Integrating the Service with Controller

We'll cover the following

Integrating

It's time to modify the `TaskController` we wrote earlier, to add two new methods and to modify the existing `tasks()` method—that is, the GET operation.

Kotlin vs. Java

Once again, if we were to write in Java, here's what the modified controller would look like:

```
//Java code only for comparison purpose
package com.agiledeveloper.todo;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;
import org.springframework.http.ResponseEntity;

@RestController
@RequestMapping("/task")
class TaskController {

  private final TaskService service;

  public TaskController(TaskService service) {
    this.service = service;
  }

  @GetMapping
  ResponseEntity<Iterable<Task>> tasks() {
    return ResponseEntity.ok(service.getAll());
  }

  @PostMapping
  ResponseEntity<String> create(@RequestBody Task task) {
    Task result = service.save(task);

    return ResponseEntity.ok(
      "added task with description " + result.getDescription());
  }
```

```
  @DeleteMapping("/{id}")
  ResponseEntity<String> delete(@PathVariable Long id) {

    if (service.delete(id)) {
      return ResponseEntity.ok("Task with id " + id + " deleted");
    }

    return ResponseEntity.status(404)
      .body("Task with id " + id + " not found");
  }
}
```

Let's change the Kotlin version of the `TaskController` to add the necessary operations:

```
package com.agiledeveloper.todo

import org.springframework.web.bind.annotation.*
import org.springframework.http.ResponseEntity

@RestController
@RequestMapping("/task")

class TaskController(val service: TaskService) {

  @GetMapping
  fun tasks() = ResponseEntity.ok(service.getAll())

  @PostMapping
  fun create(@RequestBody task: Task): ResponseEntity<String> {
    val result = service.save(task)

    return ResponseEntity.ok(
      "added task with description ${result.description}")
  }

  @DeleteMapping("/{id}")
  fun delete(@PathVariable id: Long) = if (service.delete(id)) {
    ResponseEntity.ok("Task with id $id deleted")
  } else {
    ResponseEntity.status(404).body("Task with id $id not found")
  }

}
```

TaskController.kt

First, we modified the class to add a property service of type `TaskService`. Spring will automatically inject a reference to that dependency when the instance of `TaskController` is created. Next, we modified the `tasks()` method to return the result of the `getAll()` method of the service. We also added two more methods, `create()` and `delete()`.

The `create()` method has been annotated to specify that it supports the POST

HTTP method, and it accepts a `Task` as post data through the body of the incoming

request. The method invokes the service's `save()` method to save the given object to the database.

The `delete()` method supports the DELETE HTTP method and the annotation also specifies that the request should include the `id` of the task to be deleted. The method forwards the request to the service's `delete()` method and returns an appropriate HTTP response.

The next lesson will show this application in action.