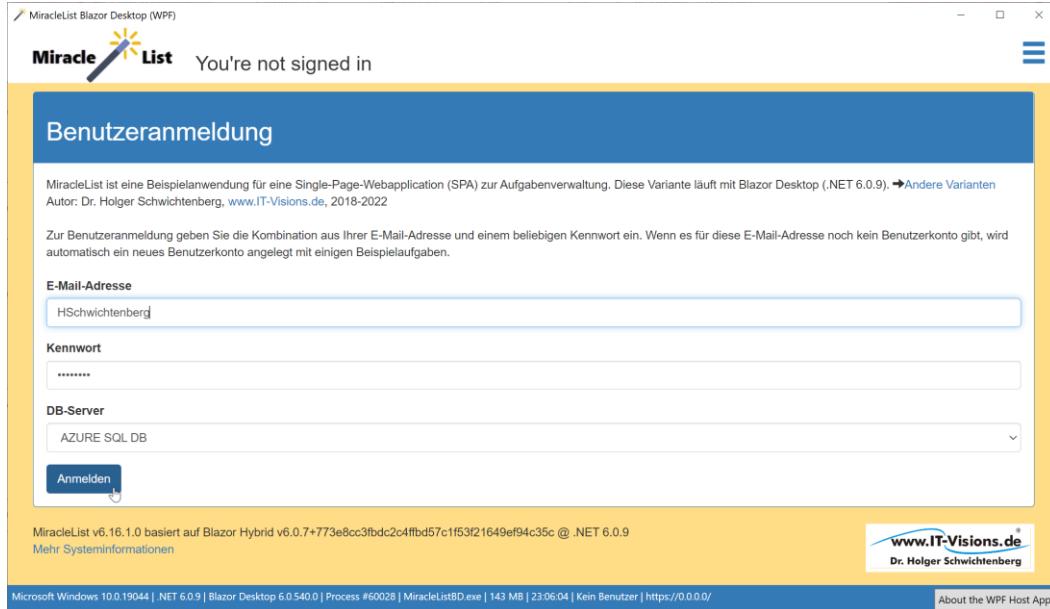


Abbildung: MiracleList in der Blazor Desktop-Variante, gehostet in einer Windows-Desktop-Anwendung mit Windows Presentation Foundation (WPF) mit direktem Datenbankzugriff wie bei Blazor Server

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

Hinweis: Neue Benutzer erhalten automatisch eine Reihe von Beispielaufgaben!

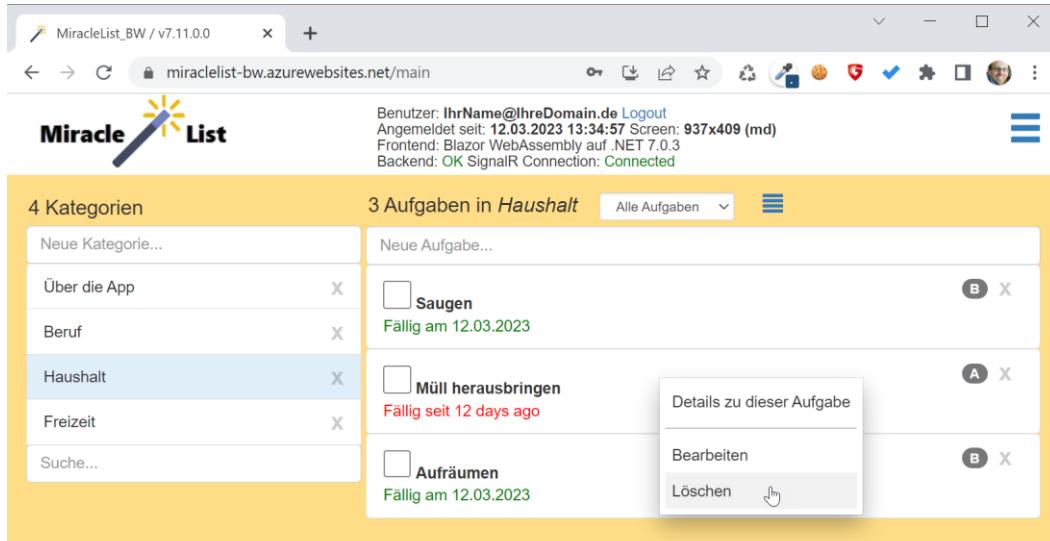


Abbildung: Hauptansicht der MiracleList-Anwendung mit Aufgabenfilter und Kontextmenü für die einzelnen Aufgaben



Abbildung: Hauptansicht der MiracleList-Anwendung in der in WPF gehosteten Variante "Blazor Desktop" mit zusätzlicher Export-Funktion in Dateisystem

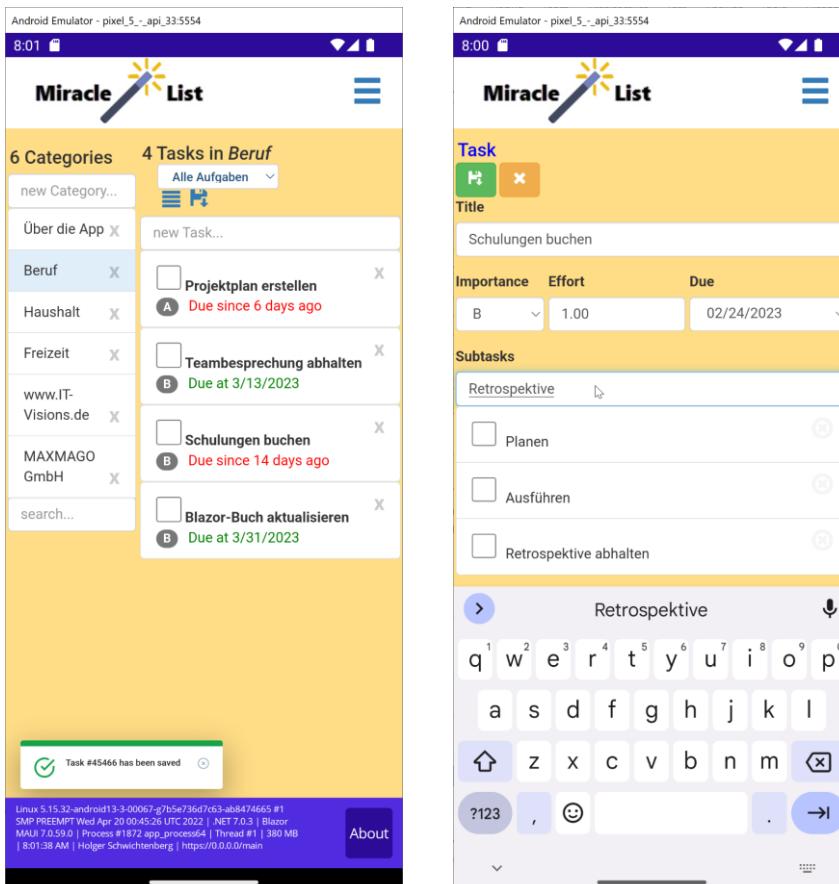


Abbildung: Hauptansicht und Bearbeitungsansicht der MiracleList-Anwendung in einem Android-basierten SmartPhone (implementiert mit Blazor MAUI)

Durch Klick auf eine Aufgabe kommt der Benutzer zur Bearbeitungsansicht, die rechts als dritte Spalte erscheint. Einige Eingabefelder besitzen Validierungsbedingungen, bei deren Nicht-Einhaltung die Felder rot umrandet werden und es erscheint eine Fehlermeldung. Beim Titel erscheint auch ein rotes x. Auch das Hochladen von Dateien zu einer Aufgabe ist möglich.

The screenshot shows the MiracleList web application interface. On the left, a sidebar lists categories: '4 Kategorien' (Neue Kategorie..., Über die App, Beruf, Haushalt, Freizeit, Beispiel). The main content area displays search results for '4 Suchergebnisse in 4 Kategorien' (1 Aufgabe gefunden in Kategorie Über die App, 1 Aufgabe gefunden in Kategorie Beruf, 1 Aufgabe gefunden in Kategorie Haushalt, 1 Aufgabe gefunden in Kategorie Freizeit). A specific task, 'Beispieldaufgaben erforschen' (Due at 21.09.2022), is selected and shown in a detailed editing dialog on the right. The dialog includes fields for 'Titel' (Title), 'Wichtigkeit' (Importance), 'Aufwand' (Effort), and 'Fälligkeit' (Due Date). It also shows validation errors: 'Title must be at least 1 character.' and 'Due date must not be in the past'. Other sections visible include 'Unteraufgaben' (Subtasks), 'Notizen' (Notes), and 'Dateien' (Files).

Abbildung: Suchergebnisse in der Mitte und Bearbeitungsansicht inklusive Validierung rechts für die ausgewählte Aufgabe

The screenshot shows the MiracleList application running on Windows. The left sidebar shows categories: '4 Categories' (new Category..., Über die App, Beruf, Haushalt, Freizeit). The main content area displays a task list for 'Über die App'. A context menu is open over the first task, 'Beispieldaufgaben erforschen', with options like 'Send this task list to Microsoft Word and save in C:\Users\hs\Documents\MiracleList_Export_Uber die App_20220920.docx'. A Microsoft Word window is open, showing the task list as a Word document. The Word ribbon is visible, and the document content matches the tasks listed in the application. The status bar at the bottom of the application window shows 'Microsoft Windows 10.0.19044 | .NET 6.0.8 | Blazor MAUI 6.0.409.77'.

Abbildung: Eine Export-Funktion in ein Microsoft Word-Dokument via COM-Objektmodell gibt es nur in Blazor Desktop (immer auf Windows) und Blazor MAUI (wenn es auf Windows läuft)

The screenshot shows a web browser window with the URL miraclelist-bw.azurewebsites.net/About. The page content is as follows:

Technische Daten

System DateTime: 12.03.2023 13:38:54
 System DateTime MEZ: 12.03.2023 13:38:54
 Operating System: Browser (Other 1.0.0.0)
 OS Architecture: Wasm (32 Bit)
 Process Architecture: Wasm (32 Bit)
 .NET Runtime Variant: .NET Core
 .NET Runtime Version: .NET 7.0.3
 Target Framework: .NETCoreApp, Version=v7.0
 Execution Mode: Release
 Internal Application Name: MiracleList_BW
 Copyright: Dr. Holger Schwichtenberg, www.IT-Visions.de 2019-2023
 Application Version: 7.11.0.0
 Application Informational Version: 7.11.0.0-03/09/2023 21:01:33
 Directory GetCurrentDirectory: /
 Environment Current Directory: /
 ApplicationContext Base Directory: /
 Environment SpecialFolder ApplicationData: Environment SpecialFolder MyDocuments: /home/web_user
 Command Line:
 Process Path: Process_PlatformNotSupportedException
 Process ID: Process_PlatformNotSupportedException
 Assembly Location: unknown
 Assembly File Size: unknown
 Assembly File Datetime: Fatal Error with WASI
 System Name: localhost
 Processor Count: 1
 Total Memory: n/a
 Process Memory: n/a
 Current User: localhost/Browser
 User Interactive: True
 Web Framework: Blazor WebAssembly v7.0.3.0 @ .NET 7.0.3
 JavaScript Runtime: Microsoft.AspNetCore.Components.WebAssembly.Services.DefaultWebAssemblyJSRuntime
 Browser: Chrome 110.0.0 @ Windows 10 (Device: Other)
 Browser Details: Browser Type: Mozilla / Browser Version: 5.0 (Windows NT 10.0; Win64; x64; AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.0.0 Safari/537.36
 Client IP: n/a
 Current Culture: de-DE (de (DE))
 RAM used: ...
 Network Latency: 1 ms
 Base Address: <https://miraclelist-bw.azurewebsites.net/>
 Environment: Production

Backend-Server

URL: <https://miraclelistbackend.azurewebsites.net/>
 Application Version: 7.11.0.0
 .NET Version: .NET 7.0.2 (.NETCoreApp, Version=v7.0 (v7.0.2))

Über MiracleList

MiracleList ist eine umfangreiche, sehr realitätsnahe Benutzeroberfläche und Funktionen wie das inzwischen. Diese Beispieldatenanwendung verwendet Dr. Holger Schulten und Vorträgen.

Während in vielen Veröffentlichungen und Weiterleitungen kleine, aus dem Kontext gerissene Beispiele gezeigt werden, ist es in MiracleList möglich viele Funktionen in einem praxisnahen Gesamtzusammenhang zu zeigen.

Vier Blazor-Implementierungen der MiracleList

Es gibt vier auf ASP.NET Core Blazor basierende Implementierungen des MiracleList-Frontends:

- **MiracleList_BW:** MiracleList mit Blazor WebAssembly (3-Tier mit Zugriff auf die Daten über WebAPI-basierten Application Server)
- **MiracleList_BS:** MiracleList mit Blazor Server (2-Tier mit Direktzugriff auf die Datenbank)
- **MiracleList_BD:** MiracleList mit Blazor Desktop in einer WPF-Rahmenanwendung (2-Tier mit Direktzugriff auf die Datenbank)
- **MiracleList_BM:** MiracleList mit Blazor MAUI (3-Tier mit Zugriff auf die Daten über WebAPI-basierten Application Server)

Features der Blazor-Implementierung der MiracleList

Welche Funktionen die Blazor-Implementierungen der MiracleList aus Benutzersicht bieten, finden Sie in der Tabelle auf www.MiracleList.net

Aus technischer Sicht demonstrieren die Blazor-Implementierungen der MiracleList für Softwareentwickler folgende Funktionen von Blazor:

- Alle vier Blazor-Varianten (Blazor WebAssembly, Blazor Server, Blazor Desktop, Blazor MAUI) für Browser und als Hybridanwendung
- Sehr viel Shared Code zwischen allen Blazor-Varianten mit einem gemeinsamen UI in einer Razor Class Library
- Abstraktion zwischen einer 2-Tier- und 3-Tier-Anwendung: Mit dem gleichen Komponenten sowohl direkt auf eine Datenbank zugreifen als auch Nutzung von WebAPIs auf einem Application Server
- Listenansichten mit Suchen und Filtern
- Bearbeitungsformulär mit Validierung
- Datei-Upload
- Tastaturereignisse
- Editerbares Datagrid
- Drag&Drop
- Kontextmenüs
- Modale Dialoge
- Toast-Nachrichten
- Ständig aktualisierte Statusanzeige für Backendssysteme
- Generierung eines Berichts via Microsoft Word (nur in Hybridanwendung)
- Authentifizierung und Autorisierung
- Nutzung des Local Storage des Browsers

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 einfach nicht in das Szenario "MiracleList".
- Aus didaktischen Gründen ist es an einigen Stellen geboten, einfachere Beispiele zu behandeln.

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.

Blazor-Beispiele aus dem Buch 'ASP.NET Core Blazor'

Viele (aber nicht alle) Beispiele aus dem Buch drehen sich um das MiracleList-Szenario. Diese Liste, sortiert nach Buchkapiteln, umfasst Geschäftslogik von MiracleList beziehen, weil sie fachlich nicht zu MiracleList passen oder weil didaktisch besser ist, ein Feature oder V

Überblick über Blazor

- Performancemessungen
- API-Zugriffe

Übersetzung und Debugging

- Debugging

Komponenten (Razor Components)

- Komponente nur mit .razor-Datei und Inline-Code
- Code-Behind mit Vererbung
- Code-Behind mit partieller Klasse
- Code-Behind nur mit Code
- Nutzung einer nicht-visuellen Komponente
- Komponenten ohne Razor nur aus Programmcode

Routing und Navigation

- Routing und Navigation

Razor-Syntax

- Razor-Syntax
- Lottospiel
- Razor-Listen

Ereignisse und Ereignisbehandlung

- Razor-Ereignisse

Abbildung: Die Blazor-Beispiele außerhalb des MiracleList-Szenarios sind in das Menü der MiracleList integriert und dort gut nach Buchkapitel sortiert.

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_BW/Demos und /src/MiracleList_BS/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.

Wichtig: Für .NET 9.0 und Blazor 9.0 brauchen Sie mindestens Version 17.12 von Visual Studio 2022!

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

6.6.1 Varianten von Visual Studio

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.

6.6.2 Stabile Version und Preview-Version

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

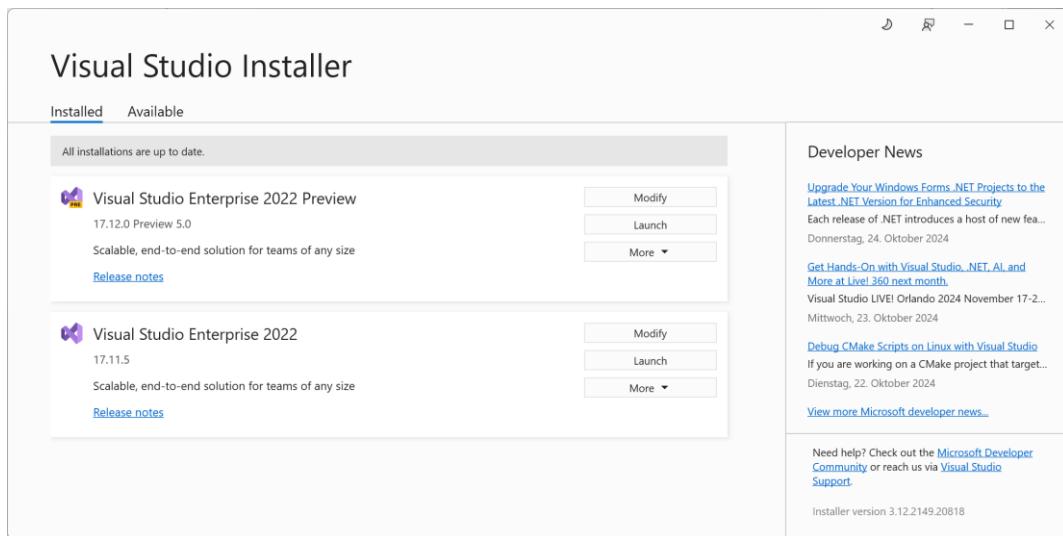


Abbildung: Parallelinstallation der stabilen Version und einer Preview-Version

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, 17.4, 17.5 usw.

6.6.3 Workloads

Visual Studio besteht aus vielen Bausteinen, die Optionen bei der Installation darstellen. Vorgefertigte Zusammenstellungen von diesen Bausteinen heißen Workloads.

Hinweis: Sowohl Visual Studio als auch das .NET SDK kennen den Begriff "Workload". Ein Workload in Visual Studio ist aber nicht das gleiche wie im .NET SDK.

Für die Softwareentwicklung mit Blazor müssen Sie in Visual Studio den Workload "ASP.NET and web development" installieren. Der Workload "Azure development" wird nur benötigt, wenn Sie die Anwendung direkt in Microsoft Azure verbreiten möchten, ohne DevOps-Pipeline.

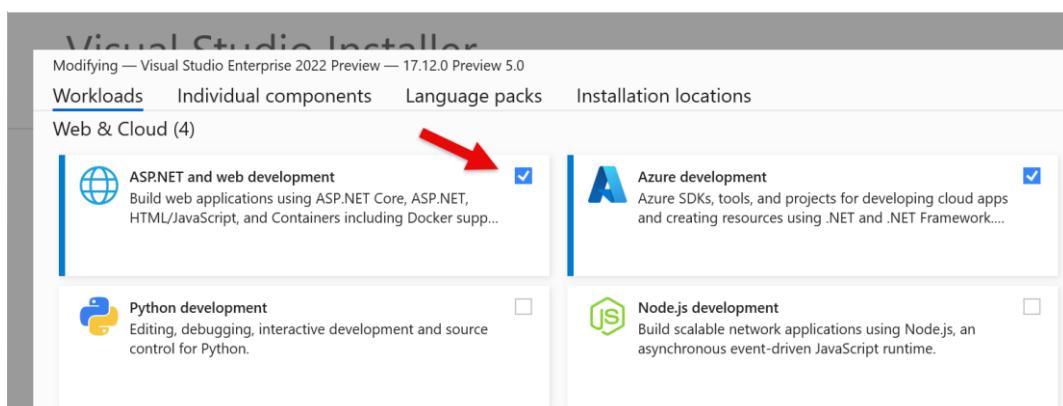


Abbildung: Notwendige Visual Studio-Workloads für Blazor

6.6.4 Sprachversionen

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

Bezeichnung nicht kennen. Ich werde daher in diesem Buch immer nur die englischen Namen für Menüpunkte, Einstellungen etc. verwenden.

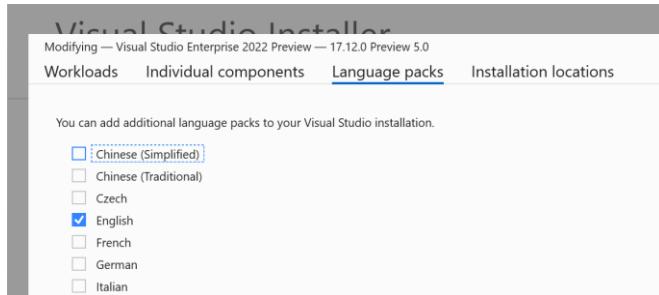


Abbildung: Die Sprachversion wählen Sie über "Language Packs" im Visual Studio-Setup

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

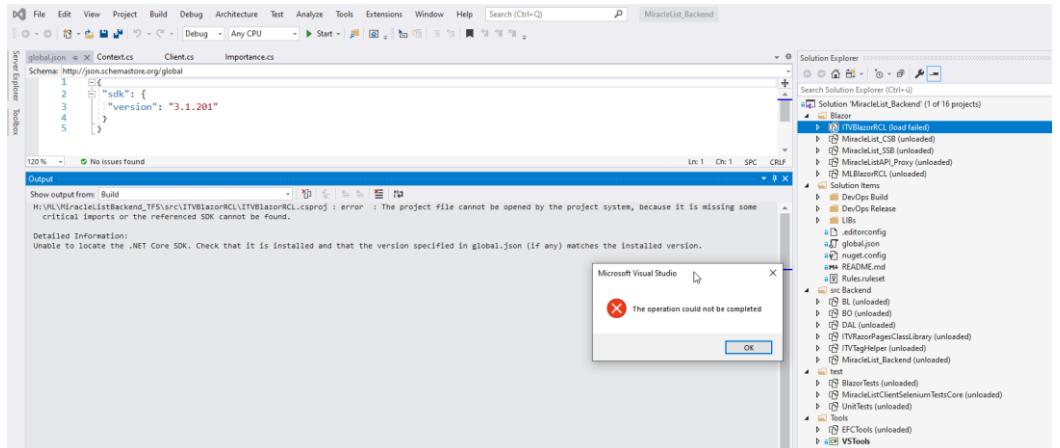


Abbildung: Die notwendige SDK-Version fehlt

Sie können die installierten SDK-Versionen an der Kommandozeile abrufen:

```
dotnet --list-sdks
```

```
pwsh in T:\ dotnet --list-sdks
2.1.818 [C:\Program Files\dotnet\sdk]
3.1.426 [C:\Program Files\dotnet\sdk]
5.0.103 [C:\Program Files\dotnet\sdk]
5.0.203 [C:\Program Files\dotnet\sdk]
5.0.302 [C:\Program Files\dotnet\sdk]
5.0.403 [C:\Program Files\dotnet\sdk]
5.0.407 [C:\Program Files\dotnet\sdk]
5.0.408 [C:\Program Files\dotnet\sdk]
6.0.125 [C:\Program Files\dotnet\sdk]
6.0.129 [C:\Program Files\dotnet\sdk]
6.0.203 [C:\Program Files\dotnet\sdk]
6.0.320 [C:\Program Files\dotnet\sdk]
6.0.321 [C:\Program Files\dotnet\sdk]
6.0.417 [C:\Program Files\dotnet\sdk]
6.0.421 [C:\Program Files\dotnet\sdk]
7.0.100-rc.1.22431.12 [C:\Program Files\dotnet\sdk]
7.0.114 [C:\Program Files\dotnet\sdk]
7.0.118 [C:\Program Files\dotnet\sdk]
8.0.204 [C:\Program Files\dotnet\sdk]
8.0.300-preview.24203.14 [C:\Program Files\dotnet\sdk]
9.0.100-preview.3.24204.13 [C:\Program Files\dotnet\sdk]
```

Abbildung: Abruf der Liste der auf einem System installierten .NET SDKs via `dotnet --list-sdks`

Seit .NET 6.0 ist es möglich, den Aktualitätsstand der installierten SDK- und Runtime-Versionen abzufragen. Der Befehl dazu lautet:

`dotnet sdk check`

Hinweis: Der Kommandozeilenbefehl `dotnet sdk check` ruft einen Webservice bei Microsoft auf. Das heißt aber auch: Auf einem Entwicklerrechner ohne Internetzugang (ja, sowas gibt es tatsächlich in einigen Firmen) funktioniert `dotnet sdk check` nicht.

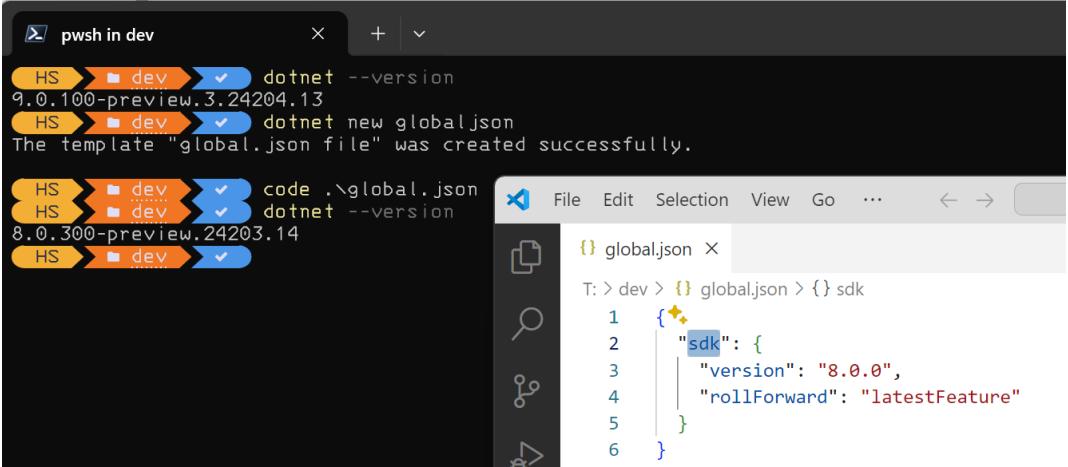
Es gibt grundsätzlich zwei Verfahren, wie die .NET SDK-Version gewählt wird:

- 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` in einem Dateisystemordner 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 eines der bei Ihnen installierten .NET SDKs. Dabei müssen Sie keine exakte Versionsnummer angeben: Sie können mit "rollForward": "latestFeature" bzw. "rollForward": "latestMinor" bestimmen, dass aktuellere als die angegebene Version verwendet werden dürfen.



pwsh in dev

```
dotnet --version
9.0.100-preview.3.24204.13
dotnet new global.json
The template "global.json file" was created successfully.

code .\global.json
dotnet --version
8.0.300-preview.24203.14
```

global.json

```
{
  "sdk": {
    "version": "8.0.0",
    "rollForward": "latestFeature"
  }
}
```

Abbildung: Erstellen einer `global.json`-Datei mit Einstellung der Verwendung der aktuellsten installierten SDK-Version zu der angegebenen Hauptversion.

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 .NET WebAssembly Build Tools

Für einige Funktionen bei Blazor WebAssembly wie AOT-Kompilierung und Native Code-Aufruf werden zusätzlich zum .NET SDK die .NET WebAssembly Build Tools benötigt.

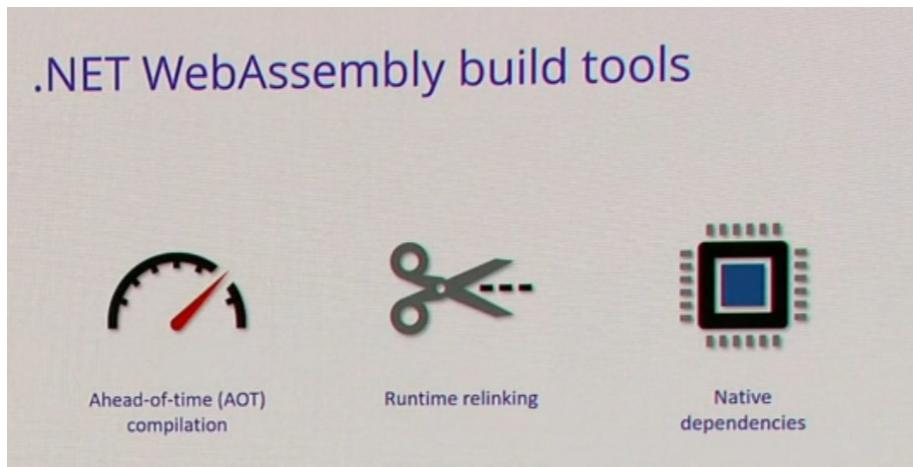


Abbildung: Funktionen der .NET WebAssembly Build Tools (Quelle: Microsoft .NET Conf 2021)

Die ".NET WebAssembly Build Tools" installiert man via Visual Studio (siehe Abbildung)

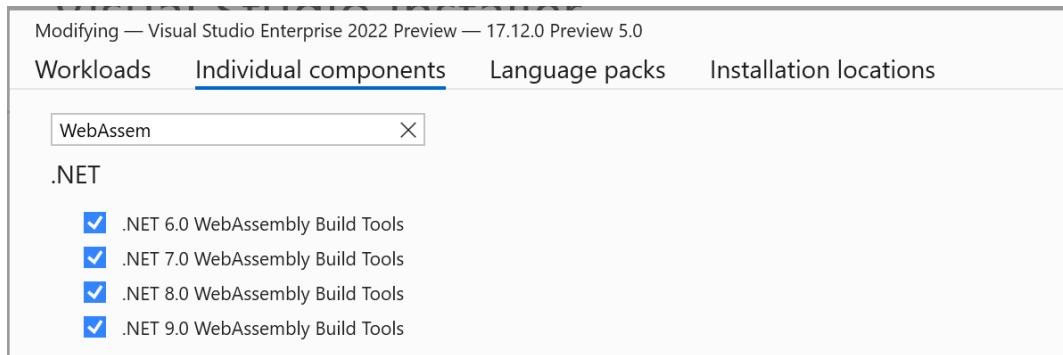


Abbildung: Installation der .NET WebAssembly Build Tools in Visual Studio

oder via Kommandozeile

```
dotnet workload install wasm-tools
```

6.9 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.

Geben Sie im Suchdialog "WebAssembly" ein, siehe Abbildung. Wählen Sie "Blazor WebAssembly Standalone App". Nach der Auswahl des Namens (geben Sie ein "MiracleListBW" für den Projektnamen 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 8.0", "Authentication=None", "Configure for HTTPS", "Progressive Web App" und "Include sample pages" (damit man Boostrap als CSS-Framework und drei Beispielseiten bekommt= sowie "Do not use top-level statements" (damit man eine klassische .NET-Startklasse statt losem Startcode bekommt) zu wählen.

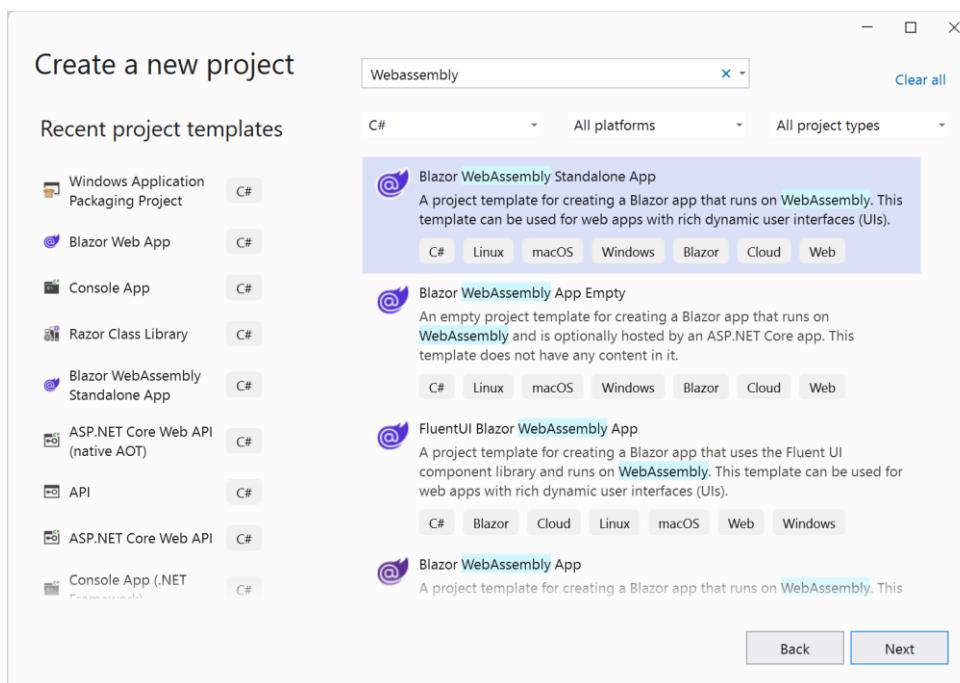


Abbildung: Wahl der Projektvorlage