Ktor

REST APIs mit Kotlin entwickeln

Werner Eberling

E-Mail: werner.eberling@mathema.de
Twitter: @Wer_Eb

Der Sprecher

Werner Eberling

Principal Consultant / Autor

Email: werner.eberling@mathema.de

Twitter: @Wer_Eb

Was ist Ktor?

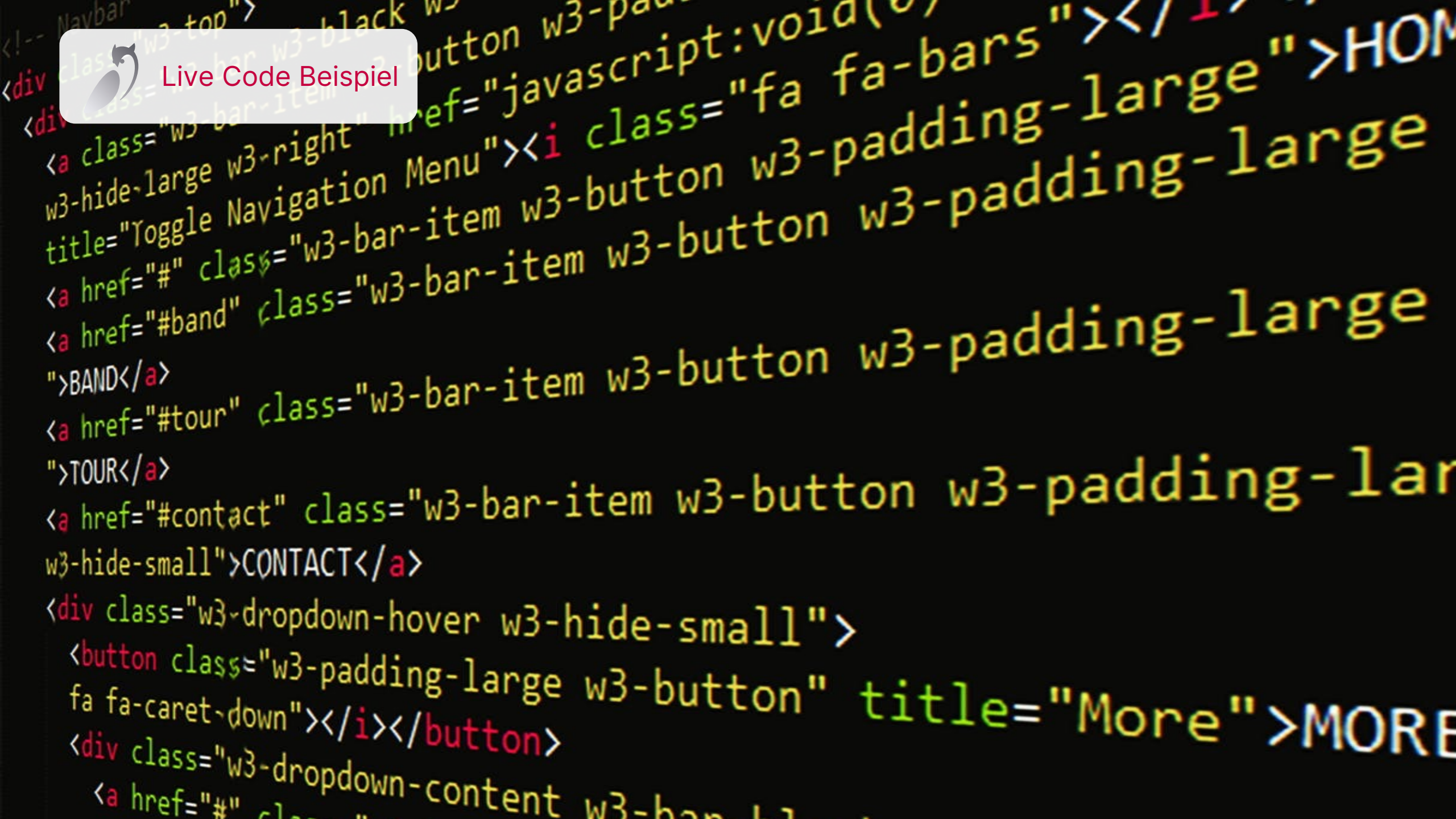
„Ktor is a framework to easily build connected applications – web applications, ***HTTP services***, mobile and browser applications. “

Ktor auf einer Folie

```
fun main() {
    embeddedServer(Netty, port = 8080, host = "0.0.0.0") {
        routing {
            route("/hello") {
                get {
                    call.respondText("Hallo ETKA!")
                }
            }
        }
    }.start(wait = true)
}
```

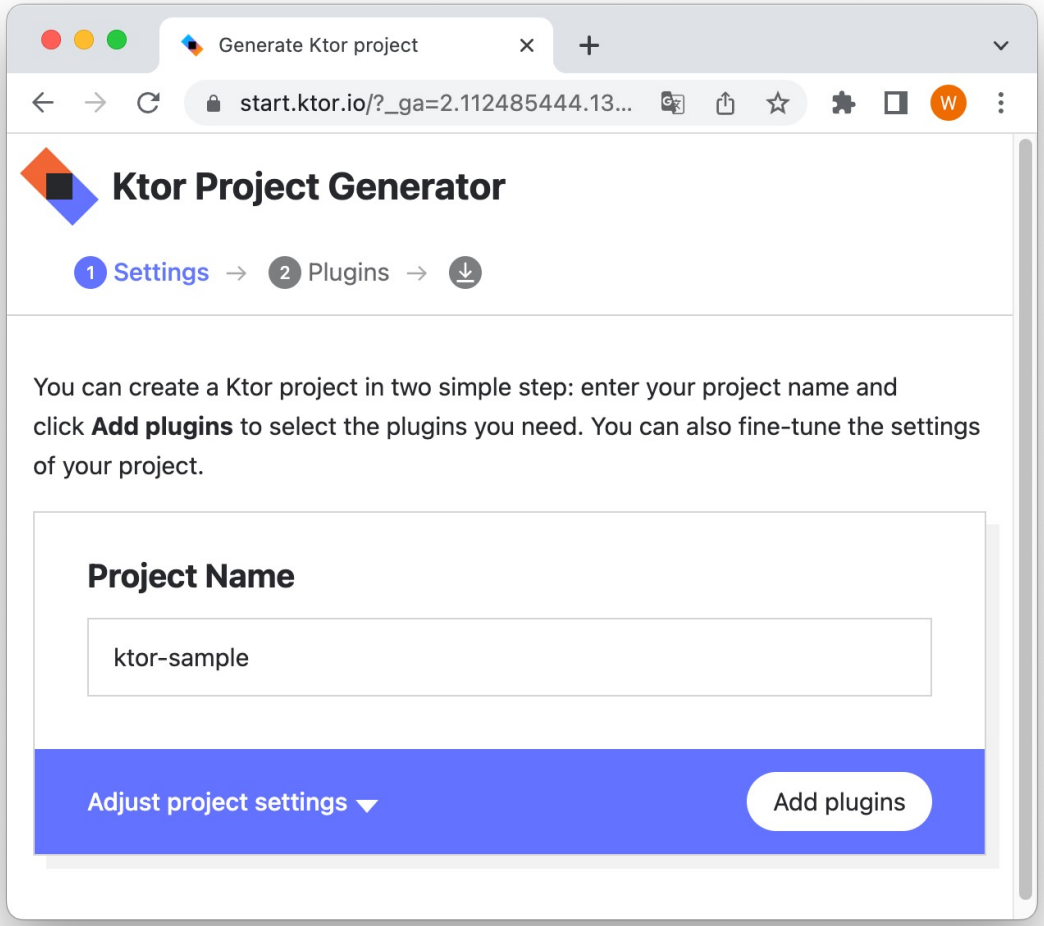
- Kotlin DSL zur Definition von HTTP Endpunkten

  Ok, eigentlich „nur“ Funktionen und Funktions-aufrufe in „normaler“ Kotlin Syntax ;)

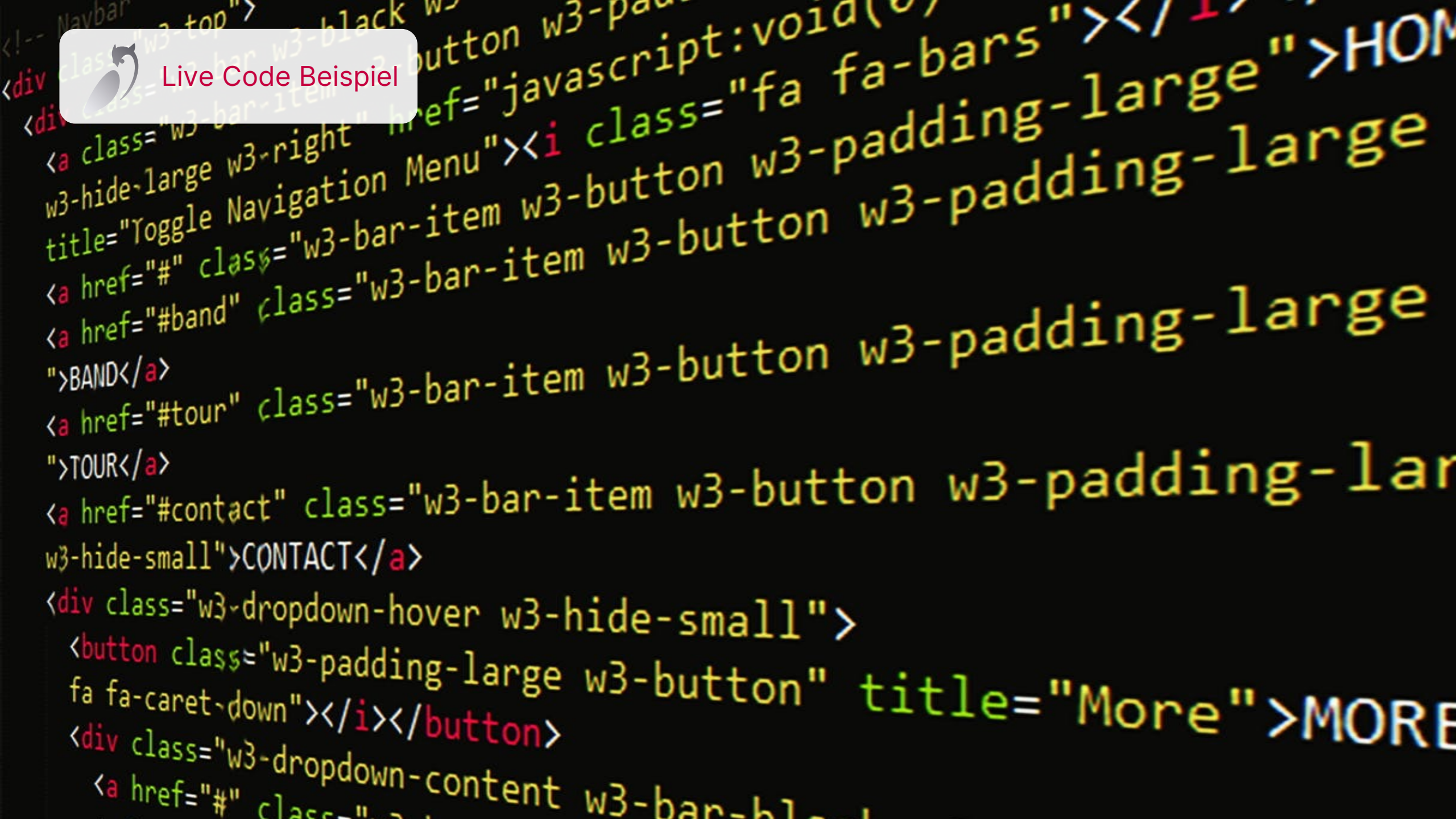

Live Code Beispiel
<!-- Navbar
<div
<a class="
w3-hide-large w3-right
title="Toggle Navigation Menu"><i class="fa fa-bars
<a href="#" class="w3-bar-item w3-button w3-padding-large
<a href="#band" class="w3-bar-item w3-button w3-padding-large
">BAND
<a href="#tour" class="w3-bar-item w3-button w3-padding-large
">TOUR
<a href="#contact" class="w3-bar-item w3-button w3-padding-lar
w3-hide-small">CONTACT

<button class="w3-padding-large w3-button" title="More">MORE
fa fa-caret-down"></i></button>
<div class="w3-dropdown-content
<a href="#"

Aufsetzen eines Ktor Projektes



- Ktor Project Generator erzeugt Projektrümpfe mit den gewünschten Plugins
  - Ähnlich Spring Boot Starter
- Alternativ:
  IntelliJ IDEA Ultimate + Ktor Plugin


Live Code Beispiel
<!-- Navbar
<div
<a class="
w3-hide-large w3-right
title="Toggle Navigation Menu"><i class="fa fa-bars
<a href="#" class="w3-bar-item w3-button w3-padding-large
<a href="#band" class="w3-bar-item w3-button w3-padding-large
">BAND
<a href="#tour" class="w3-bar-item w3-button w3-padding-large
">TOUR
<a href="#contact" class="w3-bar-item w3-button w3-padding-lar
w3-hide-small">CONTACT

<button class="w3-padding-large w3-button" title="More">MORE
fa fa-caret-down"></i></button>
<div class="w3-dropdown-content
<a href="#"

Definition der unterstützten Routen

```
fun Route.greetingRouting(){
   route("/greetings") {
      get {
         call.respond(greetingStore.map { e -> e.value })
      }
   }
}
```

```
@Serializable
data class Greeting(val type:String, val greeting: String)

val greetingStore = mutableMapOf<String, Greeting>()
```

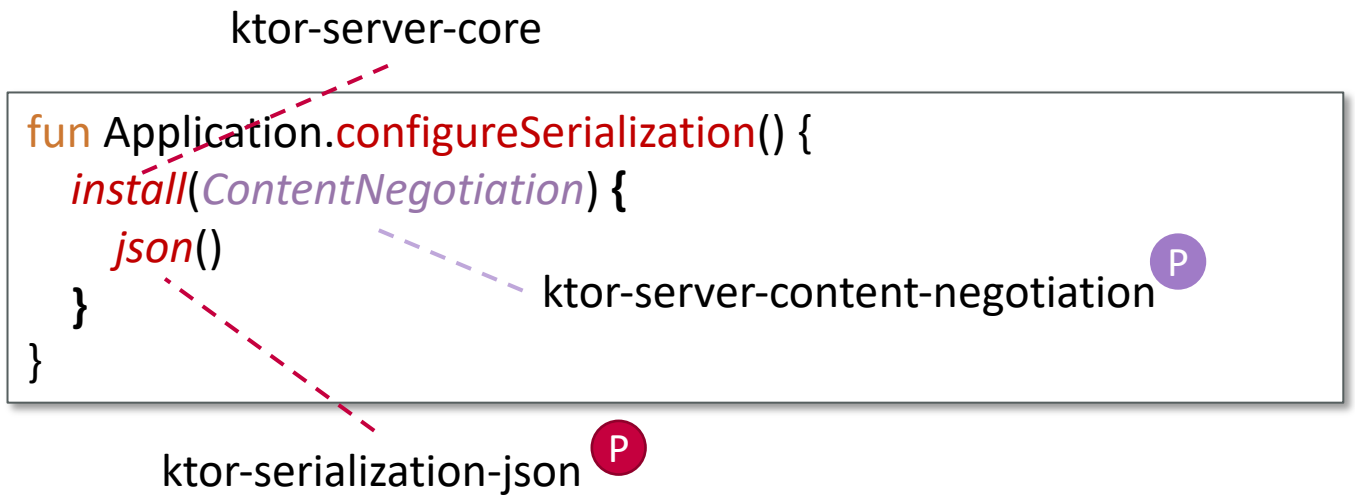
- Routendefinition über die Funktion `route`
- Nutzung von Lambda Ausdrücken für unterstütze Verben und deren hinterlegte Funktionen

Arbeiten mit Variablen

```
fun Route.greetingRouting(){
    route("/greetings") {
        // [...]
        get("{type}") {
            val type = call.parameters["type"]
            call.respond(greetingStore[type] ?:
                HttpStatusCode.NotFound
            )
    }
}
```

- Erweiterung des Pfades am HTTP Verb
  - Angabe von Pfadvariablen möglich
- Zugriff auf Pfadvariablem über `call.parameters`

Content (De-)Serialisierung



- Content Negotiation und Serialisierung werden über entsprechende Plugins P realisiert
  - Funktion muss im Rahmen der Anwendungskonfiguration aufgerufen werden

Zugriff auf den Request Body

```
fun Route.greetingRouting(){
   route("/greetings") {
      // [...]
      post {
         val greeting = call.receive<Greeting>()
         greetingStore[greeting.type] = greeting
         call.respond(HttpStatusCode.Created)
      }
   }
}
```

- Erweiterung des Pfades am HTTP Verb
  - Angabe von Pfadvariablen möglich
- Zugriff auf Pfadvariablem über `call.parameters`

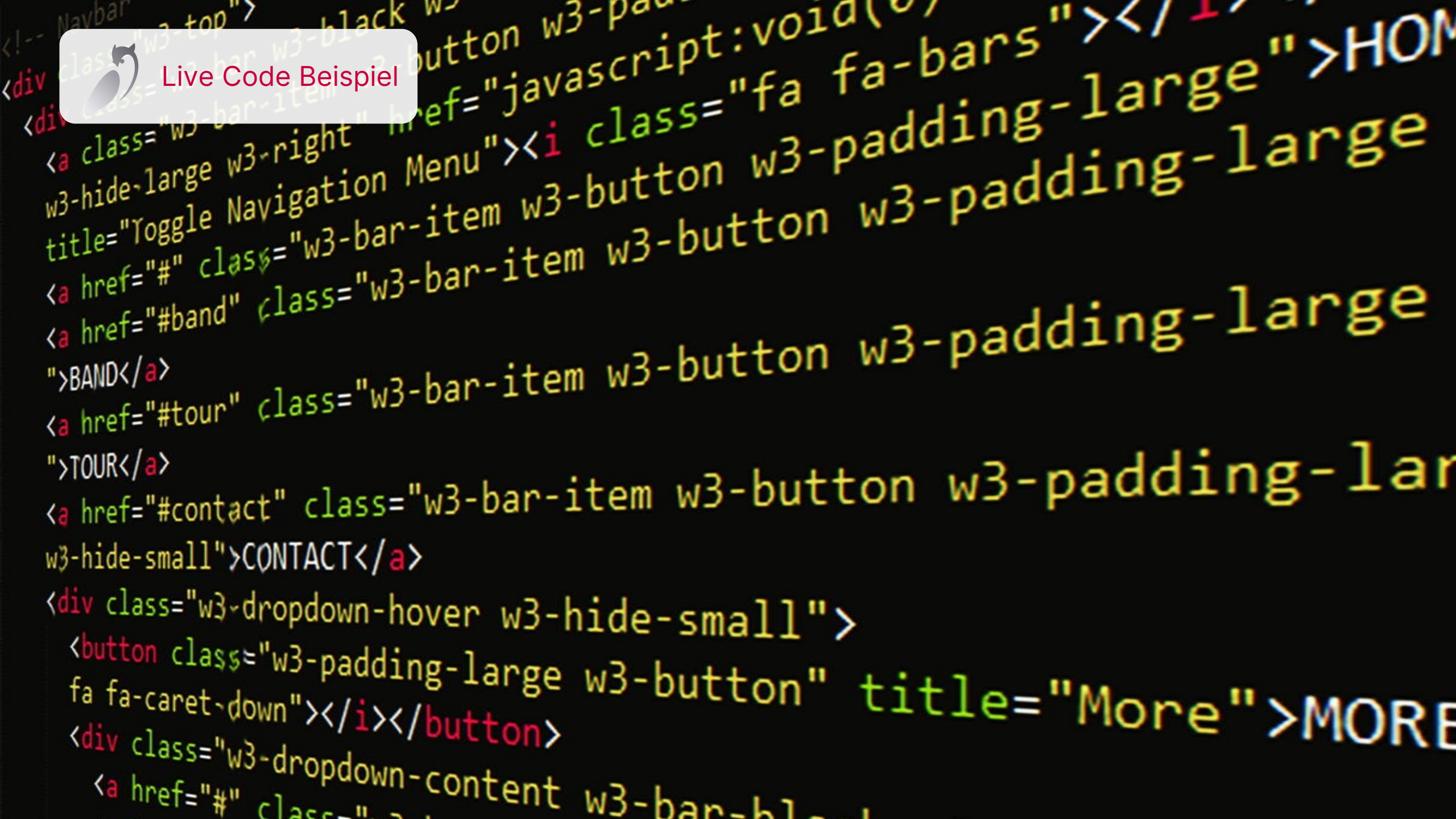
Start des embedded Servers

```
fun main() {
    embeddedServer(Netty, port = 8080, host = "0.0.0.0") {
        configureRouting()
        configureSerialization()
    }.start(wait = true)
}
```

```
fun Application.configureRouting() {
    routing {
        greetingRouting()
    }
}
```

```
fun Application.configureSerialization() {
    install(ContentNegotiation) {
        json()
    }
}
```

- Programmatischer Start und Konfiguration des Servers
  - Verschiedene Laufzeitumgebungen möglich
- Konfiguration der benötigten Plugins via Funktionsaufruf
  - Implementierung als Extension Functions

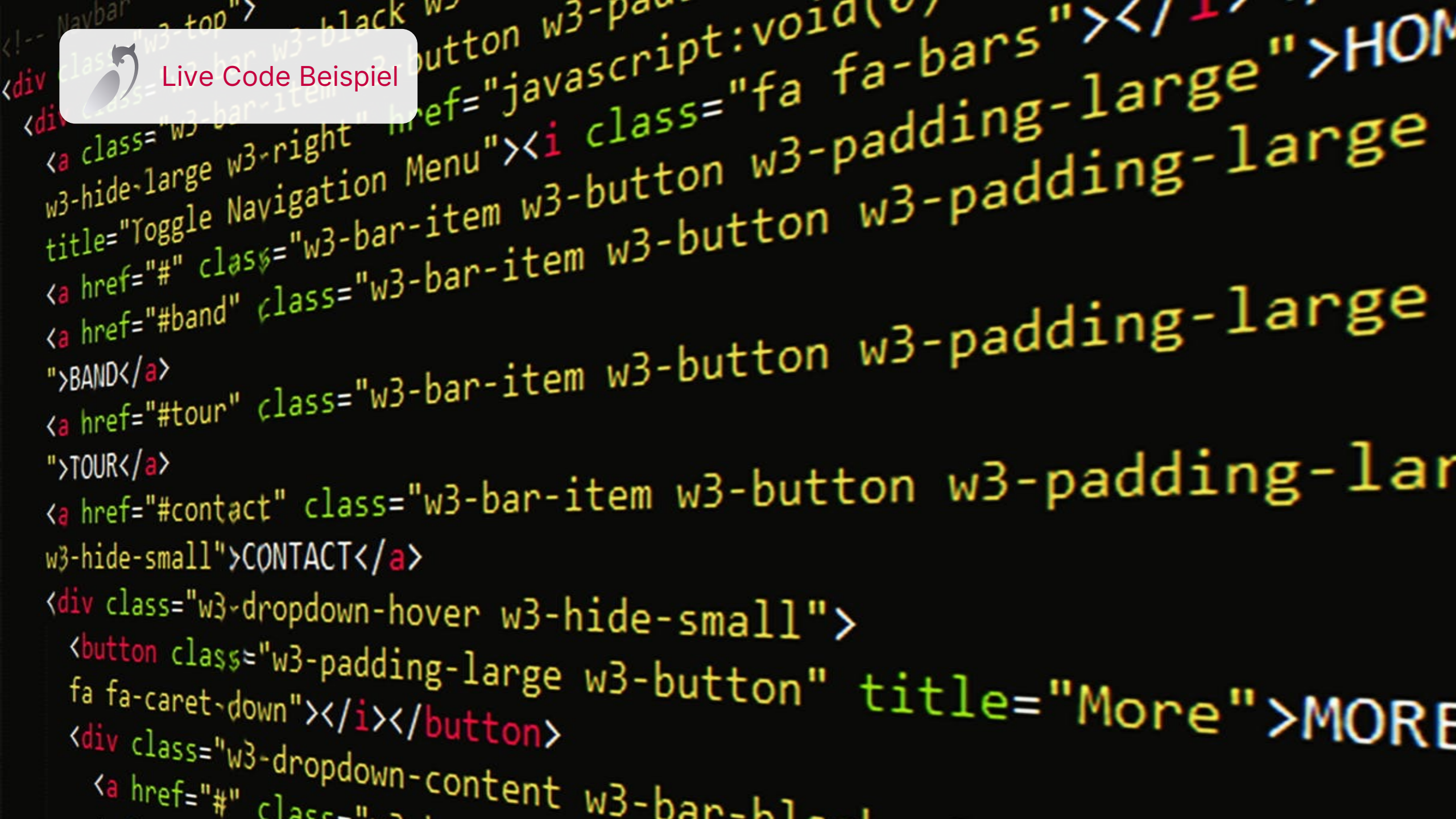

Live Code Beispiel
<!-- Navbar
<div
<a class="
w3-hide-large w3-right
title="Toggle Navigation Menu"><i class="fa fa-bars
<a href="#" class="w3-bar-item w3-button w3-padding-large
<a href="#band" class="w3-bar-item w3-button w3-padding-large
">BAND
<a href="#tour" class="w3-bar-item w3-button w3-padding-large
">TOUR
<a href="#contact" class="w3-bar-item w3-button w3-padding-lar
w3-hide-small">CONTACT

<button class="w3-padding-large w3-button" title="More">MORE
fa fa-caret-down"></i></button>
<div class="w3-dropdown-content
<a href="#"

Testen der Anwendung

```
class ApplicationTest {
    @Test
    fun testStartWithEmptyGreetingStorage() = testApplication {
        application {
            configureRouting()
            configureSerialization()
        }
        client.get("/greetings").apply {
            assertEquals(HttpStatusCode.OK, status)
            assertEquals("[]", bodyAsText())
        }
    }
}
```

- Eigene Test Applikation
  - ohne Serverinstanz
  - ohne Netzwerkanbindung
  - mit eigener Konfiguration
- Ktor HTTP Client zum
  - Absetzen von Anfragen
  - Verarbeiten der Antwort


Live Code Beispiel
<!-- Navbar
<div
<a class="
w3-hide-large w3-right
title="Toggle Navigation Menu"><i class="fa fa-bars
<a href="#" class="w3-bar-item w3-button w3-padding-large
<a href="#band" class="w3-bar-item w3-button w3-padding-large
">BAND
<a href="#tour" class="w3-bar-item w3-button w3-padding-large
">TOUR
<a href="#contact" class="w3-bar-item w3-button w3-padding-lar
w3-hide-small">CONTACT

<button class="w3-padding-large w3-button" title="More">MORE
fa fa-caret-down"></i></button>
<div class="w3-dropdown-content
<a href="#"

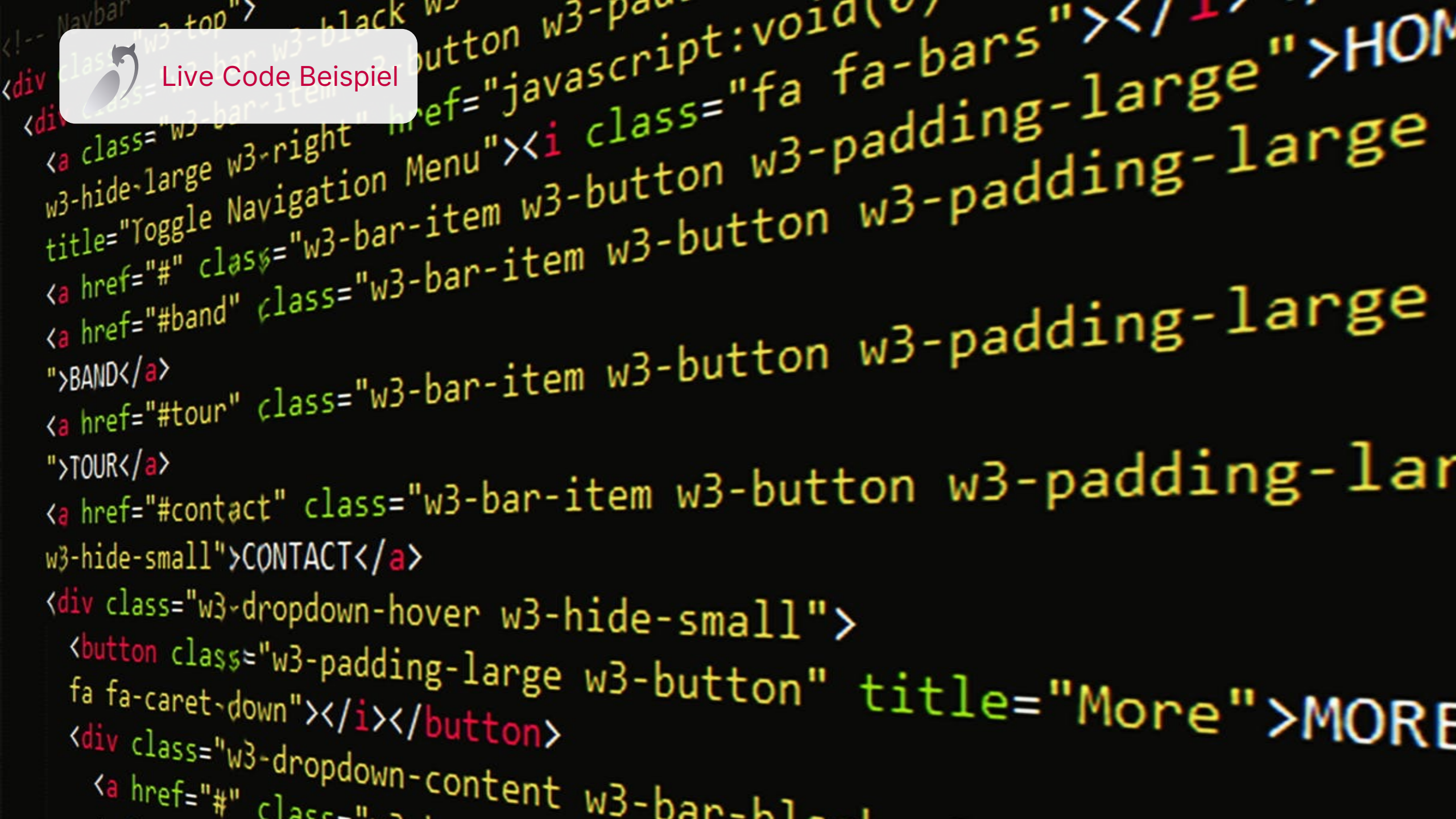
Konfiguration via application.conf

```
ktor {
    deployment {
        port = 8080
    }
    application {
        modules = [ de.mathema.ApplicationKt.module ]
    }
}
```

```
fun main (args: Array<String>) =
                        io.ktor.server.netty.EngineMain.main(args)

fun Application.module() {
    configureRouting()
    configureSerialization()
}
```

- Anwendung zieht sich ihre Konfiguration automatisch
  - Mögliche Auswirkungen auf Tests!
- Konfiguration wird über Module geladen
  - Wirkung über Modulgrenzen hinweg!


Live Code Beispiel
<!-- Navbar
<div
<a class="
w3-hide-large w3-right
title="Toggle Navigation Menu"><i class="fa fa-bars
<a href="#" class="w3-bar-item w3-button w3-padding-large
<a href="#band" class="w3-bar-item w3-button w3-padding-large
">BAND
<a href="#tour" class="w3-bar-item w3-button w3-padding-large
">TOUR
<a href="#contact" class="w3-bar-item w3-button w3-padding-lar
w3-hide-small">CONTACT

<button class="w3-padding-large w3-button" title="More">MORE
fa fa-caret-down"></i></button>
<div class="w3-dropdown-content
<a href="#"

Einfach mal ausprobieren ;)

- Keine Annotationswüste
  - (Fast) alles ist eine Funktion
- Leichtgewichtige Testmöglichkeiten
- Strukturierte Nebenläufigkeit durch Koroutinen
  - Bodys der Verb-Funktionen sind Suspending Functions
- Leichtgewichtige Alternative zu Spring MVC
  - Falls keine weiteren Spring Features benötigt werden!
  - Keine magische „Dependency triggered“ Funktionalität ;)
  - Bei Bedarf auch WAR Deployment möglich

Vielen Dank! Fragen?

Werner Eberling

E-Mail: werner.eberling@mathema.de
Twitter: @Wer_Eb

Beispiele: https://github.com/wern/ktor-sample

Beispiele?
Scan me ;)

www.mathema.de