2장 - 스프링 시작하기

> - 스프링 프로젝트 생성
> - 간단한 스프링 예제
> - 스프링 컨테이너

1. 스프링 프로젝트 시작하기

1.1 프로젝트 폴더 생성

1.2 메이븐 프로젝트 생성

메이븐

- 메이븐 프로젝트의 핵심은 `pom.xml` 파일이다
  - 메이븐 프로젝트는 루트 폴더에 `pom.xml` 파일을 갖는다
  - `pom.xml` 은 프로젝트에 대한 설정 정보를 관리한다
    - 필요 의존 모듈, 플러그인 등의 설정 정보
- pom.xml 파일 작성

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
        http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>sp5</groupId>
    <artifactId>sp5-chap02</artifactId>
    <version>0.0.1-SNAPSHOT</version>

    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.0.2.RELEASE</version>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.7.0</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                    <encoding>utf-8</encoding>
                </configuration>
            </plugin>
        </plugins>
    </build>
```

```
</project>
```

- `<artifactId>sp5-chap02</artifactId>`: 프로젝트 식별자
- `<dependency> ... </dependency>`: 프로젝트에서 5.0.2 RELEASE 버전의 spring-context 모듈을 사용한다고 설정
- `<plugin> ... </plugin>`: 1.8 버전 기준으로 자바 소스를 컴파일하고 결과 클래스를 생성한다. 자바 컴파일러가 소스 코드를 읽을 때 사용할 인코딩은 UTF-8로 설정

1.2.1 메이븐 의존 설정

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.0.2.RELEASE</version>
</dependency>
```

- 메이븐은 한 개의 모듈을 아티팩트 단위로 관리한다
- 위 설정은 spring-context라는 식별자를 가진 5.0.2RELEASE 버전의 아티팩트에 대한 의존을 추가한 것
  - 의존 추가: 자바 어플리케이션에서 클래스 패스에 spring-context 모듈을 추가한다는 뜻
  - 아티팩트 이름: "아티팩트이름-버전.jar"
- 위 설정은 메이븐 프로젝트 소스코드 컴파일 및 실행 시 사용할 클래스 패스에 spring-context-5.0.2.RELEASE.jar 파일을 추가한다는 의미이다

1.2.2 메이븐 리포지토리

- spring-context-5.0.2.RELEASE.jar 아티팩트 파일은 어디서 구할까
  - 메이븐 로컬 리포지토리에서 해당 파일이 있는지 검사한다
  - 없으면 메이븐 원격 중앙 리포지토리로부터 해당 파일을 다운받는다
- 메이븐은 기본적으로 [사용자 홈 폴더]/.m2/repository 폴더를 로컬 리포지토리로 사용한다
  - 아티팩트 파일도 여기서 찾을 수 있다
  - 없는 경우 다운받는다

```
C:\eclipes-workspace\sp5-chap02>mvn compile
```

  - 사이트에서 spring-context-5.0.2.RELEASE.jar 파일을 다운 받는다
  - 메이븐 중앙 리포지토리에서 다운 받는 것이다

1.2.3 의존 전이(Transitive Dependencies)

- **의존하는 대상이 의존하는 대상까지도 의존하기에 `의존 전이` 라고 한다**
- `mvn compile`을 통해 spring-context-5.0.2.RELEASE.jar 파일 외에도 다양한 아티팩트 파일을 다운받음
  - `<dependency>`에서 설정한 아티팩트가 다시 의존하는 파일이 포함된다

> Ex. spring-context-5.0.2.RELEASE.jar 다운전에 spring-context-5.0.2.RELEASE.pom 파일을 다운받는다
> spring-context-5.0.2.RELEASE.pom에는 spring-aop, spring-beans, spring-core 아티팩트에 의존한다는 내용이 포함되어 있다

> - **즉 spring-context에 대한 의존 설정이 있으면 spring-context가 의존하는 다른 아티팩트도 함께 다운받는다**

1.2.4 메이븐 기본 폴더 구조

- `src\main\java`
  - **자바 소스 코드**가 위치한다
- XML이나 프로퍼티 파일과 같이 **자바 소스 이외의 다른 자원 파일**의 경우
  - `src\main\resources` 폴더에 위치하면 된다
- 웹 어플리케이션을 개발할 때는
  - `src\main\webapp` 폴더를 기준 폴더로 사용
  - JSP 소스코드, WEB-INF\web.xml 파일 등을 넣는다

```
sp5-chap
  pom.xml
  src - main - java
               resources
               webapp - WEB-INF,web.xml
```

1.2.5 메이븐 프로젝트 임포트

- 이클립스에서 [File] - [Import]를 통해 [Existing Maven Project]를 임포트 한다

> Maven Dependencies 폴더에 메이븐 의존에 설정한 아티팩트가 추가됨

1.3 그레이들 프로젝트 생성

- 메이븐과 생성 과정이 비슷하다
- 차이점: pom.xml 파일 대신에 **build.gradle 파일을 작성한다**

```
apply plugin: 'java'

sourceCompatibility = 1.8
targetCompatibility = 1.8
compileJava.options.encoding = "UTF-8"

repositories {
    mavenCentral()
}

dependencies {
    compile 'org.springframework:spring-context:5.0.2.RELEASE'
}

task wrapper(type: Wrapper) {
    gradleVersion = '4.4'
}
```

```
C:\eclipes-workspace\sp5-chap02>gradle wrapper
```

  - 루트 폴더에 gradlew.bat, gradlew 파일 및 gradle 폴더가 생성된다

```
C:\eclipes-workspace\sp5-chap02>gradlew compileJava
```

- gradle이 아닌 gradlew.bat을 이용해 실행한다

1.3.1 그레이들 프로젝트 임포트

- 이클립스 [File]-[Import...] 를 통해 임포트

1.4 예제 코드 작성

> Greeter.java : 콘솔에 간단한 메시지를 출력하는 자바 클래스
>
> AppContext.java : 스프링 설정 파일
>
> Main.java : main()메서드를 통해 스프링과 Greeter를 실행하는 자바 클래스

- Greeter 클래스

```
package chap02;

public class Greeter {
    private String format;

    public String greet(String guest) {
        return String.format(format, guest);
    }

    public void setFormat(String format) {
        this.format = format;
    }

}
```

- AppContext 클래스

```
package chap02;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class AppContext {

    @Bean
    public Greeter greeter() {
        Greeter g = new Greeter();
        g.setFormat("%s, 안녕하세요!");
        return g;
    }
}
```

- `@Configuration` 애노테이션은 해당 클래스를 **스프링 설정 클래스로 지정한다**
- `@Bean` 밑 줄 코드들은 한 개의 객체를 생성하고 초기화하는 설정을 담고있다
  - 빈 객체에 대한 정보를 담고 있는 메서드가 `greeter()` 메서드이다

        - `@Bean` 애노테이션을 메서드에 붙이면 **해당 메서드가 생성한 객체를 스프링이 관리하는 빈 객체로 등록한다**
    - `빈(Bean)` : 스프링이 생성하는 객체
    - `@Bean` 애노테이션을 붙인 메서드의 이름은 빈 객체를 구분할 때 사용

- Main 클래스

```
package chap02;

import
org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class Main {
    public static void main(String[] args) {
        AnnotationConfigApplicationContext ctx = new
AnnotationConfigApplicationContext(AppContext.class);
        Greeter g = ctx.getBean("greeter", Greeter.class);

        String msg = g.greet("스프링");
        System.out.println(msg);

        ctx.close();
    }
}
/*
1월 10, 2022 11:35:45 오전
org.springframework.context.support.AbstractApplicationContext
prepareRefresh
INFO: Refreshing
org.springframework.context.annotation.AnnotationConfigApplicationContext@64
38a396: startup date [Mon Jan 10 ...
스프링, 안녕하세요!
1월 10, 2022 11:35:46 오전
org.springframework.context.support.AbstractApplicationContext doClose
INFO: Closing
org.springframework.context.annotation.AnnotationConfigApplicationContext@64
38a396: startup date [Mon Jan 10 11:35:45 KST 2022]; root of context
hierarchy
```

    - `AnnotationConfigApplicationContext` 클래스는 자바 설정에서 정보를 읽어와 빈 객체를 생성하고 관리한다
    - `AnnotationConfigApplicationContext` 객체 생성시 `AppContext` 클래스를 생성자 파라미터로 사용
        - `AnnotationConfigApplicationContext`는 `AppContext`에 정의한 `@Bean` 설정 정보를 읽어와 `Greeter`객체를 생성하고 초기화한다
    - `getBean()` **메서드는 빈 객체를 검색할 때 사용**
        - 첫 파라미터 : `@Bean` 애노테이션의 메서드 이름인 `greeter`
        - 두 번째 파라미터 : 검색할 빈 객체의 타입인 `Greeter.class`
        - `greeter()` **메서드가 생성한 Greeter 객체를 리턴한다**

2. 스프링은 객체 컨테이너

- 스프링의 핵심은 객체를 생성하고 초기화하는 것
  - 이와 관련된 기능은 `ApplicationContext`라는 인터페이스에 정의되어 있다
  - `AnnotationConfigApplicationContext` 클래스는 이 인터페이스를 구현한 클래스 중 하나이다
    - `AnnotationConfigApplicationContext` 클래스 계층 가장 상위에 `BeanFactory` 인터페이스가 있다
- `AnnotationConfigApplicationContext` 클래스는 자바 클래스에서 정보를 읽어와 객체 생성과 초기화를 수행한다
- `BeanFactory` 인터페이스는 객체 생성과 검색에 대한 기능을 정의한다
  - Ex. `getBean()` 메서드가 정의되어 있다
- `ApplicationContext` 인터페이스는 메시지, 프로필/환경 변수 등을 처리 기능을 추가로 정의한다

> - ApplicationContext : 자바 애노테이션을 이용한 클래스로부터 객체 설정 정보를 가져온다
> - GenericXmlApplicationContext : XML로부터 객체 설정 정보를 가져온다
> - GenericGroovyApplicationContext : 그루비 코드를 이용해 설정 정보를 가져온다

- 모든 구현 클래스는 설정 정보로 부터 빈 객체를 생성하고 내부에 보관한다
  - `getBean()` 메서드를 실행하여 빈 객체를 제공한다

- `ApplicationContext` 또는 `BeanFactory`는 빈 객체의 생성, 초기화, 보관, 제거 등을 관리한다
  - 그래서 **컨테이너(Container)**라고도 부른다, **스프링 컨테이너**

> Greeter 타입의 객체를 greeter라는 빈으로 설정했다고 하면
>
> [스프링 컨테이너(ApplicationContext)]
> [greeter] ----------> Greeter 객체
>
> 컨테이너는 greeter 이름과 Greeter 객체를 연결한 정보를 관리한다

2.1 싱글톤(Singleton) 객체

- Main2

```
package chap02;

import
org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class Main2 {

    public static void main(String[] args) {
        AnnotationConfigApplicationContext ctx = new
AnnotationConfigApplicationContext(AppContext.class);
        Greeter g1 = ctx.getBean("greeter", Greeter.class);
        Greeter g2 = ctx.getBean("greeter", Greeter.class);

        System.out.println("(g1==g2) = " + (g1==g2));
        ctx.close();
    }

}
```

```
/*
(g1==g2) = true
```

- `greeter`인 빈 객체를 구해서 각각 `g1`과 `g2` 변수에 할당했다
  - `getBean()` 메서드는 같은 객체를 리턴하여, true가 출력됐다
- 별도 설정을 하지 않으면 **스프링은 한 개의 빈 객체만을 생성한다**
  - 이때 빈 객체는 '싱글톤 범위를 갖는다'고 표현한다
  - 단일 객체를 의미한다
- 스프링은 기본적으로 한 개의 `@Bean` 애노테이션에 대해 한 개의 빈 객체를 생성한다

```
@Bean
    public Greeter greeter() {
        Greeter g = new Greeter();
        g.setFormat("%s, 안녕하세요!");
        return g;
    }

@Bean
    public Greeter greeter1() {
        Greeter g = new Greeter();
        g.setFormat("%s, 하이하이!");
        return g;
    }
```

- 이 경우 `greeter`, `greeter1`에 해당하는 객체가 따로따로 2개의 빈 객체가 생성된다