**Controller in ASP.NET Core**

Ein **Controller** in **ASP.NET Core** ist eine **Klasse**, die HTTP-Anfragen verarbeitet und darauf reagiert. Er ist das **Herzstück des MVC (Model-View-Controller)-Musters** oder einer **API-Struktur**.

In einer **Webanwendung**:

* Ein Controller empfängt HTTP-Requests, verarbeitet Daten und gibt eine View (HTML-Seite) oder JSON zurück.

In einer **Web-API**:

* Ein Controller stellt API-Endpunkte bereit und gibt meistens JSON-Daten zurück.

**Aufbau eines Controllers**

Ein Controller:

* Erbt von Controller (bei MVC) oder ControllerBase (bei APIs).
* Enthält Methoden (sog. **Action-Methoden**), die auf HTTP-Anfragen reagieren.
* Wird über Routen ([Route] oder [Http...]) angesprochen.
* Arbeitet oft mit **Services** und **Datenbanken** zusammen.

**Beispiel: Feuerwehr-Controller für eine API**

[Route("api/[controller]")] // URL: /api/feuerwehr

[ApiController] // Kennzeichnet die Klasse als API-Controller

public class FeuerwehrController : ControllerBase

{

private static List<Einsatz> einsaetze = new()

{

new Einsatz { Id = 1, Beschreibung = "Wohnungsbrand", Ort = "Hauptstraße 12" },

new Einsatz { Id = 2, Beschreibung = "Verkehrsunfall", Ort = "Autobahn A3" }

};

// Alle Einsätze abrufen (GET: /api/feuerwehr)

[HttpGet]

public IActionResult GetAll() => Ok(einsaetze);

// Einzelnen Einsatz abrufen (GET: /api/feuerwehr/1)

[HttpGet("{id}")]

public IActionResult GetById(int id)

{

var einsatz = einsaetze.FirstOrDefault(e => e.Id == id);

return einsatz == null ? NotFound() : Ok(einsatz);

}

// Neuen Einsatz hinzufügen (POST: /api/feuerwehr)

[HttpPost]

public IActionResult Create(Einsatz einsatz)

{

einsaetze.Add(einsatz);

return CreatedAtAction(nameof(GetById), new { id = einsatz.Id }, einsatz);

}

// Einsatz aktualisieren (PUT: /api/feuerwehr/1)

[HttpPut("{id}")]

public IActionResult Update(int id, Einsatz einsatz)

{

var existing = einsaetze.FirstOrDefault(e => e.Id == id);

if (existing == null) return NotFound();

existing.Beschreibung = einsatz.Beschreibung;

existing.Ort = einsatz.Ort;

return NoContent();

}

// Einsatz löschen (DELETE: /api/feuerwehr/1)

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

var einsatz = einsaetze.FirstOrDefault(e => e.Id == id);

if (einsatz == null) return NotFound();

einsaetze.Remove(einsatz);

return NoContent();

}

}

// Model-Klasse für einen Feuerwehreinsatz

public class Einsatz

{

public int Id { get; set; }

public string Beschreibung { get; set; }

public string Ort { get; set; }

}

**Beispiel: Feuerwehr-Controller für eine MVC-Webanwendung**

Ein **MVC-Controller** (für Webseiten mit Razor Views) sieht ähnlich aus, enthält aber View()-Methoden.

using Microsoft.AspNetCore.Mvc;

public class FeuerwehrController : Controller

{

// Zeigt die Startseite mit einer Liste aller Einsätze

public IActionResult Index()

{

var einsaetze = new List<Einsatz>

{

new Einsatz { Id = 1, Beschreibung = "Wohnungsbrand", Ort = "Hauptstraße 12" },

new Einsatz { Id = 2, Beschreibung = "Verkehrsunfall", Ort = "Autobahn A3" }

};

return View(einsaetze);

}

// Zeigt Details zu einem bestimmten Einsatz

public IActionResult Details(int id)

{

var einsatz = new Einsatz { Id = id, Beschreibung = "Test-Einsatz", Ort = "Test-Ort" };

return View(einsatz);

}

}

**Views:**

* Die Index()-Methode würde eine Index.cshtml-View rendern.
* Die Details()-Methode würde eine Details.cshtml-View rendern.

**Vergleich: MVC-Controller vs. API-Controller**

|  |  |  |
| --- | --- | --- |
| Feature | MVC-Controller | API-Controller |
| Basisklasse | Controller | ControllerBase |
| Rückgabe | View(), Redirect(), Json() | Ok(), NotFound(), Created() |
| Verwendung | HTML-Seiten (Views) | JSON- oder XML-Daten |
| Typisches Projekt | Webanwendung (mit Razor Views) | REST-API für Webdienste |

**Zusammengefasst**

Ein **Controller** ist das zentrale Element einer ASP.NET Core Anwendung, das Anfragen verarbeitet.  
Je nach Anwendungstyp gibt es:

* **MVC-Controller** für Webanwendungen mit Razor Views.
* **API-Controller** für REST-APIs, die JSON-Daten liefern.