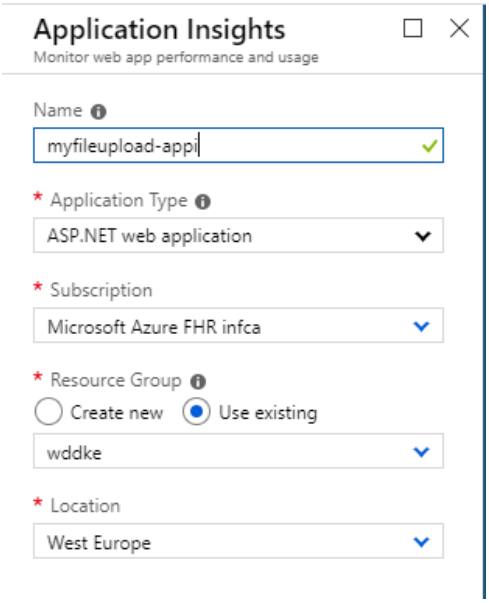
Monitor Web App und Azure Function Hands On

Für den im Azure WebApps + Storage + Functions Hands On implementierten UseCase wird ein Monitoring implementiert.

- Hinzufügen von Application Insights zur Middleware
- Hinzufügen von Application Insights für Azure Functions
    - Auswerten eines fehlerhaften Uploads
- Custom Logging
    - Hinzufügen von Custom Monitoring
        - In der Middleware mit dem TelemetryClient
    - In der UI
        - Tracken von Page Views mit einem custom monitoring service
- Anzeigen/Auswerten im Portal

Requirements:

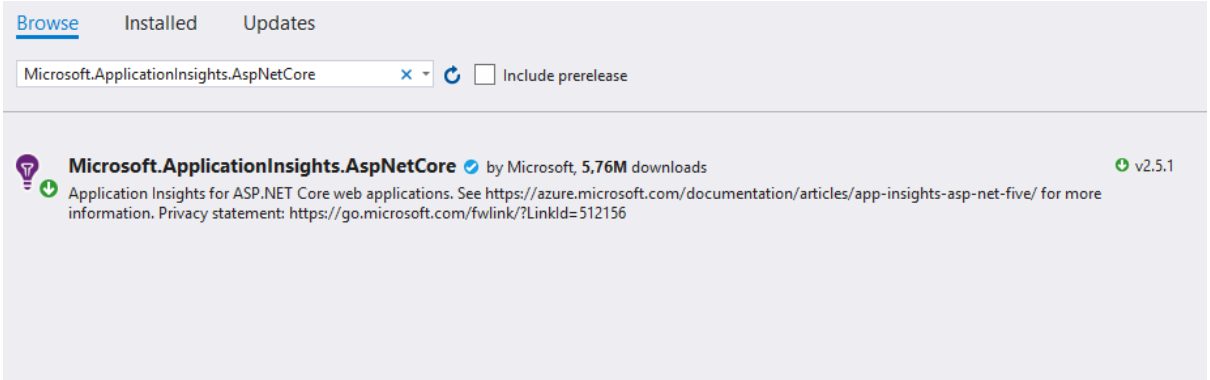
- Application Insights Resourcen (3 Stück) in eigener RG erstellen und sprechend benennen.
    - Middleware
        - Application Insights für die ASP.NET web application erstellen
    - Function
        - Application Insights für die ASP.NET web application erstellen
    - UI
        - Application Insights für Node.js application erstellen (UI)



1 Implementieren von Application Insights in die Middleware

In Visual Studio in die File Upload App wechseln

Unter Manage NuGet Packages App Insights hinzufügen und auf aktuelle Version (2.5.1) prüfen



Startup.cs

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_1);
    services.AddApplicationInsightsTelemetry(Configuration);

    // In production, the Angular files will be served from this directory
    services.AddSpaStaticFiles(configuration =>
    {
        configuration.RootPath = "ClientApp/dist";
    });
}
```

Program.cs

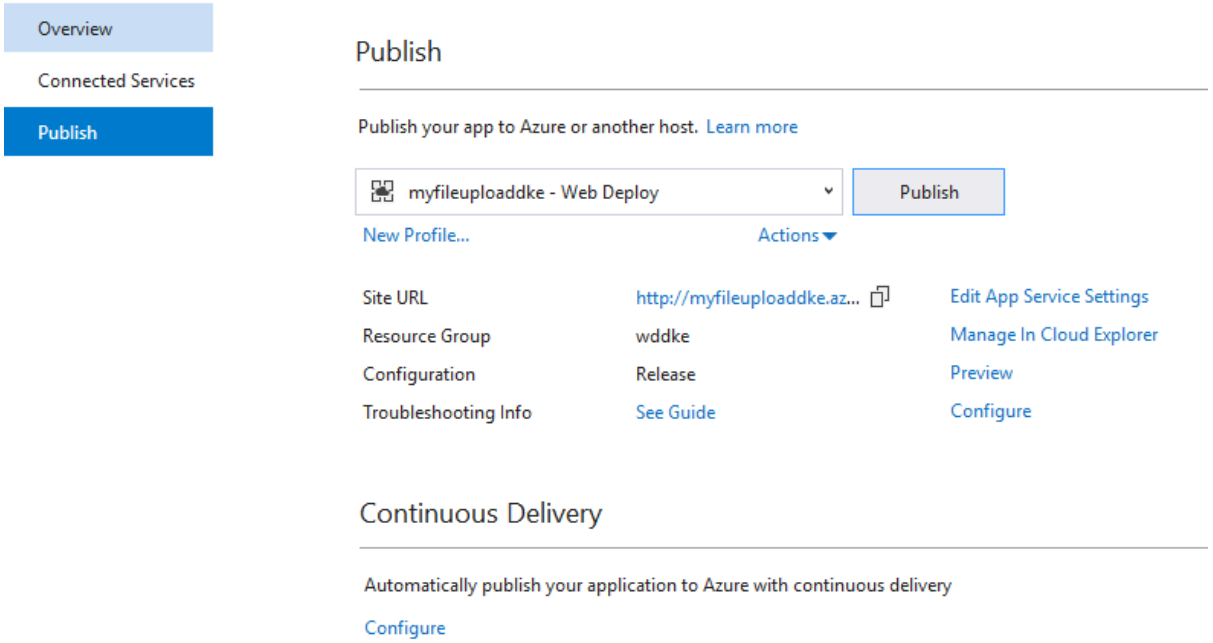
```
1 reference | 0 exceptions
public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
    WebHost.CreateDefaultBuilder(args)
        .UseStartup<Startup>().UseApplicationInsights();
```

Appsettings.json (Instrumentation Key der richtigen App Insights Resource eintragen)

```
"ApplicationInsights": {
    "InstrumentationKey": "XXX"

},
```

Ausführen

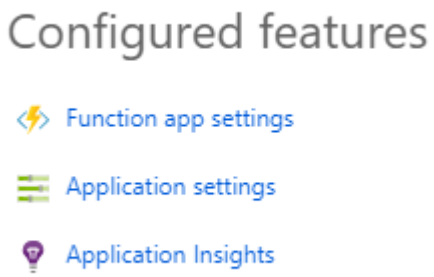
- Lokal ausführen und prüfen ob in der Cloud App Insights resource etwas ankommt (Kann ein wenig Zeit dauern)
- Wenn es funktioniert die App wieder Publishen
- Dann Online testen



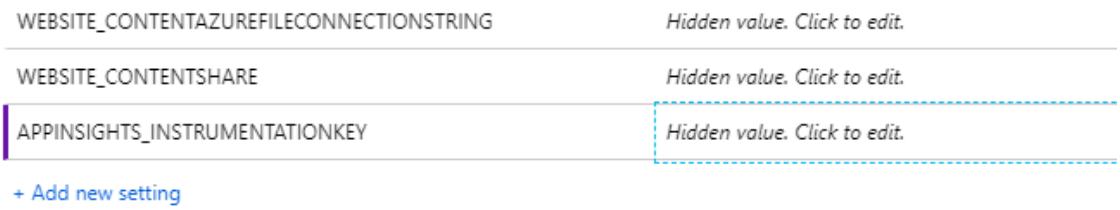
2 Implementieren von Application Insights für die Azure Function

Kopieren des Instrumentation Key der Azure Function Application Insights resource

In der Azure Function → Application Settings



Add new setting



ÜBUNG

Zu der Azure Function Application Insights resource wechseln

- Bilder hochladen und prüfen ob App Insights etwas registriert
- Eine Datei (Kein Bild!) hochladen z.B .pdf
- Die Ursache für einen Fehler in der function mithilfe von App Insights suchen (Wo steht wieso der Dateiupload fehlgeschlagen ist)

3 Implementieren von Custom Events mit dem Telemetry Client

FileUploadController.cs (Den Filenamen und die Filegröße mit einem Upload Event tracken)

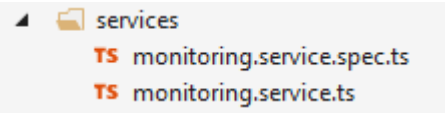
```
namespace file_upload.Controllers
{
    [Route("api/[controller]")]
    1 reference | 0 requests
    public class FileUploadController : Controller
    {
        //Appsettings Configuration
        private readonly IConfiguration _configuration;
        private TelemetryClient _telemetry;

        0 references | 0 exceptions
        public FileUploadController(IConfiguration config, TelemetryClient telemetry)
        {
            _configuration = config;
            _telemetry = telemetry;
        }

        [HttpPost]
        0 references | 0 requests | 0 exceptions
        public async Task<IActionResult> UploadFileAsync([FromForm]IFormFile file)
        {
            //Parse ConnectionString
            if (CloudStorageAccount.TryParse(_configuration.GetConnectionString("StorageAccount"), out CloudStorageAccount storageAccount))
            {
                //Create client and create BlobContainer
                var client = storageAccount.CreateCloudBlobClient();
                var container = client.GetContainerReference("originalfile");
                await container.CreateIfNotExistsAsync();
                _telemetry.TrackEvent("UploadEvent",
                 new Dictionary<string, string>()
                 {
                     { "Filename", file.FileName },
                     { "FileSize", file.Length.ToString() }
                });

                //Creates a Blob and uploads file into Blob
                var blob = container.GetBlockBlobReference(file.FileName);
                await blob.UploadFromStreamAsync(file.OpenReadStream());

                return Ok(blob.Uri);
            }

            return StatusCode(StatusCodes.Status500InternalServerError);
        }
    }
}
```

4 Implementieren von Application Insights für die UI (Custom Logging)

Environment.ts und environment.prod.ts (Key hinzufügen)

```
export const environment = {
  production: false,
  appInsights: {
    instrumentationKey: 'b03ca7b0-ab9f-45b0-aef2-ed62942e499e'
  }
};
```

Under Ordner app → neuen Ordner services erstellen und monitoring service files anlegen



Monitoring.service.spec.ts

```
import { TestBed, inject } from '@angular/core/testing';

import { MonitoringService } from './monitoring.service';

describe('MonitoringService', () => {
  beforeEach(() => {
    TestBed.configureTestingModule({
      providers: [MonitoringService]
    });
  });

  it('should be created', inject([MonitoringService], (service: MonitoringService) => {
    expect(service).toBeTruthy();
  }));
});
```

monitoring.service.ts

```
import { Injectable } from '@angular/core';
import { AppInsights } from 'applicationinsights-js';
import { environment } from '../../environments/environment';

@Injectable()
export class MonitoringService {

  private config: Microsoft.ApplicationInsights.IConfig = {
    instrumentationKey: environment.appInsights.instrumentationKey
  };


  constructor() {
    if (!AppInsights.config) {
      AppInsights.downloadAndSetup(this.config);
    }
  }

  logPageView(name?: string, url?: string, properties?: any,
    measurements?: any, duration?: number) {
    AppInsights.trackPageView(name, url, properties, measurements, duration);
  }

  logEvent(name: string, properties?: any, measurements?: any) {
    AppInsights.trackEvent(name, properties, measurements);
  }

}
```

App.module.ts

```
import { FileUploadModule } from 'primeng/fileupload';
import { MonitoringService } from './services/monitoring.service';

@NgModule({
  declarations: [
    AppComponent,
    NavMenuComponent,
    HomeComponent,
    CounterComponent,
    FetchDataComponent,
    FileUploadComponent
  ],
  imports: [
    BrowserModule.withServerTransition({ appId: 'ng-cli-universal' }),
    HttpClientModule,
    FormsModule,
    FileUploadModule,
    RouterModule.forRoot([
      { path: 'file-upload', component: FileUploadComponent },
      { path: '', component: HomeComponent, pathMatch: 'full' },
      { path: 'counter', component: CounterComponent },
      { path: 'fetch-data', component: FetchDataComponent },
    ])
  ],
  providers: [
    MonitoringService
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Counter.components.ts

```
import { Component } from '@angular/core';
import { MonitoringService } from '../services/monitoring.service';


@Component({
  selector: 'app-counter-component',
  templateUrl: './counter.component.html'
})
export class CounterComponent {
  public currentCount = 0;


  constructor(private monitoringService: MonitoringService) {
    this.monitoringService.logPageView("Page view: Counter");
  }

  public incrementCounter() {
    this.currentCount++;
  }
}
```

Fetch-data.component.ts

```
import { Component, Inject } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { MonitoringService } from '../services/monitoring.service';

@Component({
  selector: 'app-fetch-data',
  templateUrl: './fetch-data.component.html'
})
export class FetchDataComponent {
  public forecasts: WeatherForecast[];

  constructor(http: HttpClient, @Inject('BASE_URL') baseUrl: string, private monitoringService: MonitoringService ) {
    this.monitoringService.logPageView("Page view: Fetch-Data");
    http.get<WeatherForecast[]>(baseUrl + 'api/SampleData/WeatherForecasts').subscribe(result => {
      this.forecasts = result;
    }, error => console.error(error));
  }
}

interface WeatherForecast {
  dateFormatted: string;
  temperatureC: number;
  temperatureF: number;
  summary: string;
}
```

File-upload.conponents.ts

```
import { Component, OnInit } from '@angular/core';
import { MonitoringService } from '../services/monitoring.service';

@Component({
  selector: 'app-file-upload',
  templateUrl: './file-upload.component.html',
  styleUrls: ['./file-upload.component.css']
})
export class FileUploadComponent implements OnInit {

  constructor(private monitoringService: MonitoringService) {
    this.monitoringService.logPageView("Page view: File-Upload");
  }

  ngOnInit() {

  }

}
```

Ausführen

- Lokal und prüfen in App Insights was angezeigt wird
- Publishen und nochmals prüfen