CREATING WEB APPLICATIONS IN KOTLIN

THE ~~YEAST~~ GIST OF THIS TALK

(MORE ON THAT LATER...)

- Brief introduction to Ktor
- Creation of a Ktor project
- Creation of endpoints (CRUD-ish)
- Pros and cons

WHAT IS KTOR?

- A framework for building asynchronous servers and clients using Kotlin.
- It provides a Domain Specific Language (DSL) for server-side development.
- Can be used on web, iOS and Android.

  (used by JetBrains as the backend for their KotlinConf mobile and web app)

CREATING A KTOR WEB APPLICATION

Tools and requirements:

- IntelliJ IDEA
- Minimum version of Kotlin 1.3
- Application engine to handle requests and responses between client and server → `Netty`

STRUCTURE OF A KTOR WEB APPLICATION

FEATURES → in charge of providing functionality into the request/response cycle ==> Gradle dependencies

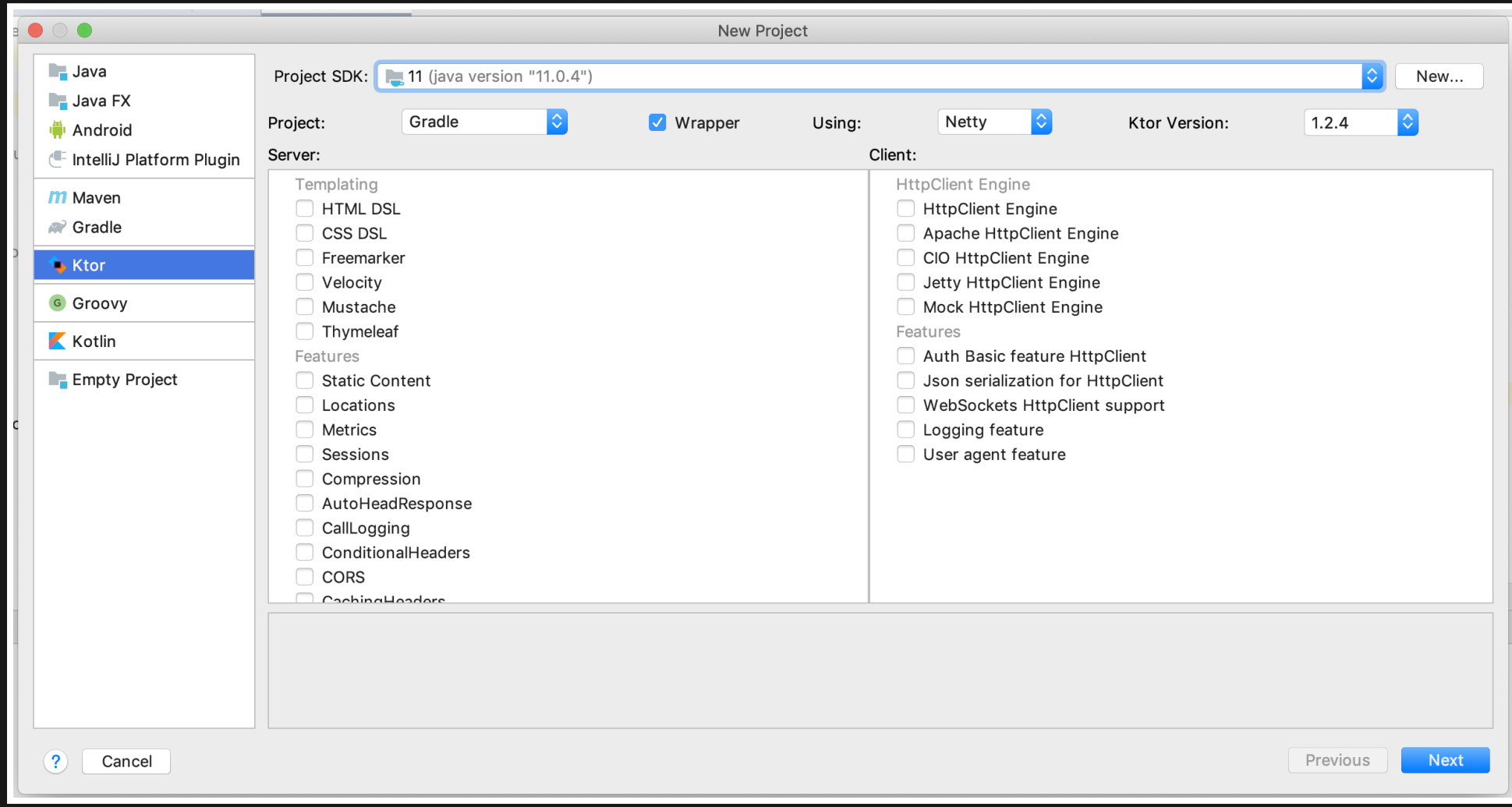
Examples:

Headers for network calls, processing JSON, authentication, routing...

(we could also create our own)

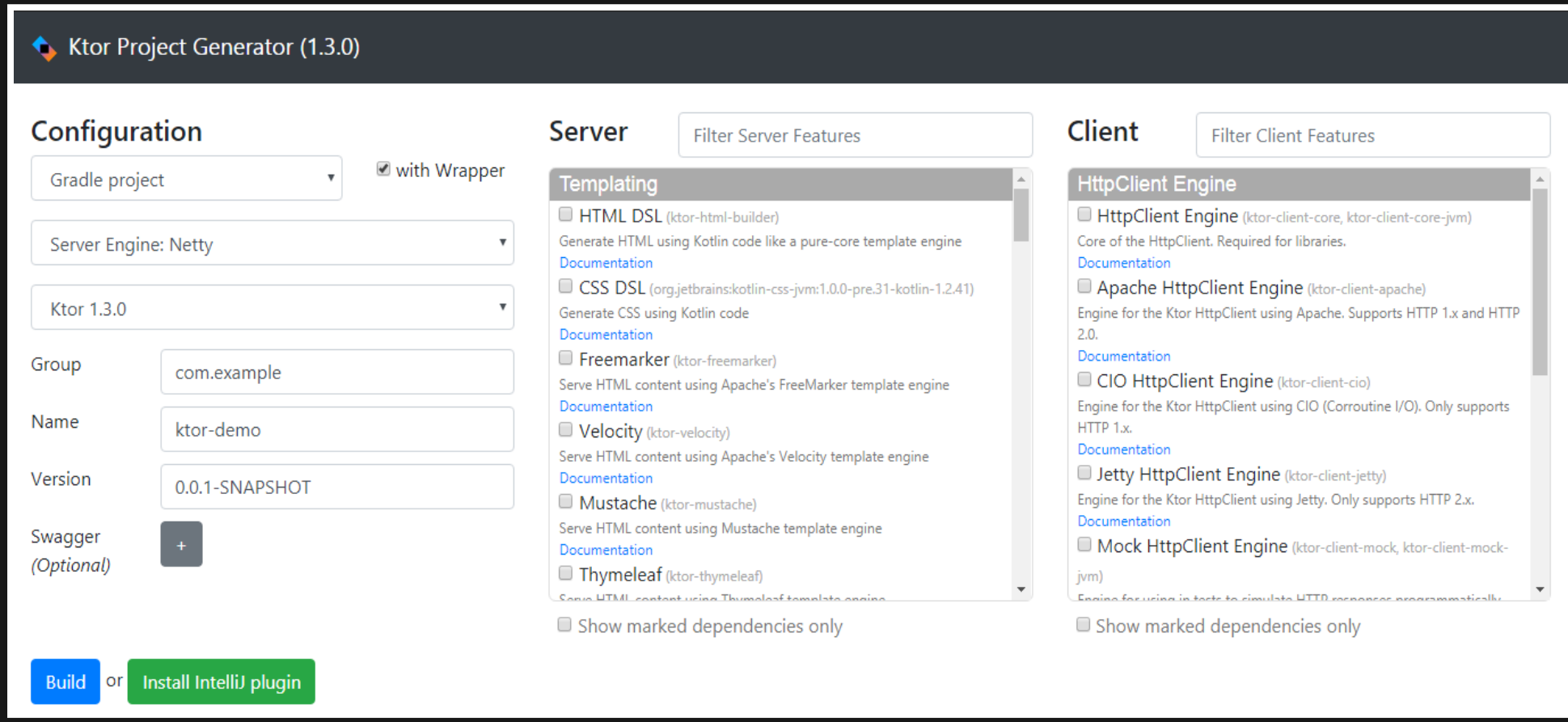
GENERATE A KTOR PROJECT

IntelliJ IDEA plugin



GENERATING A KTOR PROJECT (2ND OPTION)

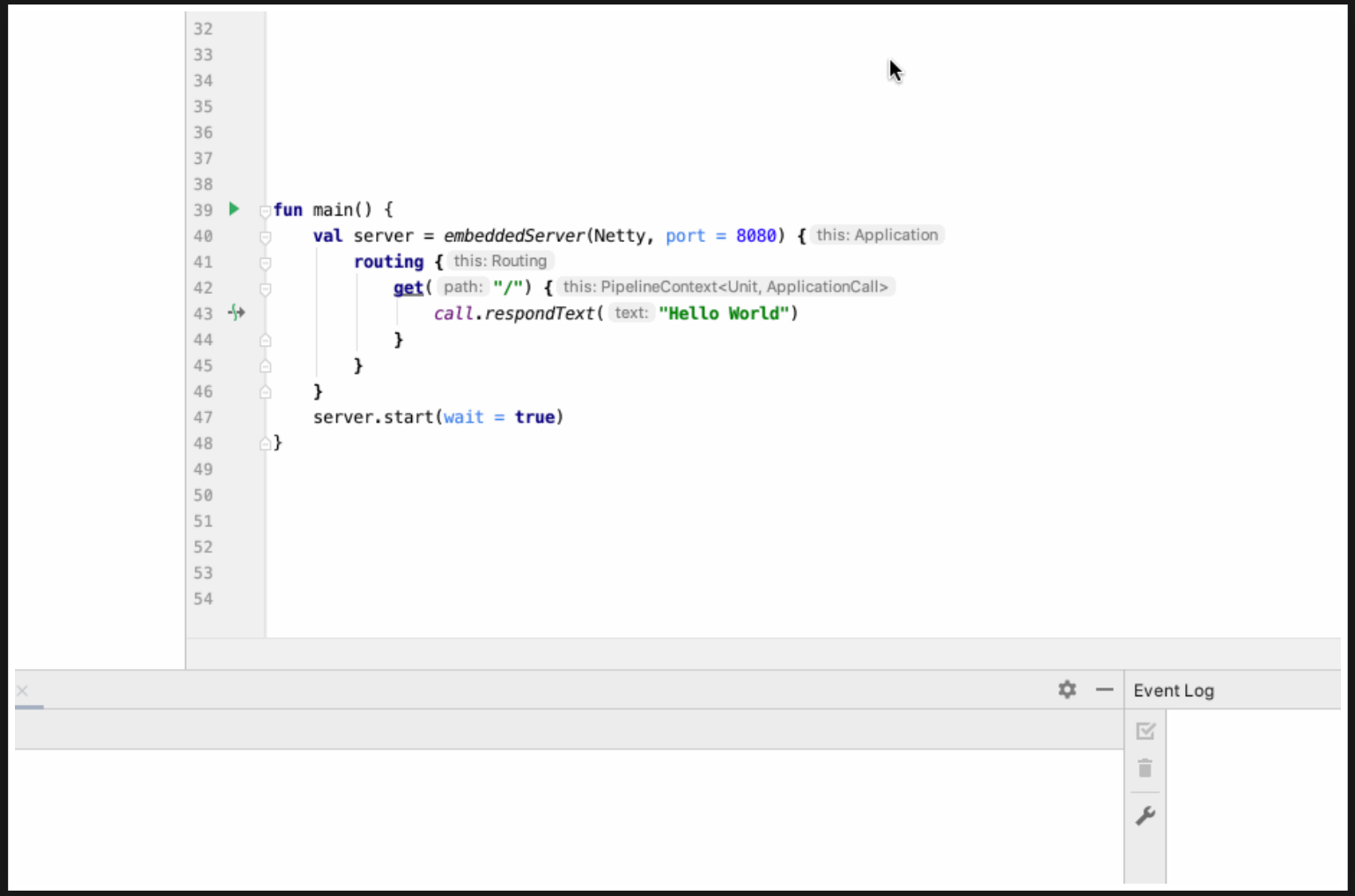
Site start.ktor.io



HELLO WORLD

```
1  import io.ktor.application.*
2  import io.ktor.http.*
3  import io.ktor.response.*
4  import io.ktor.routing.*
5  import io.ktor.server.engine.*
6  import io.ktor.server.netty.*
7
8  fun main(args: Array<String>) {
9      val server = embeddedServer(Netty, port = 8080) { this: Application
10
11         routing { this: Routing
12             get( path: "/"){ //DSL function
13                 call.respondText( text: "Welcome home (sanitarium)", ContentType.Text.Plain)
14             }
15         }
16     }
17     server.start(wait = true)
18 }
19
```

HOW TO RUN


fun main() {
val server = embeddedServer(Netty, port = 8080) { this: Application
routing { this: Routing
get(path: "/") { this: PipelineContext<Unit, ApplicationCall>
call.respondText(text: "Hello World")
}
}
}
server.start(wait = true)
}
Event Log

COMMON FEATURES

- DefaultHeaders: adds a default set of headers to every HTTP response.
- StatusPages: allows Ktor applications to respond appropriately to any failure state.
- ContentNegotiation: tells the server what to use for incoming and outbound requests. A.k.a. serialization/deserialization Kotlin objects <-> JSON.

DefaultHeaders

```
30
31  fun main() {
32      val server = embeddedServer(Netty, port = 8080) { this: Application
33
34          install(DefaultHeaders) { this: DefaultHeaders.Configuration
35              header( name: "Ktor-Developer",  value: "Alicia P") // will send this header with each response
36          }
37
38          routing { this: Routing
39              get( path: "/") { this: PipelineContext<Unit, ApplicationCall>
40                  call.respondText( text: "Hello World")
41              }
42          }
43      }
44      server.start(wait = true)
45  }
46
```

StatusPages

```
29
30  fun main() {
31      val server = embeddedServer(Netty, port = 8080) { this: Application
32
33          install(StatusPages) { this: StatusPages.Configuration
34              exception<Throwable> { e ->
35                  call.respond(HttpStatusCode.InternalServerError)
36                  // to log the exception we need to throw
37                  throw InternalServerErrorException("There was an error")
38              }
39          }
40
41          routing { this: Routing
42              get( path: "/") { this: PipelineContext<Unit, ApplicationCall>
43                  call.respondText( text: "Hello World")
44              }
45          }
46      }
47      server.start(wait = true)
48  }
49
50  class InternalServerErrorException(message: String): Throwable(message)
51
```

```
"/Applications/IntelliJ IDEA CE.app/Contents/jbr/Contents/Home/bin/java" ...
2019-10-27 16:35:36.805 [main] INFO  ktor.application - No ktor.deployment.watch patterns specified, automatic reload is not active
2019-10-27 16:35:36.850 [main] INFO  ktor.application - Responding at http://0.0.0.0:8080
2019-10-27 16:35:44.753 [nioEventLoopGroup-4-1] ERROR ktor.application - 500 Internal Server Error: GET - /
com.example.InternalServerErrorException: There was an error
    at com.example.ApplicationKt$main$server$1$2$1.invokeSuspend(Application.kt:46)
    at com.example.ApplicationKt$main$server$1$2$1.invoke(Application.kt)
    at io.ktor.util.pipeline.SuspendFunctionGun.loop(PipelineContext.kt:268)
    at io.ktor.util.pipeline.SuspendFunctionGun.access$loop(PipelineContext.kt:67)
```

ContentNegotiation

```
40
41  fun main() {
42      val server = embeddedServer(Netty, port = 8080) { this: Application
43
44          install(ContentNegotiation) { this: ContentNegotiation.Configuration
45              gson { this: GsonBuilder
46                  // Configure Gson here
47                  setPrettyPrinting()
48              }
49          }
50
51          routing { this: Routing
52              get( path: "/") { this: PipelineContext<Unit, ApplicationCall>
53                  call.respondText( text: "Hello World")
54              }
55          }
56      }
57      server.start(wait = true)
58  }
59
60
```

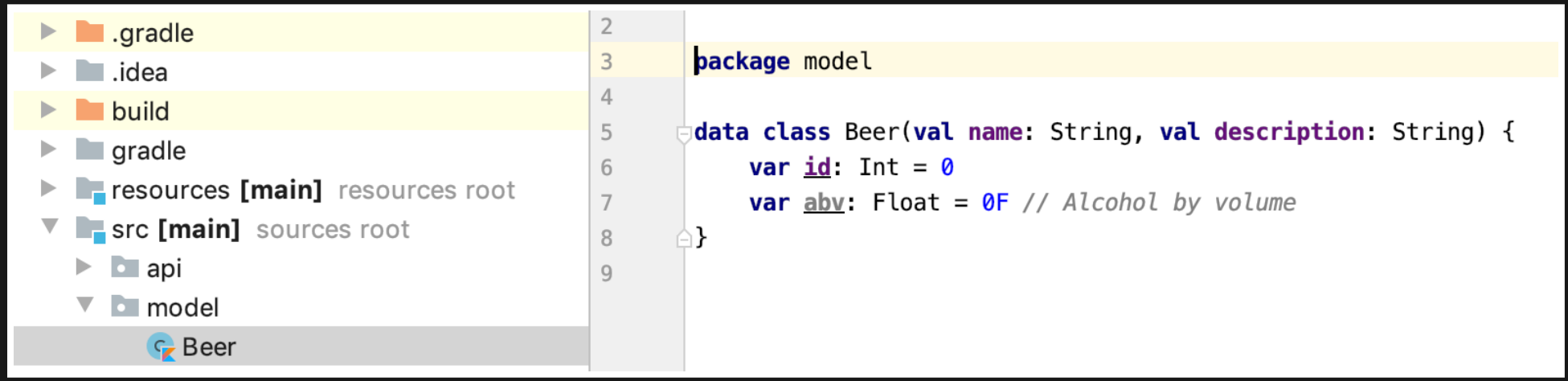
NOW WE'RE READY TO CREATE AN *IPA* USING KTOR

(maybe a typo, but it really is a beer-related API)

GitHub repository

https://github.com/aliciaphes/Kbeer

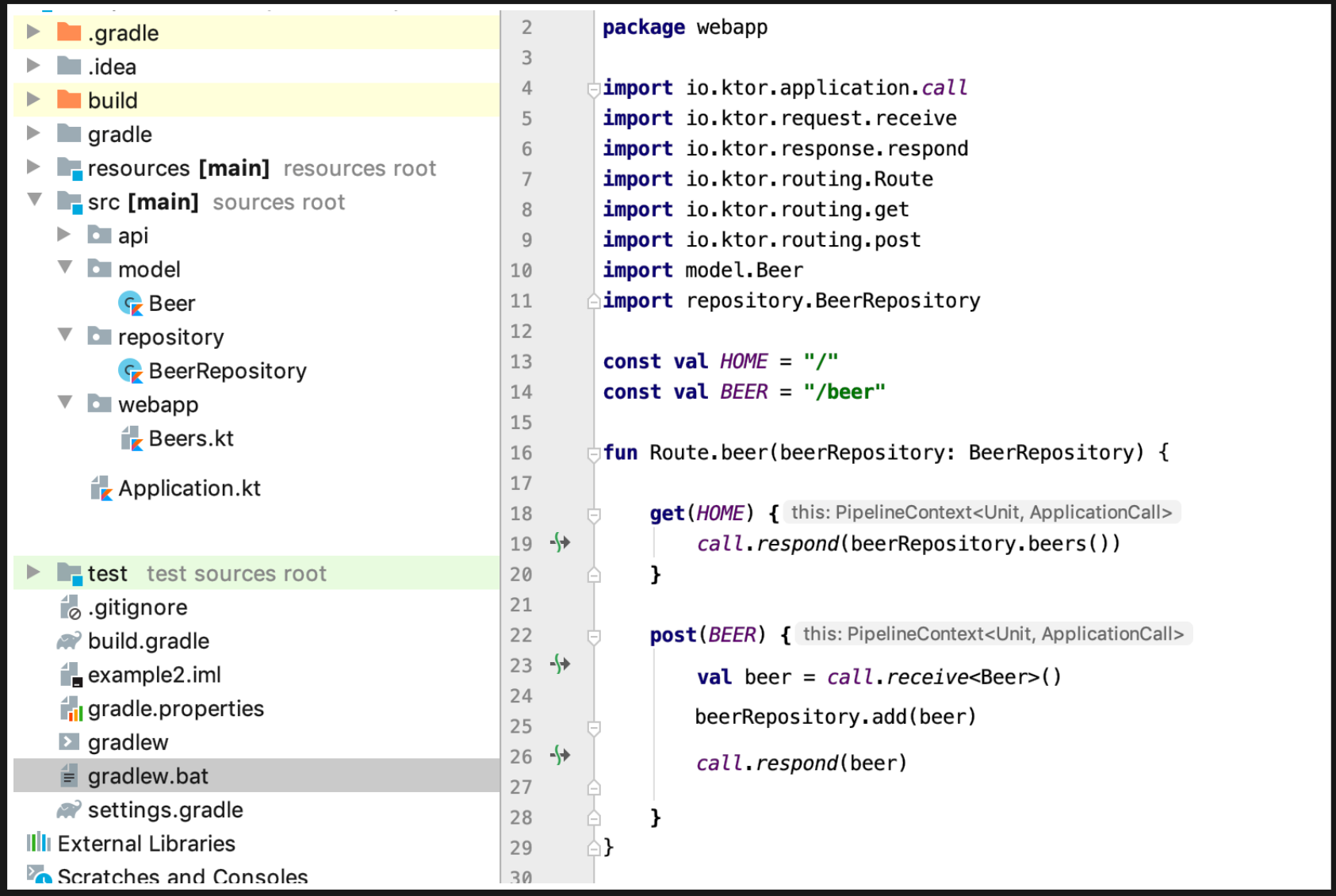
`Beer` model


.gradle
.idea
build
gradle
resources [main] resources root
src [main] sources root
api
model
Beer
package model
data class Beer(val name: String, val description: String) {
var id: Int = 0
var abv: Float = 0F // Alcohol by volume
}

In-memory repository `BeerRepository`

```kotlin
package com.kbeer.repository

import com.kbeer.model.Beer
import java.util.concurrent.atomic.AtomicInteger


class BeerRepository {

    private val idCounter: AtomicInteger = AtomicInteger()
    private val beers = ArrayList<Beer>()

    fun add(beer: Beer){
        if (!beers.contains(beer)) {
            beer.id = idCounter.incrementAndGet()
            beers.add(beer)
        }
    }

    fun beer(id: Int) = beers.find { it.id == id } ?: throw IllegalArgumentException("No beer found for $id.")

    fun beers() = beers

    fun remove(beer: Beer) {
        if (!beers.contains(beer)) {
            throw IllegalArgumentException("No beer found for ${beer.id}")
        }
        beers.remove(beer)
    }

    fun remove(id: Int) = beers.remove(beer(id))
```

Beer Route (1/2)



Beer Route (2/2)

Application.kt

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

    install(ContentNegotiation) { this: ContentNegotiation.Configuration
        gson { this: GsonBuilder
            setPrettyPrinting()
        }
    }

    val beerRepository = BeerRepository()

    routing { this: Routing

        beer(beerRepository)

    }
}
```

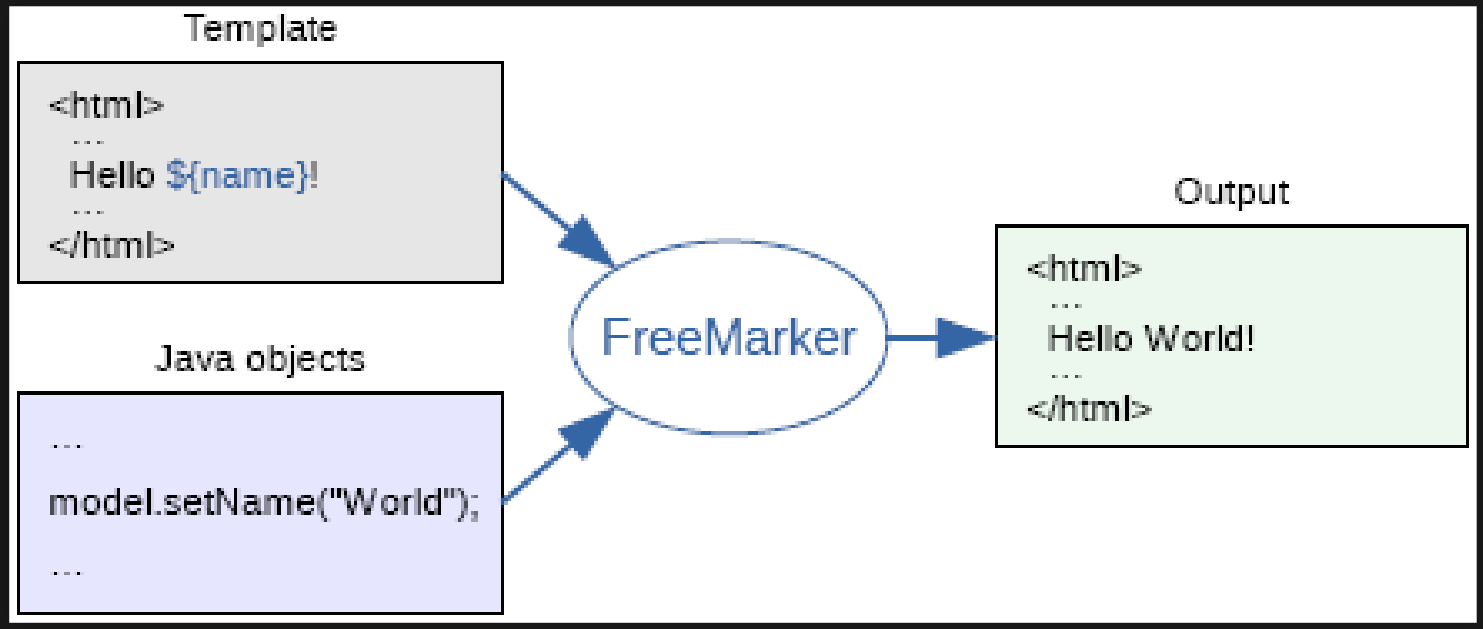
Adding a beer:

```
curl \
--request POST \
--header "Content-Type:
application/json" \
--data '{"name" : "Franziskaner",
"description": "German Hefeweizen",
"abv": "5.0"}' \
http://127.0.0.1:8080/beer
```

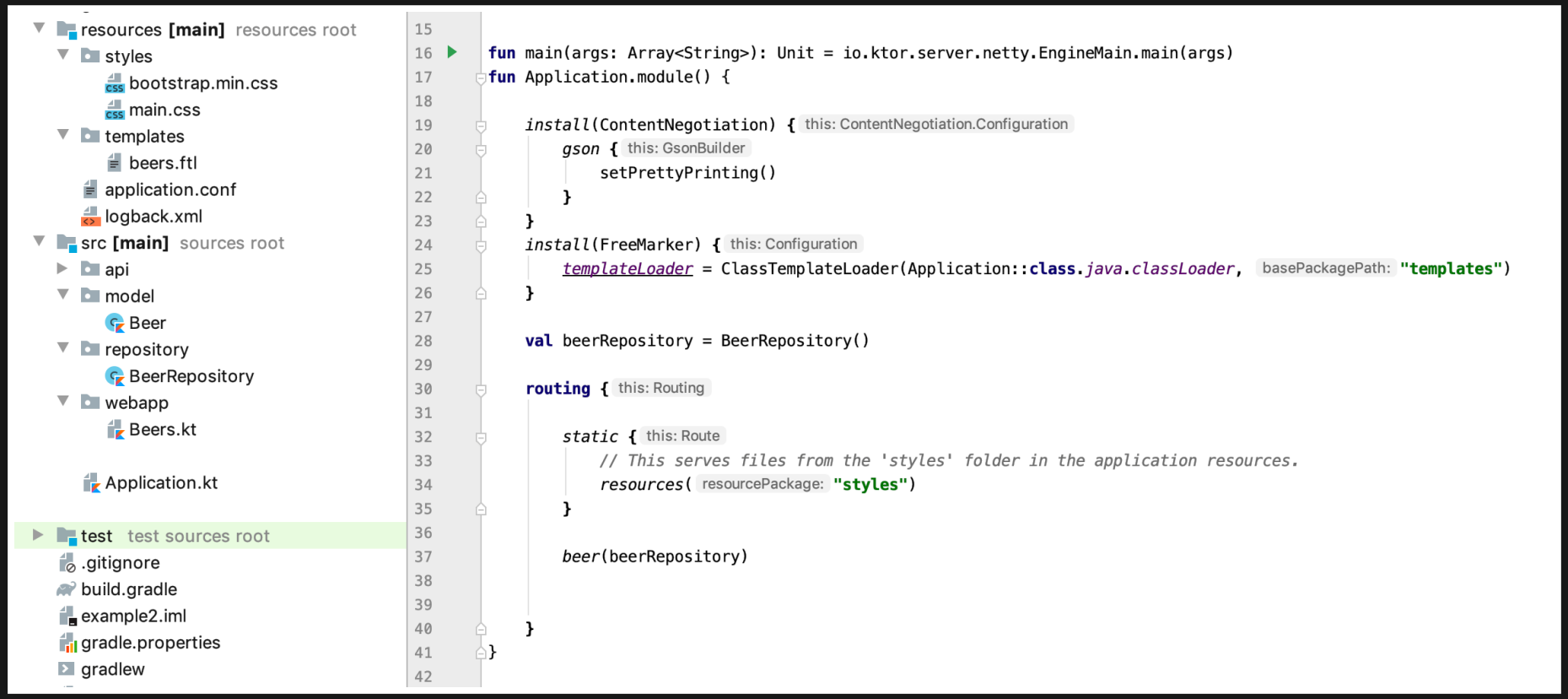
Adding a beer (in action)

It works, I guess...

Enter the template engine FreeMarker



Install feature and indicate the paths for FreeMarker



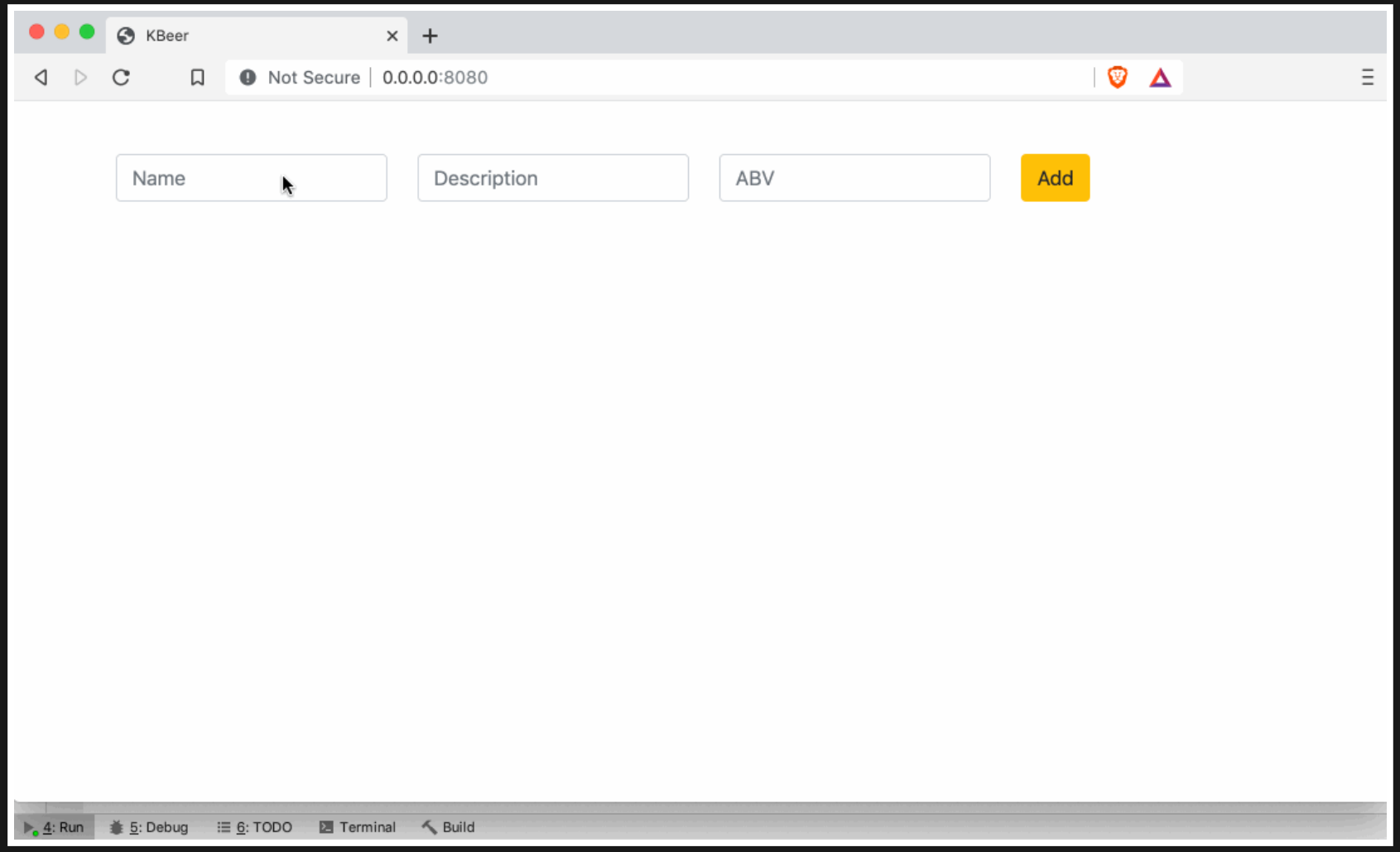
Template file `beers.ftl`

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>KBeer</title>
        <link rel="stylesheet" type="text/css" href="bootstrap.min.css">
        <link rel="stylesheet" type="text/css" href="main.css">
    </head>
    <body>
        <div class="container with-margin-top">
            <div class="row with-margin-top">
                <form action="/beer" method="post">
                    <div class="col cell"><input type="text" name="name" class="form-control" placeholder="Name"/></div>
                    <div class="col cell"><input type="text" name="description" class="form-control" placeholder="Description"/></div>
                    <div class="col cell"><input type="text" name="abv" class="form-control" placeholder="ABV"/></div>
                    <div class="col cell"><input type="hidden" name="action" value="add"></div>
                    <div class="col cell"><button type="submit" class="btn btn-warning">Add</button></div>
                </form>
            </div>
        </div>
        <#if beers?has_content>
            <div class="container with-margin-top with-margin-bottom">
                <div class="row header">
                    <div class="col-3 cell"><b>Name</b></div>
                    <div class="col-7 cell"><b>Description</b></div>
                    <div class="col-2 cell"><b>ABV</b></div>
                </div>
                <#list beers as beer>
                  <div class="row with-border-bottom">
                    <div class="col-3 cell">${beer.name}</div>
                    <div class="col-7 cell">${beer.description}</div>
                    <div class="col-2 cell">${beer.abv?string(",##0.0")}</div>
                  </div>
                </#list>
            </div>
        </#if>
    </body>
</html>
```

Adding the logic to add a beer - modify existing code

```
16  const val HOME = "/"
17  const val BEER = "/beer"
18
19
20  fun Route.beer(beerRepository: BeerRepository) {
21
22      get(HOME) { this: PipelineContext<Unit, ApplicationCall>
23          val beers = beerRepository.beers()
24          call.respond(FreeMarkerContent( template: "beers.ftl", mapOf("beers" to beers)))
25      }
26
27
28      post(BEER) { this: PipelineContext<Unit, ApplicationCall>
29          val parameters = call.receiveParameters()
30          val action = parameters["action"] ?: throw IllegalArgumentException("Missing parameter: action")
31          when (action) {
32              "add" -> {
33                  val name = parameters["name"] ?: throw IllegalArgumentException("Missing parameter: name")
34                  val description =
35                      parameters["description"] ?: throw IllegalArgumentException("Missing parameter: description")
36                  val abv =
37                      parameters["abv"] ?: throw IllegalArgumentException("Missing parameter: ABV")
38                  val beerToAdd = Beer(name, description, abv.toFloat())
39                  beerRepository.add(beerToAdd)
40              }
41          }
42          call.respondRedirect(HOME)
43      }
44  }
```

And now we run it...



Deleting a beer (template)

```
<!DOCTYPE html>
<html>
    <head>⋯
    </head>
    <body>
        <div class="container with-margin-top">
            <div class="row with-margin-top">
                <form action="/beer" method="post">
                    <div class="col cell"><input type="text" name="name" class="form-control" placeholder="Name"/></div>
                    <div class="col cell"><input type="text" name="description" class="form-control" placeholder="Description"/></div>
                    <div class="col cell"><input type="text" name="abv" class="form-control" placeholder="ABV"/></div>
                    <div class="col cell"><input type="hidden" name="action" value="add"></div>
                    <div class="col cell"><button type="submit" class="btn btn-warning">Add</button></div>
                </form>
            </div>
        </div>
        <#if beers?has_content>
            <div class="container with-margin-top with-margin-bottom">
                <div class="row header">
                    <div class="col-2 cell"><b>Name</b></div>
                    <div class="col-7 cell"><b>Description</b></div>
                    <div class="col-1 cell"><b>ABV</b></div>
                    <div class="col-2 cell"> </div>
                </div>
                <#list beers as beer>
                  <div class="row with-border-bottom">
                    <div class="col-2 cell">${beer.name}</div>
                    <div class="col-7 cell">${beer.description}</div>
                    <div class="col-1 cell">${beer.abv?string(",##0.0")}</div>
                    <div class="col-2 cell">
                        <form action="/beer" method="post">
                            <button type="submit" class="btn btn-danger">Delete</button>
                            <input type="hidden" name="action" value="delete">
                            <input class="btn btn-danger" type="hidden" name="id" value="${beer.id}" />
                        </form>
                    </div>
                  </div>
                </#list>
            </div>
        </#if>
    </body>
</html>
```

Deleting a beer (code)

```
27
28      post(BEER) {  this: PipelineContext<Unit, ApplicationCall>
29          val parameters = call.receiveParameters()
30          val action = parameters["action"] ?: throw IllegalArgumentException("Missing parameter: action")
31          when (action) {
32              "add" -> {
33                  val name = parameters["name"] ?: throw IllegalArgumentException("Missing parameter: name")
34                  val description =
35                      parameters["description"] ?: throw IllegalArgumentException("Missing parameter: description")
36                  val abv =
37                      parameters["abv"] ?: throw IllegalArgumentException("Missing parameter: ABV")
38                  val beerToAdd = Beer(name, description, abv.toFloat())
39                  beerRepository.add(beerToAdd)
40              }
41              "delete" -> {
42                  val id = parameters["id"] ?: throw IllegalArgumentException("Missing parameter: id")
43                  beerRepository.remove(id.toInt())
44              }
45          }
46          call.respondRedirect(HOME)
47      }
48  }
49
```

Deleting a beer (in action)

Adding a random beer

PUNK API

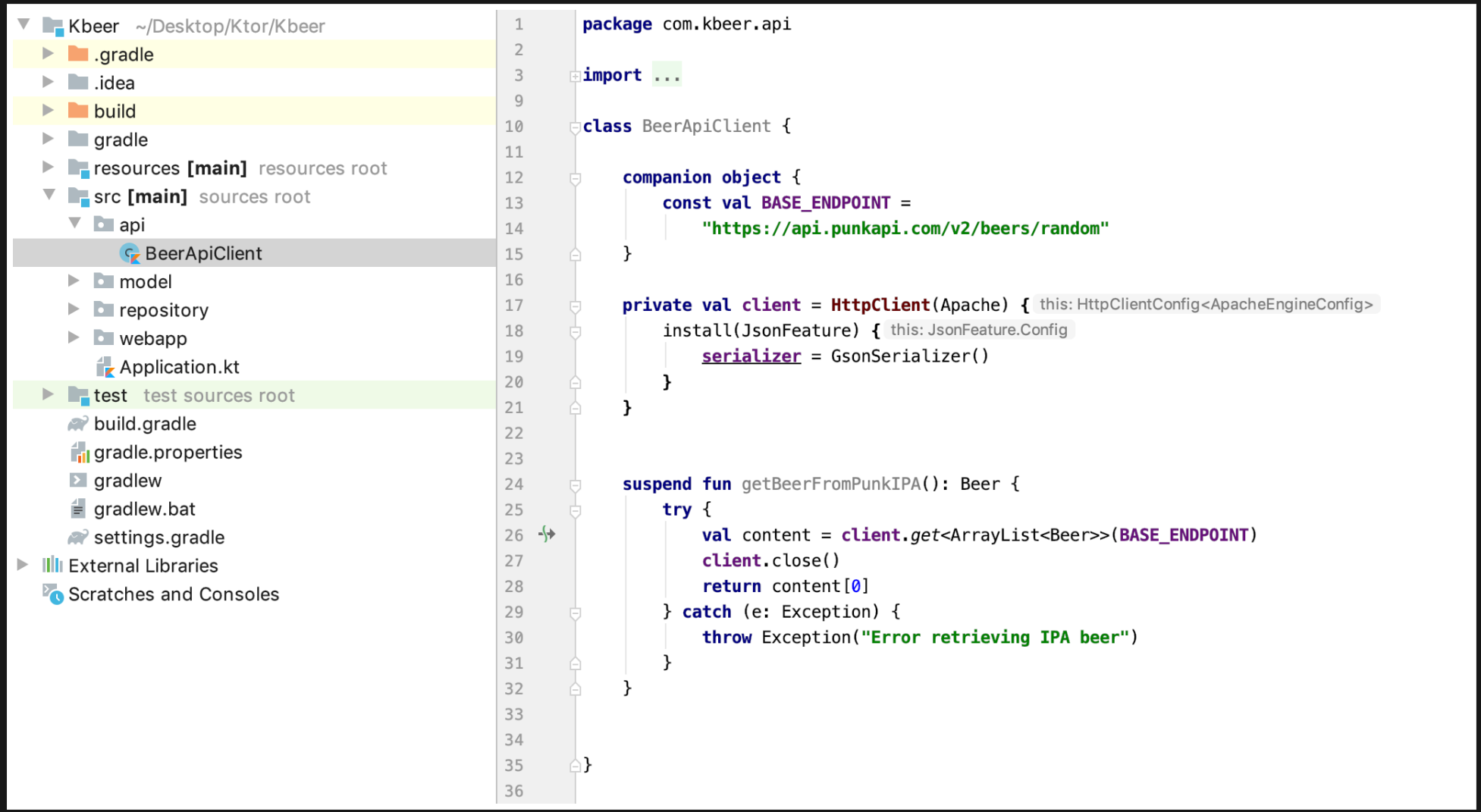
V2 Documentation

Root Endpoint

The root endpoint should prefix all resources and is only accessible over HTTPS. CORS is also enabled.

```
https://api.punkapi.com/v2/
```

Creating our API client



Creating the new endpoint and calling our API client

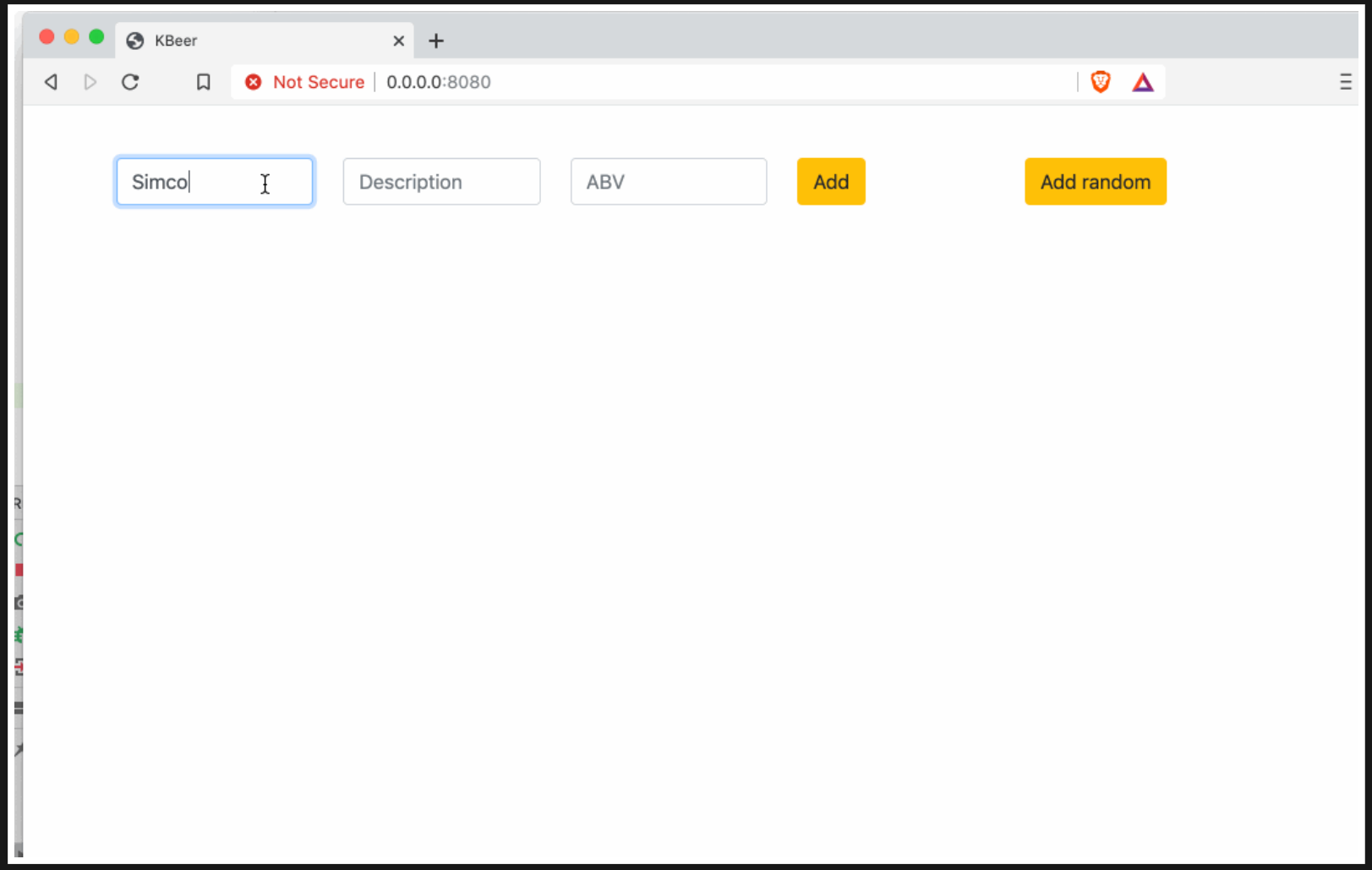
```
27
28      post(BEER) { this: PipelineContext<Unit, ApplicationCall>
29          val parameters = call.receiveParameters()
30          val action = parameters["action"] ?: throw IllegalArgumentException("Missing parameter: action")
31          when (action) {
32              "add" -> {
33                  val name = parameters["name"] ?: throw IllegalArgumentException("Missing parameter: name")
34                  val description =
35                      parameters["description"] ?: throw IllegalArgumentException("Missing parameter: description")
36                  val abv =
37                      parameters["abv"] ?: throw IllegalArgumentException("Missing parameter: ABV")
38                  val beerToAdd = Beer(name, description, abv.toFloat())
39                  beerRepository.add(beerToAdd)
40              }
41              "delete" -> {
42                  val id = parameters["id"] ?: throw IllegalArgumentException("Missing parameter: id")
43                  beerRepository.remove(id.toInt())
44              }
45              "random" -> {
46                  val beerApiClient = BeerApiClient()
47                  val beer = beerApiClient.getBeerFromPunkIPA()
48                  beerRepository.add(beer)
49              }
50          }
51          call.respondRedirect(HOME)
52      }
53  }
```

Modifying the template

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        <div class="container with-margin-top">
            <div class="row with-margin-top">
                <form action="/beer" method="post">
                    <div class="col cell"><input type="text" name="name" class="form-control" placeholder="Name"/></div>
                    <div class="col cell"><input type="text" name="description" class="form-control" placeholder="Description"/></div>
                    <div class="col cell"><input type="text" name="abv" class="form-control" placeholder="ABV"/></div>
                    <div class="col cell"><input type="hidden" name="action" value="add"></div>
                    <div class="col cell"><button type="submit" class="btn btn-warning">Add</button></div>
                </form>
                <div class="col cell">
                    <form action="/beer" method="post">
                        <input type="hidden" name="action" value="random">
                        <button type="submit" class="btn btn-warning">Add random</button>
                    </form>
                </div>
            </div>
        </div>
        <#if beers?has_content>
            <div class="container with-margin-top with-margin-bottom">
                <div class="row header">
                    <div class="col-2 cell"><b>Name</b></div>
                    <div class="col-7 cell"><b>Description</b></div>
                    <div class="col-1 cell"><b>ABV</b></div>
                    <div class="col-2 cell"> </div>
                </div>
                <#list beers as beer>
                  <div class="row with-border-bottom">
                    <div class="col-2 cell">${beer.name}</div>
                    <div class="col-7 cell">${beer.description}</div>
                    <div class="col-1 cell">${beer.abv?string(",##0.0")}</div>
                    <div class="col-2 cell">
                        <form action="/beer" method="post">
```

```
42                         <button type="submit" class="btn btn-danger">Delete</button>
43                         <input type="hidden" name="action" value="delete">
44                         <input class="btn btn-danger" type="hidden" name="id" value="${beer.id}" />
45                     </form>
46                 </div>
```

Adding a random beer (in action)



WHAT'S NEXT?

- Persistence and sessions
- Authentication
- Deployment

PROS AND CONS

✓ #KotlinEverywhere! (literally)

✓ Lightweight and customizable

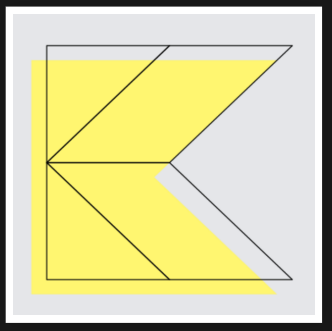
✗ Documentation a bit confusing

✗ Choosing features when creating a new project

MATERIAL I FOUND HELPFUL

- Ryan Harter's video from KotlinConf 2018: https://www.youtube.com/watch?v=V4PS3IjIzlw
- Dan Kim's video and slides from KotlinConf 2019: https://dankim.org/2019/12/16/kotlinconf-2019-ktor.html
- Hadi Hariri's post on Ktor: https://www.infoq.com/articles/microservices-kotlin-ktor
- Ray Wenderlich's "Server-Side Kotlin with Ktor": https://www.raywenderlich.com/2885892-server-side-kotlin-with-ktor

(Have feedback?)

THAN  YOU!