

Simple Server 2 and Test

진행할 것

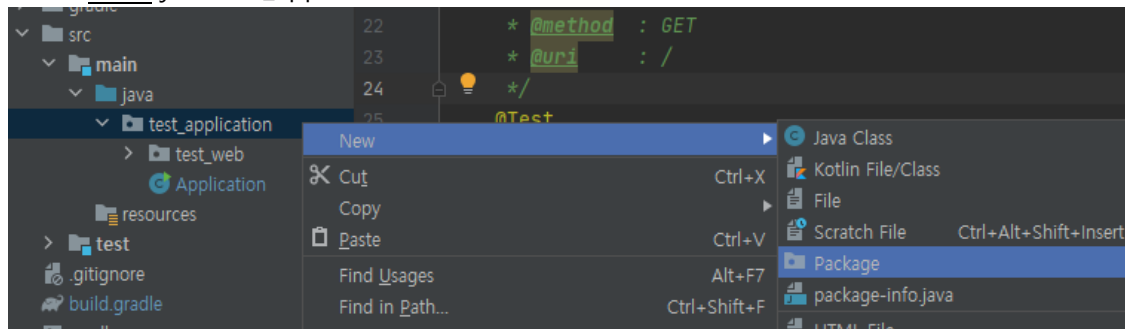
- 최종목표
 - ✓ 요청에 맞는 Java 객체를 응답한다.
- 최종목표에 도달하기 위해 수행할 것
 1. Person 클래스 생성
 2. Person 클래스 Test 진행
 3. Controller 클래스에 Person 객체를 응답하기 위한 API 생성
 4. Controller 클래스 Test 진행
 5. Controller의 응답을 웹 브라우저(크롬)로 확인

실습) Simple Server 2

※ 전 3챕터(Simple Server Test)에 이어서 진행하면 됩니다.

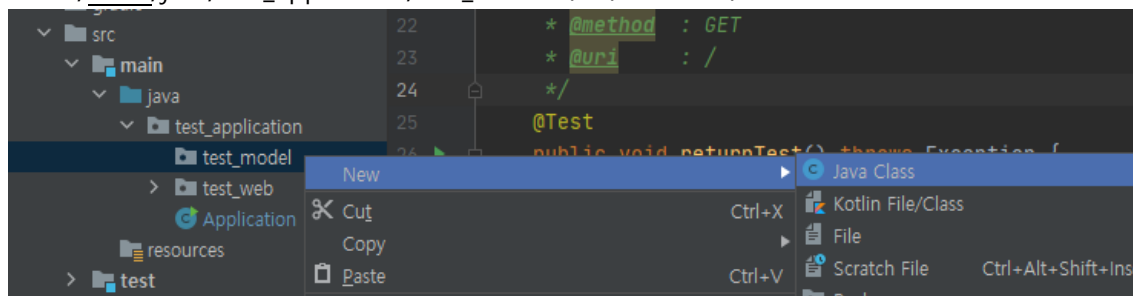
1. Person 클래스 생성

1.1. src/main/java/test_application에 새로운 패키지 생성



1.2. 패키지 이름은 "test_model"로 작성

1.3. src/main/java/test_application/test_model에 새로운 클래스 생성



1.4. 클래스 이름은 "Person"으로 작성

1.5. Person 클래스의 내용을 아래 박스와 같이 작성

```
package test_application.test_model;

public class Person {
    private final String name;
    private final int age;

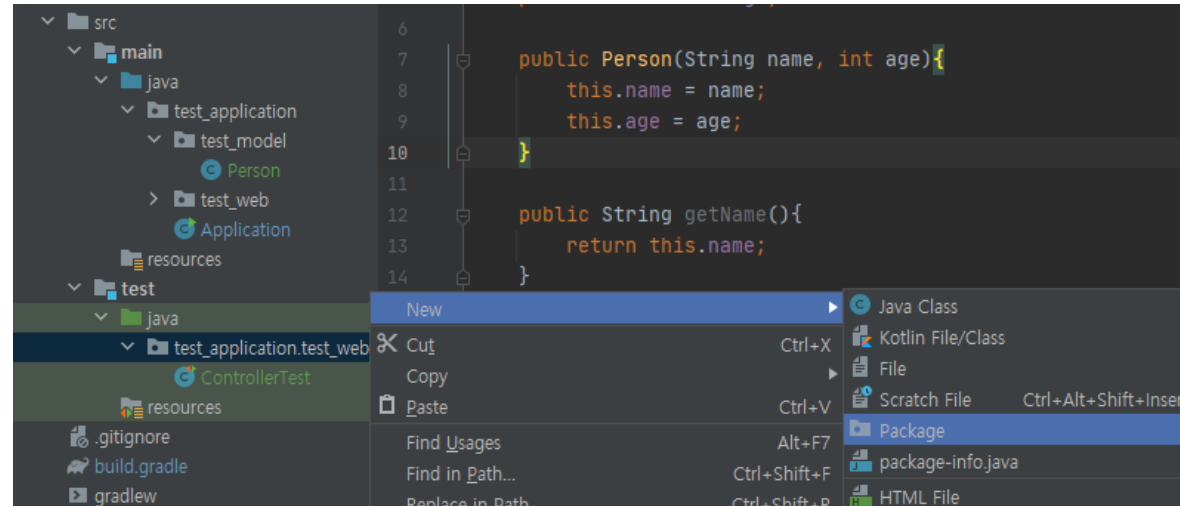
    public Person(String name, int age){
        this.name = name;
        this.age = age;
    }

    public String getName(){
        return this.name;
    }

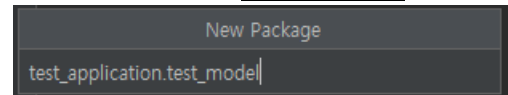
    public int getAge(){
        return this.age;
    }
}
```

2. Person 클래스 Test 진행

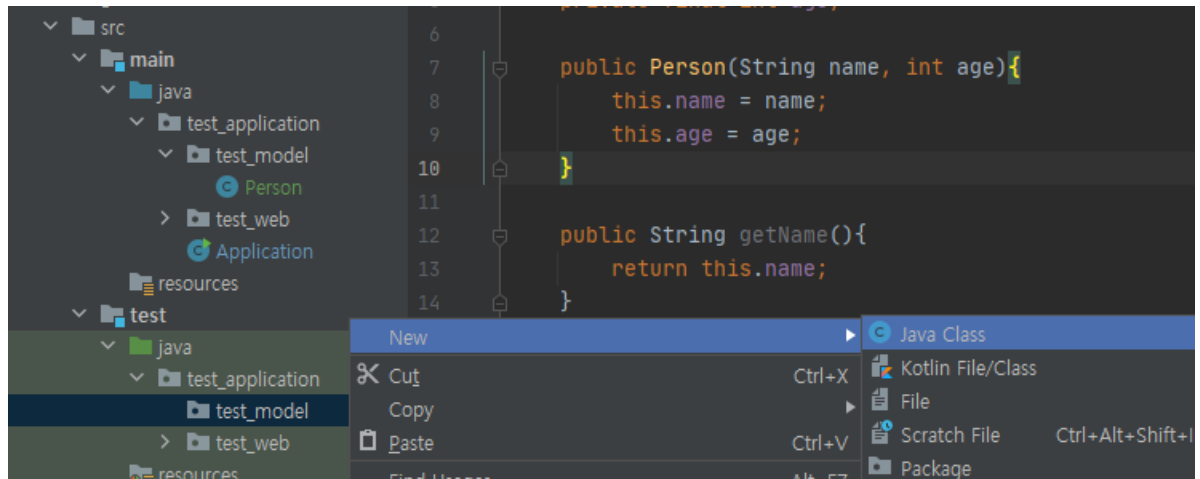
2.1. src/test/java/test_application에 새로운 패키지 생성



2.2. 패키지 이름은 “test_model”로 작성



2.3. src/test/java/test_application/test_model에 새로운 클래스 생성



2.4. 클래스 이름은 **"PersonTest"**로 작성

2.5. PersonTest 클래스의 내용을 아래 박스와 같이 작성

```
package test_application.test_model;

import org.junit.Test;
import static org.assertj.core.api.Assertions.assertThat;

public class PersonTest {

    @Test
    public void personTest(){
        // given
        String name = "HongGilDong";
        int age = 20;

        // when
        Person hong = new Person(name, age); // Person(이름 : HongGilDong, 나이 : 20)

        // then
        assertThat(hong.getName()).isEqualTo(name); // hong(name)과 name을 비교
        assertThat(hong.getAge()).isEqualTo(age); // hong(age)와 age를 비교
    }
}
```

2.6. 테스트 진행

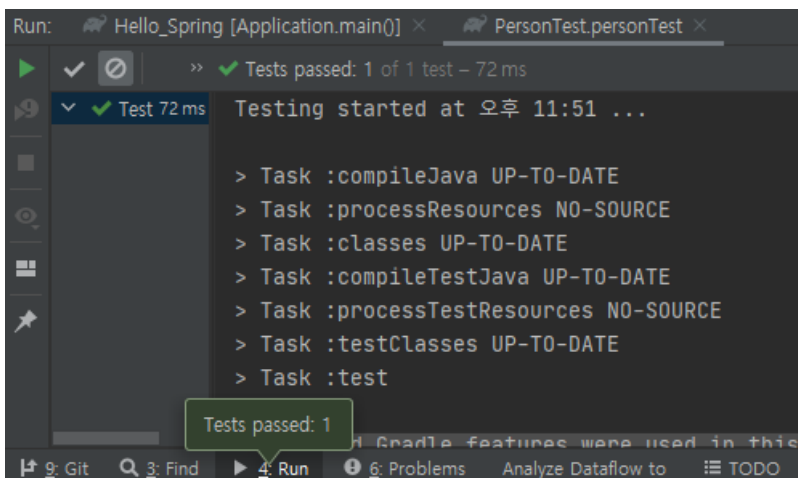


: PersonTest 클래스의 personTest() 메소드 왼쪽 초록색 화살표 클릭



: Run 'PersonTest.personTest' 클릭

2.7. 테스트 결과



>> hong 객체의 name이 "HongGilDong", age는 20

>> 지역 name 변수가 "HongGilDong" age는 20

>> hong 객체의 name, age와 지역 변수 name, age의 값이 같으므로 테스트 성공

3. Controller 클래스에 Person 객체를 응답하기 위한 API 생성

3.1. src/main/java/test_application/test_web/Controller 클래스에 아래 박스와 같이 추가작성

```
...(중략)
@GetMapping("/person")
public Person responsePerson
    (@RequestParam("name") String name, @RequestParam("age") int age) {
    // @RequestParam(String key)
    // 요청되는 파라미터의 key값에 해당하는 value를 변수의 자료형에 맞게 매핑 시켜주는 어노테이션
    // 만약, @RequestParam을 적용하였을 시 반드시 그에 해당하는 요청 데이터가 같이 전달되어야 한다.
    // key에 해당하는 요청데이터가 전달되지 않는다면 HTTP 400 ERROR가 반환된다.
    // GET 요청의 예) http://host_ip:port/person?name=HongGilDong&age=25

    return new Person(name, age);
}
...(중략)
```

4. Controller 클래스 Test 진행

4.1. src/test/java/test_application/test_web/ControllerTest 클래스에 아래 박스와 같이 추가작성

```
...(중략)
/**
 * @method : GET
 * @uri : /person
 */
@Test
public void returnPerson() throws Exception {

    // given
    String name = "HongGilDong";
    int age = 25;

    // when
    mvc.perform(get("/person")
                .param("name", name)
                .param("age", String.valueOf(age))) // 요청

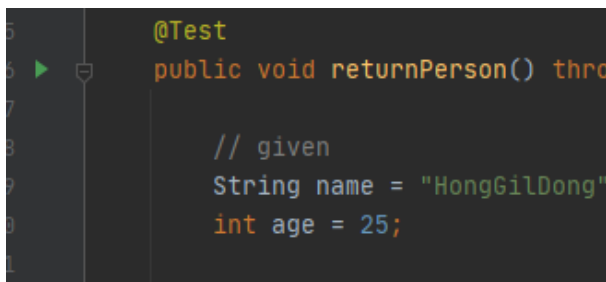
        .andDo(print()) // 요청 및 응답 내용을 출력해주는 메소드

    //then

        // 응답
        .andExpect(status().isOk()) // 상태코드 확인
        .andExpect(jsonPath("$.name").value(name)) // json형식으로 key가 name인 value와 지역 name 비교
        .andExpect(jsonPath("$.age").value(age)) // json형식으로 key가 age인 value와 지역 age 비교
        .andExpect(content().string("{\"name\":\"HongGilDong\",\"age\":25}")); // 응답 컨텐츠 모두 비교

}
...(중략)
```

4.2. 테스트 진행



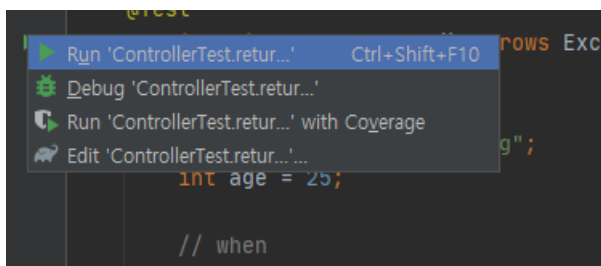
```
@Test
public void returnPerson() throws Exception {

    // given
    String name = "HongGilDong";
    int age = 25;

    // when

    // then
}
```

: ControllerTest 클래스의 returnPerson() 메소드 왼쪽 초록색 화살표 클릭

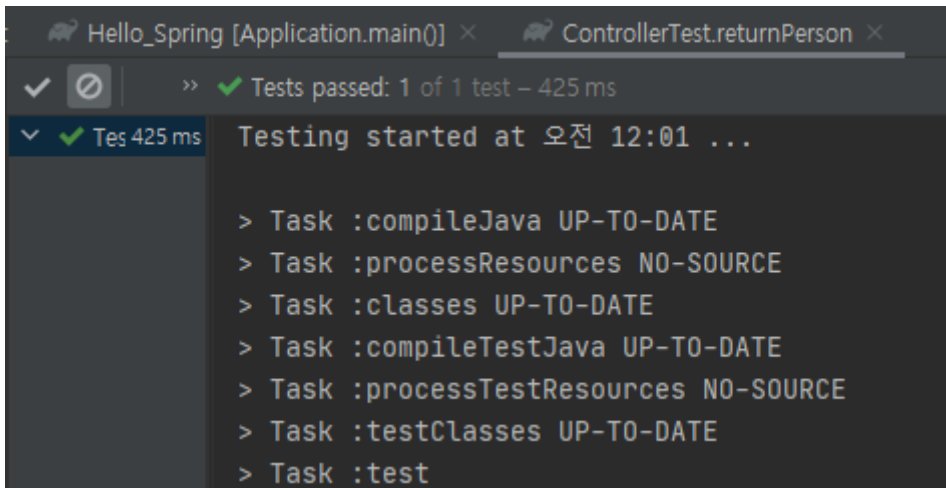


```
Run 'ControllerTest.returnPerson()' Ctrl+Shift+F10
Debug 'ControllerTest.returnPerson()'
Run 'ControllerTest.returnPerson()' with Coverage
Edit 'ControllerTest.returnPerson()'

// when
```

: Run 'ControllerTest.returnPerson()' 클릭

4.3. 테스트 결과



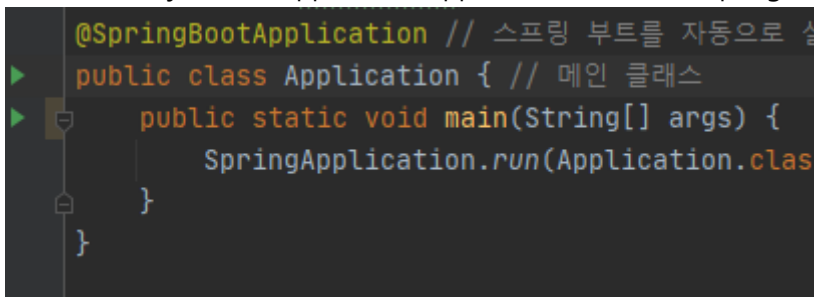
>> 요청한 name, age 토대로 /person URI API에서는 Person 객체를 생성하여 응답함

>> 응답 받은 Person 객체의 name, age와 지역 name, age를 비교하였을 시 같은 값임으로 테스트 성공

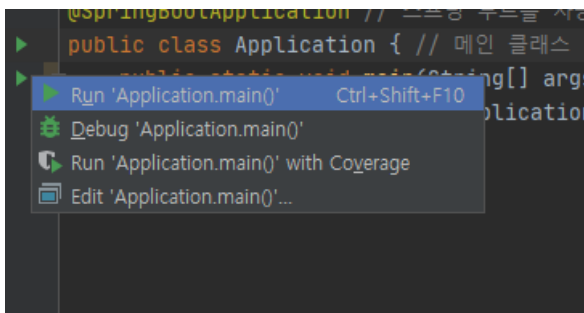
5. Controller의 응답을 웹 브라우저(크롬)로 확인

5.1. 모든 테스트가 정상적으로 성공하였으므로 실제 Application을 실행해보자

5.2. src/main/java/test_application/Application을 실행 (= Spring Boot Server 실행)



: Application 클래스의 main 메소드 왼쪽 초록색 화살표 클릭



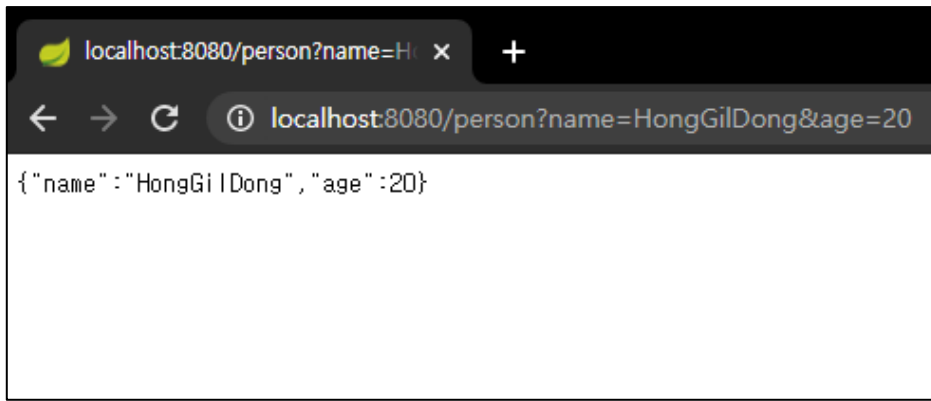
: Run 'Application.main()' 실행

5.3. 서버가 정상적으로 동작하는지 확인

: IntelliJ의 아래에 Run 탭을 클릭하여 확인하도록 한다.

5.4. 웹 브라우저에 Person 객체를 응답 받기 >> <http://localhost:port/person?name=HongGilDong&age=20>

: port 번호를 변경하지 않았다면 디폴트로 8080이다.



: 서버에서 새로 생성된 Person 객체의 필드 값을 응답 받는 것을 확인할 수 있다.

참고문헌

- INLAB_박지훈 (2020). 내 머리 속.