

7.4-2: Einbindung MongoDB

Hinweis: Dieser Auftrag setzt auf erledigtem Auftrag 01-AA-Minimal-API-MongoDB auf.

1. MongoDB-Container

Starten Sie einen MongoDB-Container, der folgende Anforderungen erfüllt:

- Ausführung im Hintergrund
- Das Datenverzeichnis ist in ein named Volume zu mounten.
- Der Port der Datenbank ist 1:1 zu mappen.

```
docker run --name mongodb -d -p 27017:27017 -v mongovolume:/var/lib/db mongo
```

```
vmadmin@lp-22-04:~/Documents/min-api-with-mongo$ docker run --name mongodb -d -p 27017:27017 -v mongovolume:/var/lib/db mongo
7f9eef8b5faef909b4ae0445386615dde958b99bfff5da5e0ee601197bc391337
```

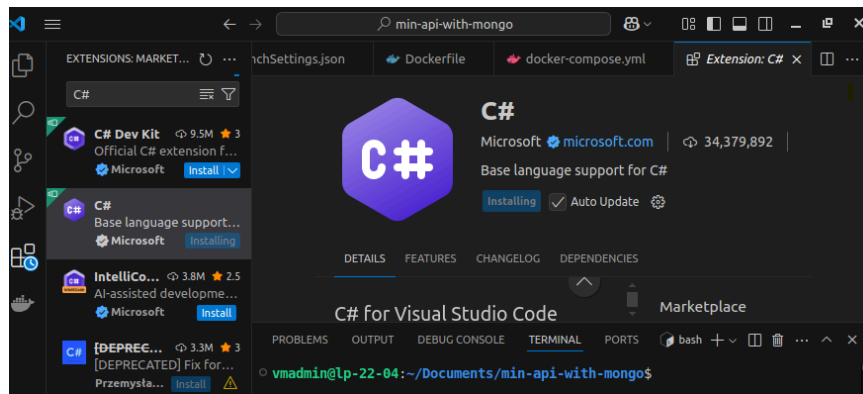
Vergewissern Sie sich, dass der Container läuft.

```
docker ps
```

```
vmadmin@lp-22-04:~/Documents/min-api-with-mongo$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS
7f9eef8b5fae   mongo     "docker-entrypoint.s..." About a minute ago Up About a minute   0.0.0.0:27017->27017/tcp, [::]:27017->27017/tcp
d15c53511b67   min-api-with-mongo-webapi "dotnet WebApi.dll"     6 days ago    Up 36 minutes    0.0.0.0:5001->5001/tcp, [::]:5001->5001/tcp
```

2. Installation VS Code-Extension für C sharp

Um VS-Code mit Syntax Highlighting, IntelliSense, etc. zu erweitern, installieren Sie die Extension C#.



3. MongoDB-Container

Für den Zugriff auf eine MongoDB existiert ein offizieller .NET MongoDB.Driver. Installieren Sie das Nuget-Package mit folgendem .NET CLI Command:

```
dotnet add package MongoDB.Driver
```

Hinweis: Achten Sie darauf, dass Sie sich im Projektverzeichnis der .NET-Anwendung (min-api-with-mongo/WebApi) befinden.

```
vmadmin@lp-22-04:~/Documents/min-api-with-mongo$ cd WebApi/  
vmadmin@lp-22-04:~/Documents/min-api-with-mongo/WebApi$ dotnet add package MongoDB.Driver  
Determining projects to restore...
```

Erzeugen Sie in WebApi/Program.cs unter app.MapGet(...) einen weiteren Endpunkt check:

```
app.MapGet("/check", () => {  
    /* Code zur Prüfung der DB ...*/  
    return "Zugriff auf MongoDB ok.";  
});
```

```

WebApi > Program.cs
1  var builder = WebApplication.CreateBuilder(args);
2  var app = builder.Build();
3
4  app.MapGet("/", () => "Hello World!");
5
6  app.MapGet("/check", () => {
7      /* Code zur Prüfung der DB ...*/
8      return "Zugriff auf MongoDB ok.";
9  });
10
11 app.Run();

```

Der Aufruf von `http://localhost:5001/check` soll zeigen, ob der Zugriff auf die MongoDB funktioniert. (Applikation starten mit `dotnet watch run`)



Erweitern Sie die eben eingefügte Methode so, dass folgende Anforderungen erfüllt werden:

- Die Verbindung zur MongoDB wird über `MongoDB.Driver.MongoClient` aufgebaut.
- die vorhandenen Datenbanken werden abgefragt und in der Antwort ausgegeben.
- Exceptions sind mit `try/catch` abgefangen und werden als Fehlermeldung zurückgegeben.


Hinweis: Der Connections-String darf fix programmiert werden. Wir werden ihn später konfigurierbar machen. Der Aufbau des Connection-Strings ist in der MongoDB-Dokumentation beschrieben.

```

WebApi > Program.cs
1 using MongoDB.Driver;
2
3 var builder = WebApplication.CreateBuilder(args);
4 var app = builder.Build();
5
6 app.MapGet("/", () => "Hello World!");
7
8 app.MapGet("/check", () =>
9 {
10     try
11     {
12         var mongoDbConnectionString = "mongodb://localhost:27017";
13         var mongoClient = new MongoClient(mongoDbConnectionString);
14         var databaseName = mongoClient.ListDatabaseNames().ToList();
15         return $"Zugriff auf MongoDB ok. Vorhandene DBs: {String.Join(", ", databaseName)}";
16     }
17
18     catch(System.Exception e)
19     {
20         /* Code zur Prüfung der DB ...*/
21         return "Zugriff auf MongoDB nicht ok." + e.Message;
22     }
23 });
24
25 app.Run();

```

▼ Erklärung



← → ↻ localhost:5001/check

Zugriff auf MongoDB ok. Vorhandene DBs: admin, config, local

4. Konfiguration des Connection-String

Durch Umsetzung des Options Patterns soll der Connection-String in appsettings.json konfiguriert werden können.

Erstellen Sie unter min-api-with-mongo/WebApi ein neues File DatabaseSettings.cs mit folgendem Inhalt:

```

public class DatabaseSettings
{
    public string ConnectionString { get; set; } = "";
}

```

```

WebApi > DatabaseSettings.cs > ...
0 references
1 public class DatabaseSettings
2 {
3     0 references
4     public string ConnectionString { get; set; } = "";
}

```

Erweitern Sie min-api-with-mongo/WebApi/appsettings.json um den Abschnitt DatabaseSettings und weisen Sie ConnectionString den bis jetzt fix codierten

Wert zu:

```
"AllowedHosts": "*",
"DatabaseSettings": {
  "ConnectionString": "mongodb://gbs:geheim@localhost:27017"
}
```

```
WebApi > bin > Debug > net8.0 > {} appsettings.json > ...
1  {
2    "Logging": {
3      "LogLevel": {
4        "Default": "Information",
5        "Microsoft.AspNetCore": "Warning"
6      }
7    },
8    "AllowedHosts": "*",
9    "DatabaseSettings": {
10     "ConnectionString": "mongodb://localhost:27017"
11   }
12 }
```

Erweitern Sie `min-api-with-mongo/WebApi/Program.cs`, um die `DatabaseSettings` als Service für DependencyInjection zu registrieren. Fügen Sie dazu nach `var builder = WebApplication.CreateBuilder(args);` folgende zwei Codezeilen ein:

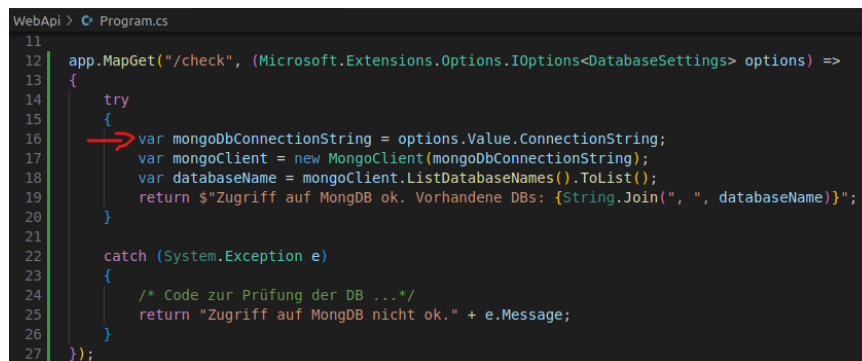
```
var movieDatabaseConfigSection = builder.Configuration.GetSection("DatabaseSettings");
builder.Services.Configure<DatabaseSettings>(movieDatabaseConfigSection);
```

Um die `DatabaseSettings` zu injecten, ersetzen Sie `pp.MapGet("/check", () => {` mit `pp.MapGet("/check", (Microsoft.Extensions.Options.IOptions<DatabaseSettings> options) => {`

```
WebApi > Program.cs
12 app.MapGet("/check", (Microsoft.Extensions.Options.IOptions<DatabaseSettings> options) =>
13 {
14     try
15     {
16         var mongoDbConnectionString = "mongodb://localhost:27017";
17         var mongoClient = new MongoClient(mongoDbConnectionString);
18         var databaseName = mongoClient.ListDatabaseNames().ToList();
19         return $"Zugriff auf MongoDB ok. Vorhandene DBs: {String.Join(", ", databaseName)}";
20     }
21     catch (System.Exception e)
22     {
```

Über `options.Value` haben Sie Zugriff auf alle Werte von `DatabaseSettings`. Ersetzen Sie die fixe Zuweisung wie folgt:

```
var mongoDbConnectionString = options.Value.ConnectionString;
```



```
WebApi > Program.cs
11
12 app.MapGet("/check", (Microsoft.Extensions.Options.IOptions<DatabaseSettings> options) =>
13 {
14     try
15     {
16         → var mongoDbConnectionString = options.Value.ConnectionString;
17           var mongoClient = new MongoClient(mongoDbConnectionString);
18           var databaseName = mongoClient.ListDatabaseNames().ToList();
19           return $"Zugriff auf MongoDB ok. Vorhandene DBs: {String.Join(" ", databaseName)}";
20     }
21
22     catch (System.Exception e)
23     {
24         /* Code zur Prüfung der DB ...*/
25         return "Zugriff auf MongoDB nicht ok." + e.Message;
26     }
27 });
```

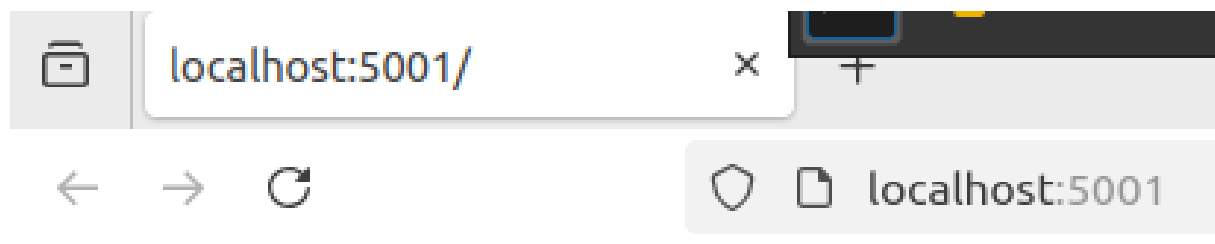
Starten Sie die Anwendung und vergewissern Sie sich, dass sie den mongoDbConnectionString aus appsettings.json verwendet.

5. docker-compose erweitern

Erweitern Sie min-api-with-mongo/docker-compose.yml, dass nebst dem WebApi auch ein MongoDB-Container gestartet wird. Beachten Sie, dass das API erst gestartet werden soll, wenn die MongoDB verfügbar ist.

Hinweise: In ASP.NET Core können Settings aus appsetting.json mit Hilfe von Umgebungsvariablen übersteuert werden. Setting MoviesDatabaseSettings.ConnectionString wird mit Umgebungsvariable MoviesDatabaseSettings_ConnectionString übersteuert. (Ein Punkt im Pfad wird durch zwei ersetzt) Das ermöglicht Ihnen, den Connection-String der MongoDB per Umgebungsvariable zu setzen.

```
vmadmin@lp-22-04:~/Documents/min-api-with-mongo/WebApi$ dotnet watch run
```



Hello World!