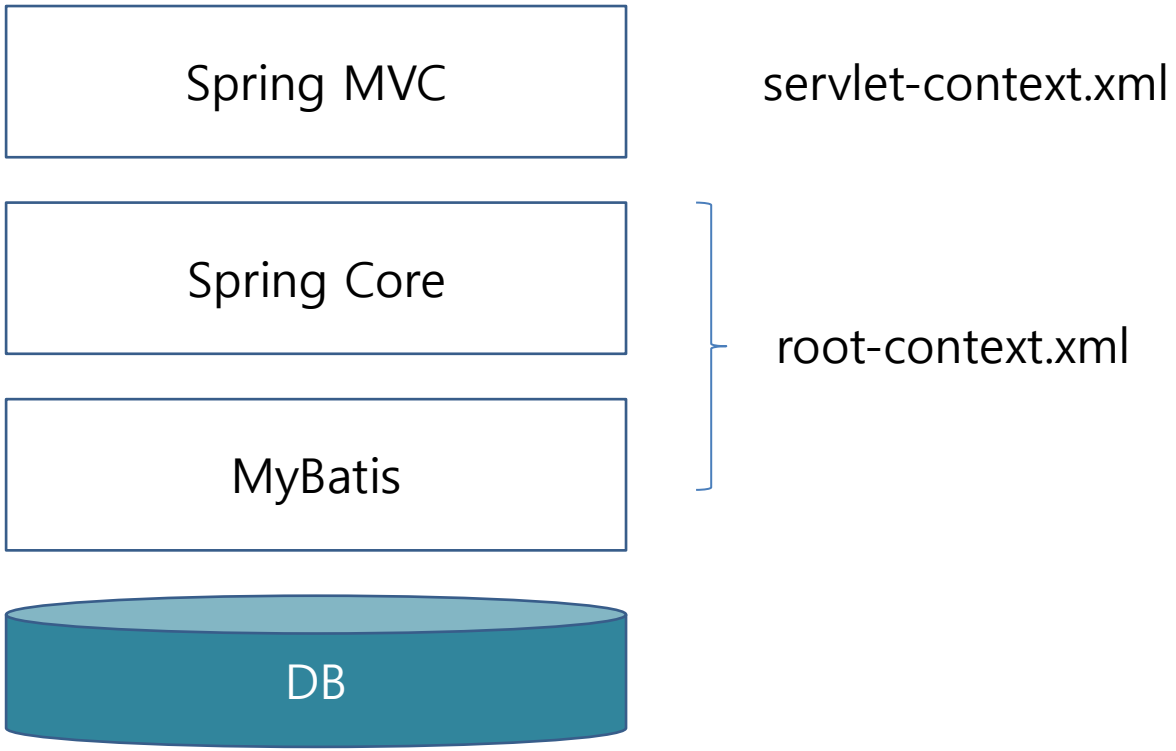
5장. Spring MVC

DI(Dependency Injection)

스프링 MVC

● 스프링 MVC의 기본 구조

스프링 MVC 프로젝트를 구성해서 사용한다는 의미는 내부적으로는 root-context.xml로 사용하는 일반 Java 영역과 servlet-context.xml로 설정하는 Web 관련 영역을 같이 연동해서 구동하게 된다.



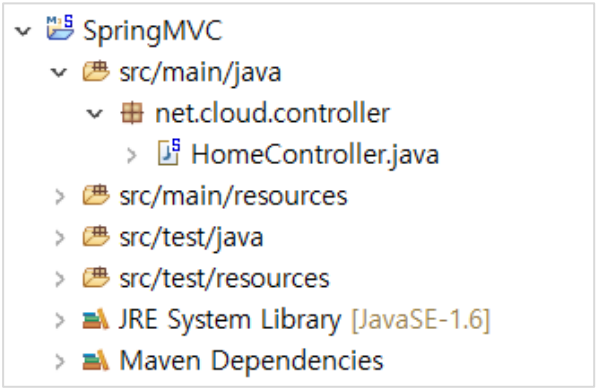
스프링 MVC

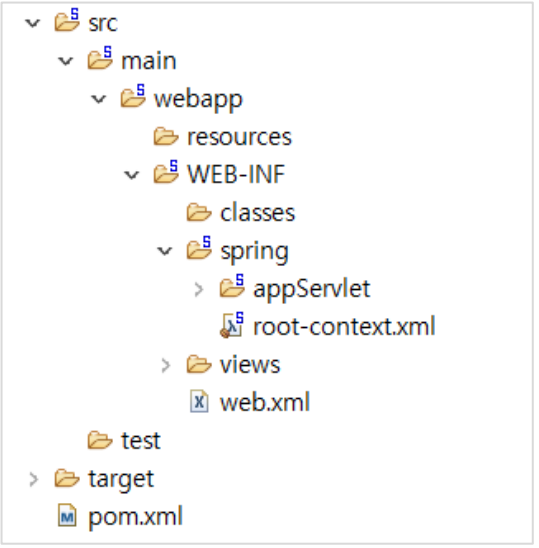
- **스프링 MVC 프로젝트 생성**

  Spring Legacy Project – 프로젝트명: **SpringMVC**

  프로젝트는 Spring MVC Project로 생성

  패키지명은 **net.cloud.controller**로 등록





스프링 MVC

- pom.xml 설정

```
<properties>
    <java-version>1.8</java-version>
    <org.springframework-version>5.2.7.RELEASE</org.springframework-version>
    <org.aspectj-version>1.6.10</org.aspectj-version>
    <org.slf4j-version>1.6.6</org.slf4j-version>
</properties>
```

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>2.5.1</version>
    <configuration>
        <source>${java-version}</source>
        <target>${java-version}</target>
        <compilerArgument>-Xlint:all</compilerArgument>
        <showWarnings>true</showWarnings>
        <showDeprecation>true</showDeprecation>
    </configuration>
</plugin>
```

스프링 MVC

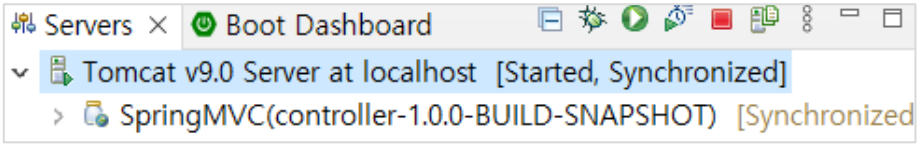
- pom.xml 설정

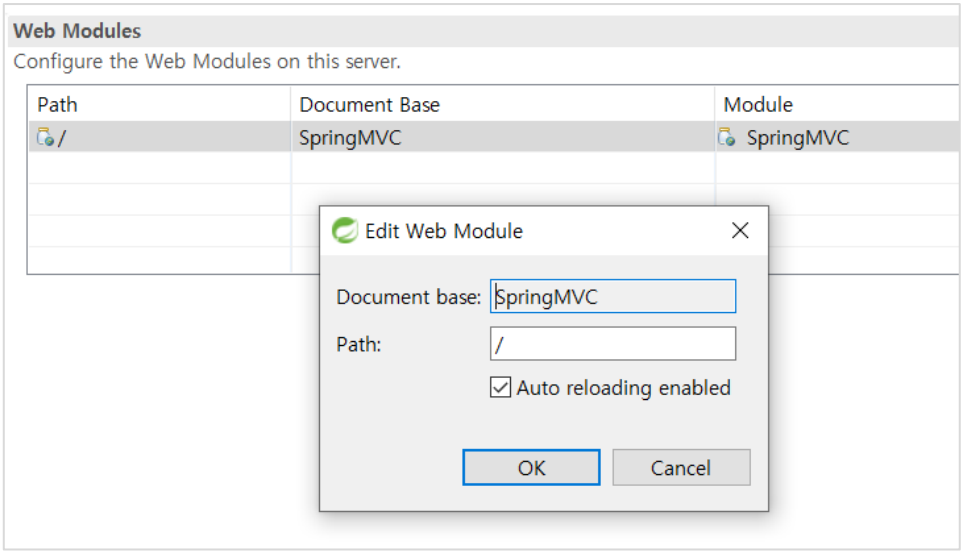
```
<!-- Test -->
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.16</version>
    <scope>provided</scope>
</dependency>
```

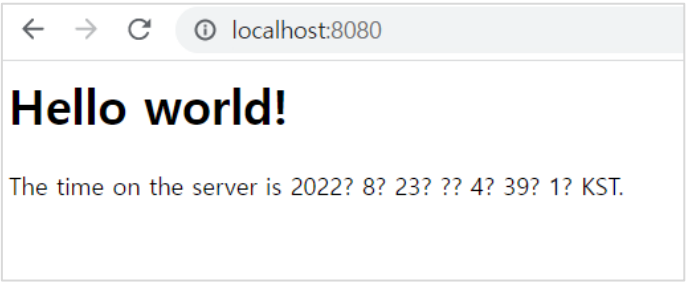
```
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>3.0.1</version>
    <scope>provided</scope>
</dependency>
```

스프링 MVC

- 톰캣 서버 구동







스프링 MVC

- 톰캣 서버 로그(Log)

```
INFO: Initializing Spring root WebApplicationContext
INFO : org.springframework.web.context.ContextLoader - Root WebApplicationContext: initializat
INFO : org.springframework.web.context.ContextLoader - Root WebApplicationContext initialized
8월 23, 2022 4:38:31 오전 org.apache.catalina.util.SessionIdGeneratorBase createSecureRandom
WARNING: [SHA1PRNG] 알고리즘을 사용하여, 세션 ID를 생성하기 위한 SecureRandom 객체를 생성하는데, [114] 밀리초가
8월 23, 2022 4:38:31 오전 org.apache.catalina.core.ApplicationContext log
INFO: Initializing Spring DispatcherServlet 'appServlet'
INFO : org.springframework.web.servlet.DispatcherServlet - Initializing Servlet 'appServlet'
INFO : org.springframework.web.servlet.DispatcherServlet - Completed initialization in 840 ms
8월 23, 2022 4:38:32 오전 org.apache.coyote.AbstractProtocol start
INFO: 프로토콜 핸들러 ["http-nio-8080"]을(를) 시작합니다.
8월 23, 2022 4:38:32 오전 org.apache.catalina.startup.Catalina start
INFO: 서버가 [3042] 밀리초 내에 시작되었습니다.
INFO : net.cloud.controller.HomeController - Welcome home! The client locale is ko_KR.
INFO : net.cloud.controller.HomeController - Welcome home! The client locale is ko_KR.
```

스프링 MVC

- **web.xml**

```
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/spring/root-context.xml</param-value>
</context-param>

<!-- Creates the Spring Container shared by all Servlets and Filters -->
<listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</liste
</listener>
```

```
<servlet>
    <servlet-name>appServlet</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-clas
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
    <servlet-name>appServlet</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>
```

스프링 MVC

- servlet-context.xml

```
<!-- Handles HTTP GET requests for /resources/** by efficiently serving up static re
<resources mapping="/resources/**" location="/resources/" />

<!-- Resolves views selected for rendering by @Controllers to .jsp resources in the
<beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver
    <beans:property name="prefix" value="/WEB-INF/views/" />
    <beans:property name="suffix" value=".jsp" />
</beans:bean>
```

스프링 MVC

- **스프링 MVC의 Controller**
  - HttpServletRequest, HttpServletResponse를 거의 사용할 필요없이 기능구현
  - 다양한 타입의 파라미터 처리, 리턴 타입 사용 가능
  - GET 방식, POST 방식 등 전송 방식에 대한 처리를 어노테이션으로 처리
  - 상속/인터페이스 방식 대신에 어노테이션만으로도 필요한 설정 가능

@Controller

```
package net.cloud.controller;

import org.springframework.stereotype.Controller;

@Controller
public class SampleController {

}
```

스프링 MVC

- **스프링 MVC의 Controller**

  SampleController 클래스는 자동으로 스프링의 객체(Bean)으로 등록되는데 이는 servlet-context.xml에 <context:component-sacn>이라는 태그를 이용해서 해당 패키지에 선언된 클래스를 객체를 생성하고 관리한다.

  servlet-context.xml의 일부

```
    <context:component-scan base-package="net.cloud.controller" />
</beans:beans>
```

@RequestMapping

현재 클래스의 모든 메서드들의 기본적인 URL 경로를 설정한다.

/sample/*은 -> localhost:8080/sample/aaa로 구성

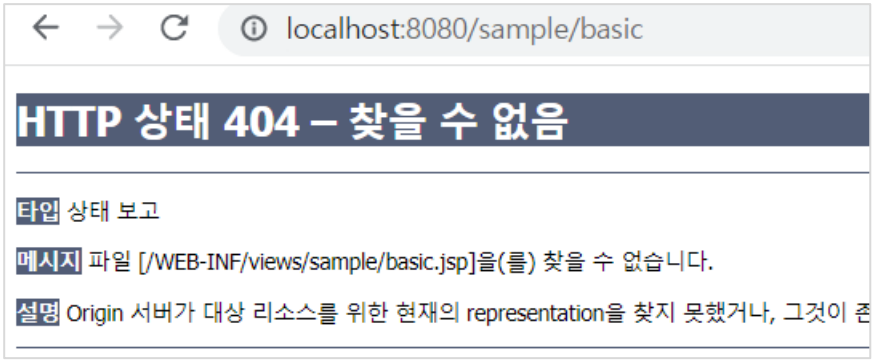
/sample/aaa

/sample/bbb

```
@Log4j
@RequestMapping("/sample/*")
@Controller
public class SampleController {

    //http://localhost:8080/sample/basic - jsp파일이 없어서 404에러 뜸
    @RequestMapping(value="/basic", method=RequestMethod.GET)
    public void basic() {
        log.info("basic get.......");
    }
```

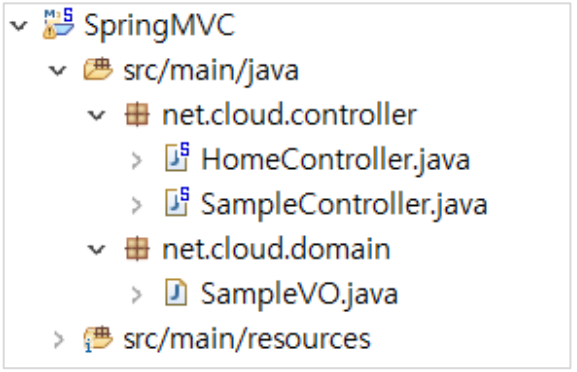
@RequestMapping

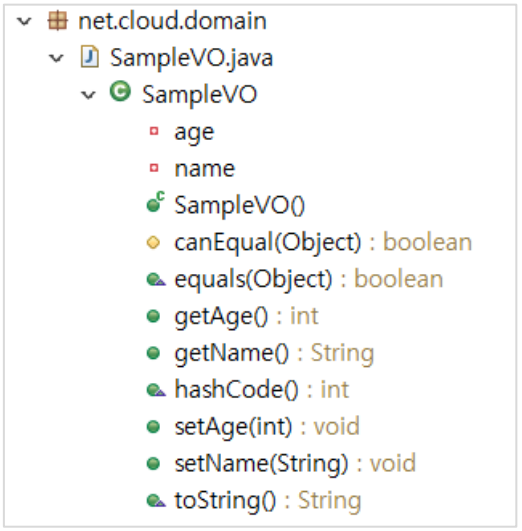
```
    //스프링 4.3이후부터 RequestMapping의 get방식은 GetMapping 사용 가능
    @GetMapping("/basicGet")
    public void basicGet() {
        Log.info("basic get only get.....");
    }
}
```




```
INFO : net.cloud.controller.SampleController - basic get.......
INFO : net.cloud.controller.SampleController - basic get only get.....
```

Controller의 파라미터 수집





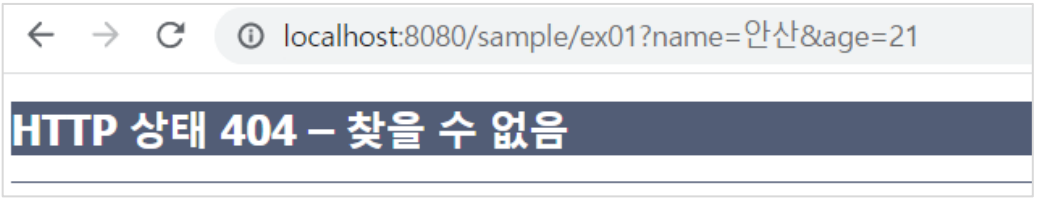

```java
package net.cloud.domain;

import lombok.Data;

@Data
public class SampleVO {

    private String name;
    private int age;
}
```

❖ Controller의 파라미터 수집

```
@GetMapping("/ex01")
public String ex01(SampleVO vo) {
    log.info("" + vo);
    return "ex01";
}
```

SampleController의 메서드가 SampleVO를 파라미터로 사용하게 되면 자동으로 setter 메서드가 작동하면서 파라미터를 수집하게 된다.

/sample/ex01?name=안산&age=21로 요청




```
INFO : net.cloud.controller.SampleController - SampleVO(name=안산, age=0)
INFO : net.cloud.controller.SampleController - SampleVO(name=안산, age=21)
```

@RequestParam 어노테이션

```
//동일한 이름의 파라미터가 여러개 전달될 경우
@GetMapping("/ex02List")
public String ex02List(@RequestParam("ids") ArrayList<String> ids) {
    log.info("ids: " + ids);

    return "ex02List";
}
```

@RequestParam은 사용된 변수의 이름과 전달되는 파라미터의 이름이 다른 경우에 유용하게 사용됨

```
/sample/ex02List?ids=101&ids=102&ids=103
```

```
INFO : net.cloud.controller.SampleController - ids: [101]
INFO : net.cloud.controller.SampleController - ids: [101, 102]
INFO : net.cloud.controller.SampleController - ids: [101, 102, 103]
```

스프링 MVC

@Model

Controller의 메서드를 작성할 때는 Model이라는 타입을 파라미터로 지정할 수 있다. Model 객체는 JSP에 컨트롤러에서 생성된 데이터를 담아서 전달하는 역할을 한다. 메서드의 파라미터에 Model 타입이 지정된 경우에는 스프링은 Model 타입의 객체를 만들어서 메서드에 주입하게 된다.

▶ Servlet에서 MVC 패턴으로 데이터를 전달하는 방식

```
request.setAttribute("boardList", boardList);
RequestDispatcher dispatcher = request.getRequestDispatcher(nextPage);
dispatcher.forward(request, response);
```

▶ Spring에서 MVC 패턴으로 데이터를 전달하는 방식

```
public String getBoardList(Model model) {  //게시글 목록 보기 요청
    List<BoardVO> boardList = boardService.getBoardList();
    model.addAttribute("boardList", boardList);
    return "boardList";
}
```

@ModelAttribute 어노테이션

```
//ex03.jsp로 리턴하기
@GetMapping("/ex03")
public String ex03(SampleVO vo, int page) {
    log.info("vo: " + vo);
    log.info("page: " + page);
    return "ex03";
}
```

ex04.jsp

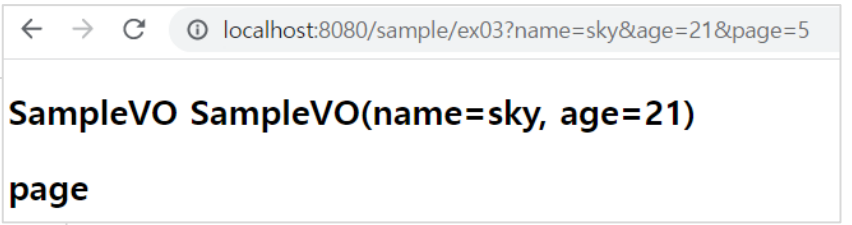
```
<title>view 구현</title>
</head>
<body>
    <h2>SampleVO ${sampleVO}</h2>
    <h2>page ${page}</h2>
</body>
```

@ModelAttribute 어노테이션

```java
//ex03.jsp로 리턴하기
@GetMapping("/ex03")
public String ex03(SampleVO vo, int page) {
    log.info("vo: " + vo);
    log.info("page: " + page);
    return "ex03";
}
```

ex04.jsp

```jsp
<title>view 구현</title>
</head>
<body>
    <h2>SampleVO ${sampleVO}</h2>
    <h2>page ${page}</h2>
</body>
```



page 데이터가 전달되지 않음

@ModelAttribute 어노테이션

강제로 전달받은 파라미터를 Model에 담아서 전달할때 필요한 어노테이션이다.

@ModelAttribute가 걸린 파라미터는 타입에 관계없이 무조건 Model에 담아서 전달되므로 파라미터로 전달된 데이터를 다시 화면에서 사용해야할 경우에 유용하다.

```
@GetMapping("/ex03")
public String ex03(SampleVO vo, @ModelAttribute("page") int page) {
    log.info("vo: " + vo);
    log.info("page: " + page);

    return "ex03";
}
```