Hello Spring

#01. Spring 개발환경 구성

> JDK 17이 설치되어 있어야 함 Tomcat의 경우 VSCode Extension에 포함되어 있기 때문에 설치하지 않아도 무관함.

1) VSCode 환경설정 추가

```
{
    "java.debug.settings.hotCodeReplace": "auto",
    "spring.dashboard.openWith": "external",
    "java.import.gradle.java.home": "C:\\jdk-17.0.2"
}
```

2) VSCode Extension 설치

익스텐션	개발자
Spring Boot Extension Pack	VMWare

Spring Boot Extension Pack에 종속된 확장

- Spring Boot Tools
- Spring Initializr Java Support
- Spring Boot Dashboard

#02. Spring 프로젝트 생성하기

1. `Ctrl + Shipt + P` > `spring initializr: Create Gradle Project` 선택
2. Spring 버전 선택
    - 첫 번째 항목이 추천항목이므로 첫 항목 선택함
3. 사용할 언어 선택
    - `Java`
4. GroupId 지정
    - 회사 이름의 도메인 역순
    - `kr.hossam`
5. ArtifactId 지정
    - 버전 없는 Jar파일 이름
    - 특수문자는 사용할 수 없고, 소문자만 사용되어야 한다.
    - 즉, 프로그램 이름을 의미함(소문자) : `hellospring`

> `3+4`에 의해 프로그램의 패키지 이름이 `kr.hossam.hellospring`으로 지정됨

5. 배포 타입 지정
    - `Jar`

6. Java Version 선택
    - `17` (설치되어 있는 버전을 선택함)
7. dependency 선택
    - 기본으로 탑제할 라이브러리를 의미
        - `Spring Web`
        - `Spring Boot DevTools`
        - `Spring Boot Acturator Ops`
        - `Thymeleaf Template Engines`
    - 검색 기능 활용 권장
8. 생성할 폴더 지정
    - 작업 폴더를 지정하면 그 안에 `ArtifactId` 이름으로 폴더가 생성됨
9. 생성이 완료되면 VSCode 우측 하단에 로드할지를 묻는 창이 표시됨
    - 여기서 폴더를 바로 열지 말고 윈도우 탐색기를 통해 생성된 프로젝트의 폴더이름을 변경한 후 VSCode로 열것!!!
    - 여기서는 폴더 이름을 `01-HelloSpring`으로 변경함

#03. 테스트

`${ArtifactId}Application.java`에 내용 추가

> `src/main/java/kr/hossam/hellospring/HellospringApplication.java`

1. 클래스 정의문 상단에 `@Controller` 추가
2. 메서드 추가

```
// 리턴하는 문자열을 웹 브라우저에게 전달함
// 이 항목을 명시하지 않을 경우 리턴하는 문자열은 view의 파일이름이 됨
// --> import org.springframework.web.bind.annotation.ResponseBody;
@ResponseBody
// 이 메서드가 연결될 URL 경로를 지정
@GetMapping("/test")
public String hello() {
    String message = "<h1>Hello Spring</h1>";
    message += "<p>안녕하세요~ 스프링~!!</p>";
    System.out.println(message);
    return message;
}
```

실행

- `Spring Boot Dashboard`의 `Apps` 섹션에서 프로젝트 이름 우측에 표시되는 `Debug` 클릭
- `http://localhost:8080/hellospring`의 URL을 통해 결과 확인

> Java Class나 HTML이 수정된 경우 별도의 재시작 없이 컴파일 내용이 갱신되므로 웹 브라우저만 새로고침 하여 결과를 확인한다.
>
> 하지만 build.gradle이 변경된 경우는 프로그램을 재시작 해야 한다.
>
> 컴퓨터 사양에 따라 잘 작동하지 않는 경우도 있다.

#04. View 사용하기

1. 실습을 위한 두 개의 View 파일 추가

View에서 다른 페이지로 링크를 적용할 때는 Controller에서 정의한 URL을 대상으로 해야 한다.

/src/main/resources/templates/now.html

```
<!DOCTYPE html>
<html lang="ko" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        .server-time {
            color: red;
            font-size: 16px;
        }
    </style>
</head>
<body>
    <h1>Hello SpringBoot</h1>
    <p>지금은 <span th:text=${nowtime} class="server-time"></span>입니다.</p>
</body>
</html>
```

/src/main/resources/templates/myview.html

```
<!DOCTYPE html>
<html lang="ko" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Hello SpringBoot</h1>
    <h2 th:text=${message1}></h2>
    <h3 th:text=${message2}></h3>
    <p th:text=${message3}></p>
</body>
</html>
```

4. 컨트롤러 메서드 추가

`HellospringApplication.java`에 아래 내용 추가함

```
/** 1) 서블릿 형태의 컨트롤러 */
// 리턴하는 문자열을 웹 브라우저에게 전달함
// 이 항목을 명시하지 않을 경우 리턴하는 문자열은 view의 파일이름이 됨
// --> import org.springframework.web.bind.annotation.ResponseBody;
@ResponseBody
// 이 메서드가 연결될 URL 경로를 지정
@GetMapping("/hellospring")
public String hello() {
    String message = "<h1>Hello SpringBoot</h1>";
    message += "<p>안녕하세요~ 스프링~!!</p>";
    System.out.println(message);
    return message;
}

/** 2) 자동으로 View의 이름을 찾는 컨트롤러 */
// import org.springframework.ui.Model;
@GetMapping("/now")
public void world(Model model) {
    // import java.util.Date;
    model.addAttribute("nowtime", new Date().toString());

    // void형의 메서드이므로 이 메서드가 사용하는 view의 이름은
    // URL 이름과 동일한 now.html이 된다.
    // --> "/src/main/resources/templates/now.html" 파일을 찾아서 실행
}

/** 3) View의 이름을 반환하는 컨트롤러 */
@GetMapping("/today")
public String nice(Model model) {
    model.addAttribute("message1", "스프링부트 View 테스트 입니다.");
    model.addAttribute("message2", "안녕하세요~");
    model.addAttribute("message3", "반갑습니다~");

    // String형의 메서드 이므로 이 메서드가 리턴하는 문자열이 View의 이름이 된다.
    // --> "/src/main/resources/templates/myview.html" 파일을 찾아서 실행
    return "myview";
}
```

#06. 정적 폴더 사용하기

`/src/main/resources/static` 폴더는 정적 자원(html, css, js, 이미지등)을 웹에 노출하기 위한 폴더

예를들어 `/src/main/resources/static/hello/world/test.html`은
http://localhost:8080`/${contextPath}/hello/world/test.html` 혹은 의 경로로 접근할 수 있음.

View에서 `/${contextPath}/hello/world/test.html` 형태로 링크, `<img>`, `<script>` 태그 등을 통해 정적 자원을 활용할 수 있음

1. 이미지 배치하기

`/src/main/resources/static/img` 폴더를 생성하고 그 안에 `spring.png` 이미지 파일을 배치

2. View에서 사용하기

```
<img th:src="@{/img/spring.png}" alt="springboot" width="300"/>
```