# WCF

<system.diagnostics>

<trace autoflush="true" />

<sources>

<source name="System.ServiceModel.MessageLogging">

<listeners>

<add name="myListener"/>

</listeners>

</source>

</sources>

<sharedListeners>

<add name="myListener"

type="System.Diagnostics.XmlWriterTraceListener"

initializeData="ServiceModel.svclog" />

</sharedListeners>

</system.diagnostics>

<system.serviceModel>

<diagnostics>

<messageLogging logEntireMessage="true"

logMessagesAtServiceLevel="true"

logMessagesAtTransportLevel="true"

logMalformedMessages="true"

maxMessagesToLog="50000"

maxSizeOfMessageToLog="20000" >

</messageLogging>

</diagnostics>

</system.serviceModel>

# WebAPI

* create Web Application
* add NuGet package for WebAPI
* add WebApiConfig class

public static class WebApiConfig

{

    public static void Register(HttpConfiguration config)

    {

// ERST SPÄTER EINFÜGEN => NACH FIDDLER!

        config.Formatters.Clear();

        config.Formatters.Add(new JsonMediaTypeFormatter());

        config.Routes.MapHttpRoute(

            name: "Api1",

            routeTemplate: "api/DGBB/{controller}/{action}",

            defaults: new { action = "Get" });

    }

}

* add Global.asax

protected void Application\_Start(object sender, EventArgs e)

{

    GlobalConfiguration.Configure(WebApiConfig.Register);

}

* add EventsController class

public string[] Get()

{

    return new[] { "WebApi & SignalR", "Cross Platform Development", "Event 3"};

* call in Browser => XML
* call in Fiddler => JSON
  + Warum ist das so?
* setze content-type in Fiddler-Request => JSON
* configure Formatters
* zeigen in Browser und Fiddler

# SignalR

Server

* add SignalR NuGet package
* add OWIN self hosting package
* add Startup class & configure SignalR
* add hub class
* add host class & start WebApp

**In main:**

var url = "http://localhost:8082";

using (WebApp.Start(url))

{

    Console.WriteLine("\n[{0}] SignalR server listening on {1}.", DateTime.Now.ToString("dd-mm-yyyy hh:MM:ss"), url);

    Console.ReadKey();

}

Client

* add SignalR Client NuGet package
* create hub connection
* create hub proxy from connection
* start connection
* register events to listen for with Proxy.On
* call server methods with Proxy.Invoke

var connection = new HubConnection("http://localhost:8082/");

connection.TraceLevel = TraceLevels.All;

connection.TraceWriter = Console.Out;

var proxy = \_connection.CreateHubProxy("PlanningGridHub");

await \_connection.Start(); **ODER:** \_connection.Start().Wait();

proxy.Invoke("WriteMessage", "Hello!");

\_proxy.On<string, decimal[]>("AccountDataChanged", (accountId, values) =>

{

DoSomething();

});