40009 ExerciseTypes.PPT10

Kotlin Web Server

Submitters

---

anb122 Adithya Narayanan

Emarking

```
Final Tests: Summary for anb122 of j1
PPT 24
-------------------------------------

  Public Tests:
    Compiles:     1 / 1
    Tests Pass:   1 / 1
    Style Check:  1 / 1

Git Repo: git@gitlab.doc.ic.ac.uk:lab2324_autumn/kotlinwebserver_anb122.git
Commit ID: ad887
```

Tests: 4/4

Quality: 3/3

Correctness: 2/3

9/10

```
    1: package webserver
    2:
    3: import java.util.*
    4:
    5: // provided files
    6:
    7: class Request(val url: String, val authToken: String = "")
    8:
    9: class Response(val status: Status, val body: String = "")
   10:
   11: enum class Status(val code: Int) {
   12:   OK(200),
   13:   FORBIDDEN(403),
   14:   NOT_FOUND(404)
   15: }
   16:
   17: fun helloHandler(x: Request): Response {
   18:   var queryParamVar = queryParams(x.url)
   19:   var nameList = queryParamVar.filter { x -> x.first == "name" }
   20:   var styleList = queryParamVar.filter { x -> x.first == "style" }
   21:   if (nameList.size != 0 && styleList.size != 0) {
   22:     return Response(Status.OK, "HELLO, " + ⬈
nameList[0].second.uppercase(Locale.getDefault()) + "!")
   23:   } else if (nameList.size != 0 && styleList.size == 0) {
   24:     return Response(Status.OK, "Hello, " + nameList[0].second + "!")
   25:   }
   26:   return Response(Status.OK, "Hello, World!")
   27: }
   28:
   29: fun imperialHandler(x: Request): Response {
   30:   var pathOfURL = path(x.url)
   31:   if (pathOfURL == "/computing") {
   32:     return Response(Status.OK, "This is DoC.")
   33:   }
   34:   return Response(Status.OK, "This is Imperial.")
   35: }
   36:
   37: fun route(requestVar: Request): Response {
   38:   if (path(requestVar.url) == "/say-hello") {
   39:     return helloHandler(requestVar)
   40:   } else if (path(requestVar.url) == "/" || path(requestVar.url) == ⬈
"/computing") {
   41:     return imperialHandler(requestVar)
   42:   }
   43:   return Response(Status.NOT_FOUND)
   44: }
```

Can use list.isEmpty()

Should call homePageHandler() here.


```
    1: package webserver
    2:
    3: // write your web framework code here:
    4:
    5: fun scheme(url: String): String = url.substringBefore("://")
    6:
    7: fun host(url: String): String {
    8:   val partBeforeHost = url.substringAfter("://")
    9:   val partAfterHost = partBeforeHost.substringAfter("/")
   10:   val hostName = partBeforeHost.replace("/" + partAfterHost, "")
   11:   return hostName
   12: }
   13:
   14: fun path(url: String): String {
   15:   var followingHost = url.substringAfter(host(url))
   16:   var querySeparation = followingHost.substringAfter("?")
   17:   var path = followingHost.replace("?" + querySeparation, "")
   18:   return path
   19: }
   20:
   21: fun queryParams(url: String): List<Pair<String, String>> {
   22:   if ("?" in url) {
   23:     var followingPath = url.substringAfter("?")
   24:     var listOfQueries = followingPath.split("&")
   25:     return listOfQueries.map { x -> Pair(x.substringBefore("="), ⬈
x.substringAfter("=")) }
   26:   } else {
   27:     return emptyList()
   28:   }
   29: }
   30:
   31: // http handlers for a particular website...
   32:
   33: fun homePageHandler(request: Request): Response = Response(Status.OK, "This is ⬈
Imperial.")
```

You can chain function calls. s.before().after().after()...

Can use substringBefore()

```
package webserver

import org.junit.Test
import kotlin.test.assertEquals

class WebServerTest {

  @Test
  fun `can extract scheme`() {
    assertEquals("http", scheme("http://www.imperial.ac.uk/"))
    assertEquals("https", scheme("https://www.imperial.ac.uk/"))
  }

  @Test
  fun `can extract host`() {
    assertEquals("www.imperial.ac.uk", host("http://www.imperial.ac.uk/"))
    assertEquals("www.imperial.ac.uk", host("https://www.imperial.ac.uk/"))
    assertEquals("www.imperial.ac.uk", host("https://www.imperial.ac.uk/computing"))
  }

  @Test
  fun `can extract path`() {
    assertEquals("/", path("http://www.imperial.ac.uk/"))
    assertEquals("/", path("https://www.imperial.ac.uk/"))
    assertEquals("/computing", path("https://www.imperial.ac.uk/computing"))
    assertEquals("/computing/programming", path("https://www.imperial.ac.uk/computing/programming"))
    assertEquals("/computing", path("https://www.imperial.ac.uk/computing?q=abc"))
  }

  @Test
  fun `can extract query params`() {
    assertEquals(listOf(Pair("q", "xxx")), queryParams("http://www.imperial.ac.uk/?q=xxx"))
    assertEquals(listOf(Pair("q", "xxx"), Pair("rr", "zzz")), queryParams("http://www.imperial.ac.uk/?q=xxx&rr=zzz"))
  }

  @Test
  fun `when no query params in url, empty list is extracted`() {
    assertEquals(listOf(), queryParams("http://www.imperial.ac.uk/"))
  }

// ***** Tests for Handlers *****

  @Test
  fun `says hello world`() {
    val request = Request("http://www.imperial.ac.uk/say-hello")
    assertEquals("Hello, World!", helloHandler(request).body)
  }

  @Test
  fun `can be customised with particular name`() {
    val request = Request("http://www.imperial.ac.uk/say-hello?name=Fred")
    assertEquals("Hello, Fred!", helloHandler(request).body)
  }

  @Test
  fun `can process multiple params`() {
    val request = Request("http://www.imperial.ac.uk/say-hello?name=Fred&style=shouting")
    assertEquals("HELLO, FRED!", helloHandler(request).body)
  }

```



```
// ***** Tests for Routing *****

  @Test
  fun `can route to hello handler`() {
    val request = Request("http://www.imperial.ac.uk/say-hello?name=Fred")
    assertEquals("Hello, Fred!", route(request).body)
  }

  @Test
  fun `can route to homepage handler`() {
    assertEquals("This is Imperial.", route(Request("http://www.imperial.ac.uk/")).body)
    assertEquals("This is DoC.", route(Request("http://www.imperial.ac.uk/computing")).body)
  }

  @Test
  fun `gives 404 when no matching route`() {
    assertEquals(Status.NOT_FOUND, route(Request("http://www.imperial.ac.uk/not-here")).status)
  }

//  ***** Tests for the Extensions *****

// *** More flexible routing ***

//  @Test
//  fun `calling configureRoutes() returns app which can handle requests`() {
//
//    val app = configureRoutes(...)
//
//    assertEquals("This is Imperial.", app(Request("http://www.imperial.ac.uk/")).body)
//    assertEquals("This is DoC.", app(Request("http://www.imperial.ac.uk/computing")).body)
//  }

//  *** Filters ***

//  @Test
//  fun `filter prevents access to protected resources `() {
//
//    val app = configureRoutes(...)
//
//    val request = Request("http://www.imperial.ac.uk/exam-marks")
//    assertEquals(Status.FORBIDDEN, app(request).status)
//  }
//
//  @Test
//  fun `filter allows access to protected resources with token`() {
//
//    val app = configureRoutes(...)
//
//    val request = Request("http://www.imperial.ac.uk/exam-marks", "password1")
//    assertEquals(Status.OK, app(request).status)
//    assertEquals("This is very secret.", app(request).body)
//  }
}
```

```
-------- Test Output --------
Running LabTS build... (Wed 15 Nov 23:38:09 UTC 2023)

Submission summary...
You made 6 commits
  - b4216b7 Completed Part 1 and 2 [4 files changed, 87 insertions, 37 deletions]
  - c6b7c7e Cleaned up formatting [1 file changed, 2 insertions, 2 deletions]
  - b7131aa Updated build.sh to resolve style check [1 file changed, 1 insertion, 1 deletion]
  - e8a42a1 Code cleanup [2 files changed, 5 insertions, 6 deletions]
  - 884504d Added newline to end of file [1 file changed, 1 insertion, 1 deletion]
  - ad8877f Cleaned up Test.kt [1 file changed, 1 insertion, 1 deletion]

Preparing...

BUILD SUCCESSFUL in 844ms

Compiling...Path for java installation '/usr/lib/jvm/openjdk-17' (Common Linux Locations) does not contain a java executable

w: file:///tmp/d20231115-34-nljf1/src/main/kotlin/webserver/Http.kt:19:41 Name shadowed: x
w: file:///tmp/d20231115-34-nljf1/src/main/kotlin/webserver/Http.kt:20:42 Name shadowed: x
w: file:///tmp/d20231115-34-nljf1/src/main/kotlin/webserver/WebServer.kt:33:21 Parameter 'request' is never used


BUILD SUCCESSFUL in 11s

Running tests...Path for java installation '/usr/lib/jvm/openjdk-17' (Common Linux Locations) does not contain a java executable


webserver.WebServerTest > says hello world PASSED

webserver.WebServerTest > can extract scheme PASSED

webserver.WebServerTest > can process multiple params PASSED

webserver.WebServerTest > can route to hello handler PASSED

webserver.WebServerTest > can be customised with particular name PASSED

webserver.WebServerTest > can route to homepage handler PASSED

webserver.WebServerTest > can extract host PASSED

webserver.WebServerTest > can extract path PASSED

webserver.WebServerTest > when no query params in url, empty list is extracted PASSED

webserver.WebServerTest > can extract query params PASSED

webserver.WebServerTest > gives 404 when no matching route PASSED

BUILD SUCCESSFUL in 1s

Checking code style...
BUILD SUCCESSFUL in 1s
Finished auto test. (Wed 15 Nov 23:38:52 UTC 2023)

-------- Test Errors --------

```

```
Test Preview: Summary for anb122 of j1
PPT 24
--------------------------------------

  Public Tests:
    Compiles:     1 / 1
    Tests Pass:   1 / 1
    Style Check:  1 / 1

Git Repo: git@gitlab.doc.ic.ac.uk:lab2324_autumn/kotlinwebserver_anb122.git
Commit ID: ad887
```

```
 1: package webserver
 2:
 3: import java.util.*
 4:
 5: // provided files
 6:
 7: class Request(val url: String, val authToken: String = "")
 8:
 9: class Response(val status: Status, val body: String = "")
10:
11: enum class Status(val code: Int) {
12:   OK(200),
13:   FORBIDDEN(403),
14:   NOT_FOUND(404)
15: }
16:
17: fun helloHandler(x: Request): Response {
18:   var queryParamVar = queryParams(x.url)
19:   var nameList = queryParamVar.filter { x -> x.first == "name" }
20:   var styleList = queryParamVar.filter { x -> x.first == "style" }
21:   if (nameList.size != 0 && styleList.size != 0) {
22:     return Response(Status.OK, "HELLO, " + ⬈
nameList[0].second.uppercase(Locale.getDefault()) + "!")
23:   } else if (nameList.size != 0 && styleList.size == 0) {
24:     return Response(Status.OK, "Hello, " + nameList[0].second + "!")
25:   }
26:   return Response(Status.OK, "Hello, World!")
27: }
28:
29: fun imperialHandler(x: Request): Response {
30:   var pathOfURL = path(x.url)
31:   if (pathOfURL == "/computing") {
32:     return Response(Status.OK, "This is DoC.")
33:   }
34:   return Response(Status.OK, "This is Imperial.")
35: }
36:
37: fun route(requestVar: Request): Response {
38:   if (path(requestVar.url) == "/say-hello") {
39:     return helloHandler(requestVar)
40:   } else if (path(requestVar.url) == "/" || path(requestVar.url) == ⬈
"/computing") {
41:     return imperialHandler(requestVar)
42:   }
43:   return Response(Status.NOT_FOUND)
44: }
```



```
 1: package webserver
 2:
 3: // write your web framework code here:
 4:
 5: fun scheme(url: String): String = url.substringBefore("://")
 6:
 7: fun host(url: String): String {
 8:   val partBeforeHost = url.substringAfter("://")
 9:   val partAfterHost = partBeforeHost.substringAfter("/")
10:   val hostName = partBeforeHost.replace("/" + partAfterHost, "")
11:   return hostName
12: }
13:
14: fun path(url: String): String {
15:   var followingHost = url.substringAfter(host(url))
16:   var querySeparation = followingHost.substringAfter("?")
17:   var path = followingHost.replace("?" + querySeparation, "")
18:   return path
19: }
20:
21: fun queryParams(url: String): List<Pair<String, String>> {
22:   if ("?" in url) {
23:     var followingPath = url.substringAfter("?")
24:     var listOfQueries = followingPath.split("&")
25:     return listOfQueries.map { x -> Pair(x.substringBefore("="), ⬈
x.substringAfter("=")) }
26:   } else {
27:     return emptyList()
28:   }
29: }
30:
31: // http handlers for a particular website...
32:
33: fun homePageHandler(request: Request): Response = Response(Status.OK, "This is ⬈
Imperial.")
```

```
    1: package webserver
    2:
    3: import org.junit.Test
    4: import kotlin.test.assertEquals
    5:
    6: class WebServerTest {
    7:
    8:   @Test
    9:   fun `can extract scheme`() {
   10:     assertEquals("http", scheme("http://www.imperial.ac.uk/"))
   11:     assertEquals("https", scheme("https://www.imperial.ac.uk/"))
   12:   }
   13:
   14:   @Test
   15:   fun `can extract host`() {
   16:     assertEquals("www.imperial.ac.uk", host("http://www.imperial.ac.uk/"))
   17:     assertEquals("www.imperial.ac.uk", host("https://www.imperial.ac.uk/"))
   18:     assertEquals("www.imperial.ac.uk", ⇗
host("https://www.imperial.ac.uk/computing"))
   19:   }
   20:
   21:   @Test
   22:   fun `can extract path`() {
   23:     assertEquals("/", path("http://www.imperial.ac.uk/"))
   24:     assertEquals("/", path("https://www.imperial.ac.uk/"))
   25:     assertEquals("/computing", path("https://www.imperial.ac.uk/computing"))
   26:     assertEquals("/computing/programming", ⇗
path("https://www.imperial.ac.uk/computing/programming"))
   27:     assertEquals("/computing", ⇗
path("https://www.imperial.ac.uk/computing?q=abc"))
   28:   }
   29:
   30:   @Test
   31:   fun `can extract query params`() {
   32:     assertEquals(listOf(Pair("q", "xxx")), ⇗
queryParams("http://www.imperial.ac.uk/?q=xxx"))
   33:     assertEquals(listOf(Pair("q", "xxx"), Pair("rr", "zzz")), ⇗
queryParams("http://www.imperial.ac.uk/?q=xxx&rr=zzz"))
   34:   }
   35:
   36:   @Test
   37:   fun `when no query params in url, empty list is extracted`() {
   38:     assertEquals(listOf(), queryParams("http://www.imperial.ac.uk/"))
   39:   }
   40:
   41: // ***** Tests for Handlers *****
   42:
   43:   @Test
   44:   fun `says hello world`() {
   45:     val request = Request("http://www.imperial.ac.uk/say-hello")
   46:     assertEquals("Hello, World!", helloHandler(request).body)
   47:   }
   48:
   49:   @Test
   50:   fun `can be customised with particular name`() {
   51:     val request = Request("http://www.imperial.ac.uk/say-hello?name=Fred")
   52:     assertEquals("Hello, Fred!", helloHandler(request).body)
   53:   }
   54:
   55:   @Test
   56:   fun `can process multiple params`() {
   57:     val request = ⇗
Request("http://www.imperial.ac.uk/say-hello?name=Fred&style=shouting")
   58:     assertEquals("HELLO, FRED!", helloHandler(request).body)
   59:   }
   60:
```



```
   61: // ***** Tests for Routing *****
   62:
   63:   @Test
   64:   fun `can route to hello handler`() {
   65:     val request = Request("http://www.imperial.ac.uk/say-hello?name=Fred")
   66:     assertEquals("Hello, Fred!", route(request).body)
   67:   }
   68:
   69:   @Test
   70:   fun `can route to homepage handler`() {
   71:     assertEquals("This is Imperial.", ⇗
route(Request("http://www.imperial.ac.uk/")).body)
   72:     assertEquals("This is DoC.", ⇗
route(Request("http://www.imperial.ac.uk/computing")).body)
   73:   }
   74:
   75:   @Test
   76:   fun `gives 404 when no matching route`() {
   77:     assertEquals(Status.NOT_FOUND, ⇗
route(Request("http://www.imperial.ac.uk/not-here")).status)
   78:   }
   79:
   80: //  ***** Tests for the Extensions *****
   81:
   82: // *** More flexible routing ***
   83:
   84: //  @Test
   85: //  fun `calling configureRoutes() returns app which can handle requests`() {
   86: //
   87: //    val app = configureRoutes(...)
   88: //
   89: //    assertEquals("This is Imperial.", ⇗
app(Request("http://www.imperial.ac.uk/")).body)
   90: //    assertEquals("This is DoC.", ⇗
app(Request("http://www.imperial.ac.uk/computing")).body)
   91: //  }
   92:
   93: //  *** Filters ***
   94:
   95: //  @Test
   96: //  fun `filter prevents access to protected resources `() {
   97: //
   98: //    val app = configureRoutes(...)
   99: //
  100: //    val request = Request("http://www.imperial.ac.uk/exam-marks")
  101: //    assertEquals(Status.FORBIDDEN, app(request).status)
  102: //  }
  103: //
  104: //  @Test
  105: //  fun `filter allows access to protected resources with token`() {
  106: //
  107: //    val app = configureRoutes(...)
  108: //
  109: //    val request = Request("http://www.imperial.ac.uk/exam-marks", "password1")
  110: //    assertEquals(Status.OK, app(request).status)
  111: //    assertEquals("This is very secret.", app(request).body)
  112: //  }
  113: }
```

```
-------- Test Output --------
Running LabTS build... (Wed 15 Nov 23:38:09 UTC 2023)

Submission summary...
You made 6 commits
  - b4216b7 Completed Part 1 and 2 [4 files changed, 87 insertions, 37 deletions]
  - c6b7c7e Cleaned up formatting [1 file changed, 2 insertions, 2 deletions]
  - b7131aa Updated build.sh to resolve style check [1 file changed, 1 insertion, 1 deletion]
  - e8a42a1 Code cleanup [2 files changed, 5 insertions, 6 deletions]
  - 884504d Added newline to end of file [1 file changed, 1 insertion, 1 deletion]
  - ad8877f Cleaned up Test.kt [1 file changed, 1 insertion, 1 deletion]

Preparing...

BUILD SUCCESSFUL in 844ms

Compiling...Path for java installation '/usr/lib/jvm/openjdk-17' (Common Linux Locations) does not contain a java executable

w: file:///tmp/d20231115-34-nljf1/src/main/kotlin/webserver/Http.kt:19:41 Name shadowed: x
w: file:///tmp/d20231115-34-nljf1/src/main/kotlin/webserver/Http.kt:20:42 Name shadowed: x
w: file:///tmp/d20231115-34-nljf1/src/main/kotlin/webserver/WebServer.kt:33:21 Parameter 'request' is never used


BUILD SUCCESSFUL in 11s

Running tests...Path for java installation '/usr/lib/jvm/openjdk-17' (Common Linux Locations) does not contain a java executable


webserver.WebServerTest > says hello world PASSED

webserver.WebServerTest > can extract scheme PASSED

webserver.WebServerTest > can process multiple params PASSED

webserver.WebServerTest > can route to hello handler PASSED

webserver.WebServerTest > can be customised with particular name PASSED

webserver.WebServerTest > can route to homepage handler PASSED

webserver.WebServerTest > can extract host PASSED

webserver.WebServerTest > can extract path PASSED

webserver.WebServerTest > when no query params in url, empty list is extracted PASSED

webserver.WebServerTest > can extract query params PASSED

webserver.WebServerTest > gives 404 when no matching route PASSED

BUILD SUCCESSFUL in 1s

Checking code style...
BUILD SUCCESSFUL in 1s
Finished auto test. (Wed 15 Nov 23:38:52 UTC 2023)

-------- Test Errors --------

```