[한국ICT인재개발원] 스프링 프레임워크

4. 스프링 MVC 살펴보기

강의자 소개 : 채종훈 강사

前) 광고데이터 분석 1년

前) IT강의 경력 2년 6개월

前) 머신러닝을 활용한 데이터 분석 프로젝트반 운영 1년

前) 리그오브 레전드 데이터 분석 등...

現) 국비반 강의 진행중

오늘 공부할 것들

1

스프링 MVC 구조 살펴보기

프로젝트 생성 및 세팅
서버 설정과 초기 설정 상태 확인

2

컨트롤러 이해하기

어노테이션을 활용한 url배정
리턴 자료형에 따른 처리 방식 이해하기

3

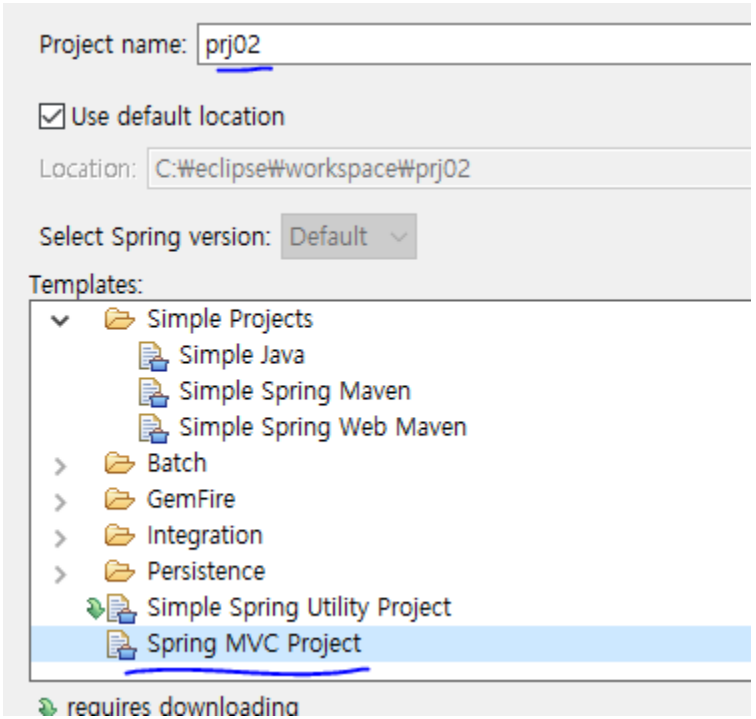
파일 업로드와 예외처리

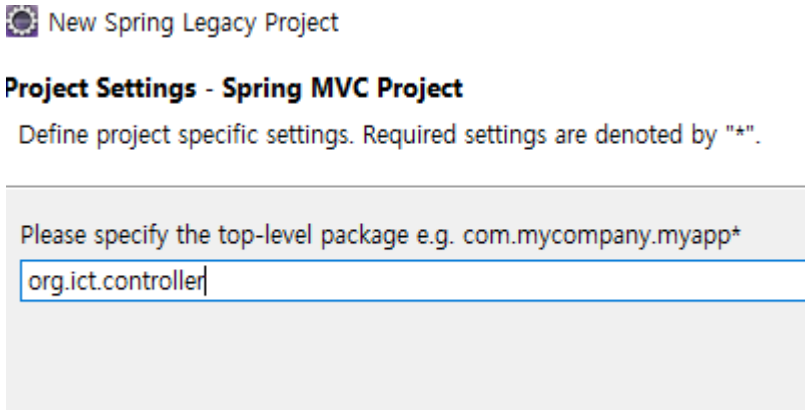
파일 업로드 맛보기
예외 페이지 설정하기

4

실제 문제 풀어보기

프로젝트 세팅





1. Spring Legacy Project로 Spring MVC Project를 생성합니다.
2. 계층구조는 org.ict.controller로 설정합니다.

프로젝트 세팅


```
11          <java-version>1.8</java-version>
12          <org.springframework-version>5.0.7.RELEASE<
13          <org.aspectj-version>1.6.10</org.aspectj-ve
14          <org.slf4j-version>1.6.6</org.slf4j-versior
15      </properties>
16      <dependencies>
17          <!-- Spring -->
```

```
93          <!-- Servlet -->
94          <dependency>
95              <groupId>javax.servlet</groupI
96              <artifactId>javax.servlet-api<
97              <version>3.1.0</version>
98              <scope>provided</scope>
99          </dependency>
100         <dependency>
```

Pom.xml을 수정하되, 라인을 확인하고 변경하시면 편합니다.

1. **스프링 버전은 5.0.7버전으로, 자바버전은 1.8버전으로 합니다.**
2. **Servlet은 3.1.0버전으로, artifactId를 javax.servlet-api로 변경합니다.**

프로젝트 세팅


```
137             <groupId>org.apache.maven.plugins</
138             <artifactId>maven-compiler-plugin</
139             <version>2.5.1</version>
140             <configuration>
141                 <source>1.8</source>
142                 <target>1.8</target>
143                 <compilerArgument>-Xlint:all</c
144                 <showWarnings>true</showWarning
145                 <showDeprecation>true</showDepr
146             </configuration>
```

141, 142번라인까지 전부 1.6 -> 1.8로 변경합니다.

프로젝트 세팅


```
<!-- https://mvnrepository.com/artifact/org
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-test</artifactId>
    <version>5.0.7.RELEASE</version>
    <scope>test</scope>
</dependency>
```

```
<!-- https://mvnrepository.com/artifact/
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.18</version>
    <scope>provided</scope>
</dependency>
```

Spring-test, lombok을 설정합니다.

1. **Spring-test는 5.0.7버전으로, 스프링 버전과 동일하게 둡니다.**
2. **Lombok은 1.18.18버전을 둡니다.**

프로젝트 세팅


```
94          <!-- Test -->
95          <dependency>
96              <groupId>junit</groupId>
97              <artifactId>junit</artifactId>
98              <version>4.12</version>
99              <scope>test</scope>
100         </dependency>
```

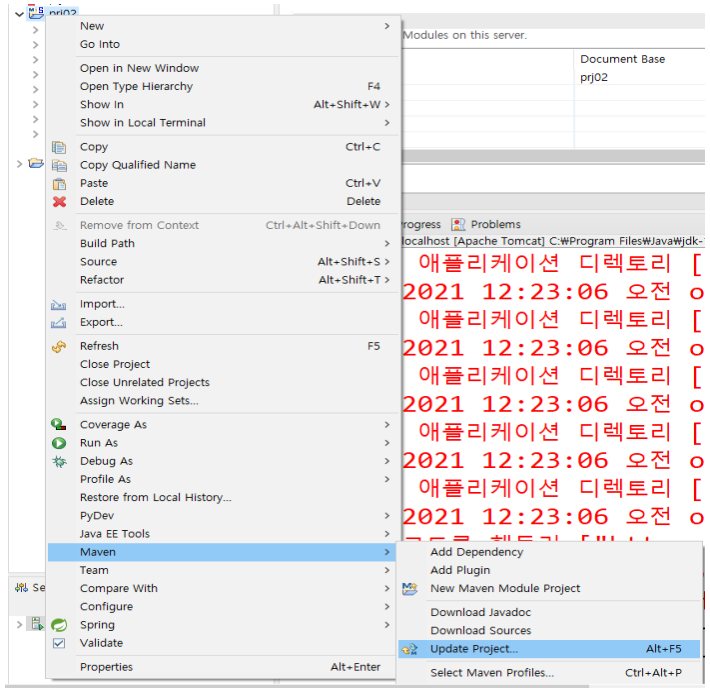
```
63              <artifactId>log4j</artifactId>
64              <version>1.2.15</version>
65              <exclusions>
66                  <exclusion>
67                      <groupId>javax.mail</groupId>
68                      <artifactId>mail</artifactId>
69                  </exclusion>
70                  <exclusion>
71                      <groupId>javax.jms</groupId>
72                      <artifactId>jms</artifactId>
73                  </exclusion>
74                  <exclusion>
75                      <groupId>com.sun.jdmk</groupId>
76                      <artifactId>jmxtools</artifactId>
77                  </exclusion>
78                  <exclusion>
79                      <groupId>com.sun.jmx</groupId>
80                      <artifactId>jmxri</artifactId>
81                  </exclusion>
82              </exclusions>
```

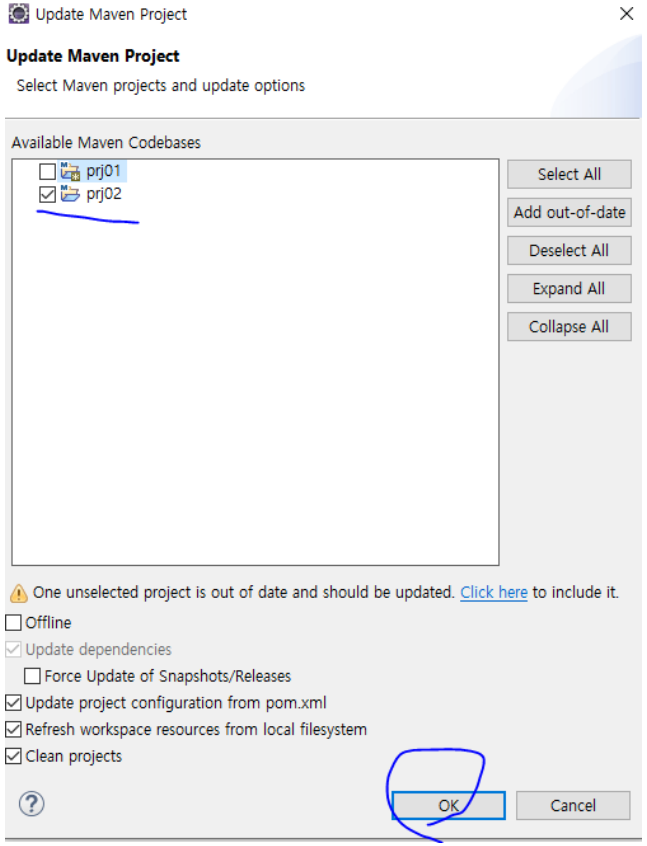
```
61          <dependency>
62              <groupId>log4j</groupId>
63              <artifactId>log4j</artifactId>
64              <version>1.2.17</version>
65              <scope>runtime</scope>
66          </dependency>
67
```

로깅 관련 처리입니다.

1. Junit4는 4.12버전으로 변경합니다.
2. Log4j는 1.2.17로, 그리고 <exclusions>태그를 전부 삭제합니다.

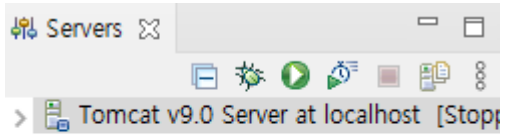
프로젝트 세팅

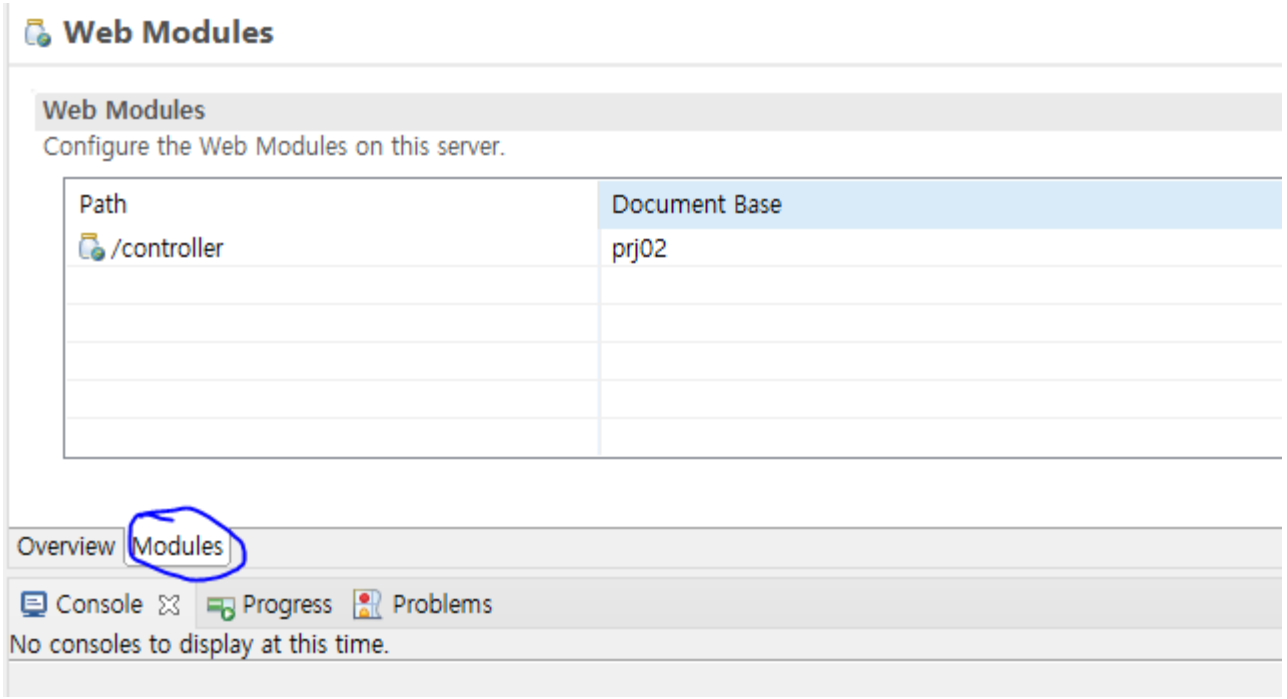




Prj02 프로젝트를 우클릭 -> Maven -> Update Project를 선택한 다음 Prj02만 체크된 상태에서 ok를 눌러 자바 버전을 1.8로 올립니다.

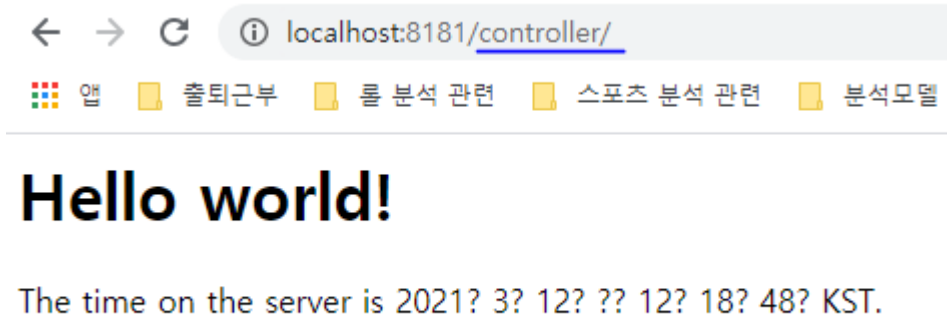
서버와 프로젝트





하단의 서버를 더블클릭한 다음 나오는 창의 Modules를 보면
위와 같이 Document Base에 어떤 프로젝트가 서버의 타겟인지 나옵니다.
서버를 가동시켰을때 어떤 프로젝트 파일을 실행할지 설정할 수 있습니다.

서버와 프로젝트

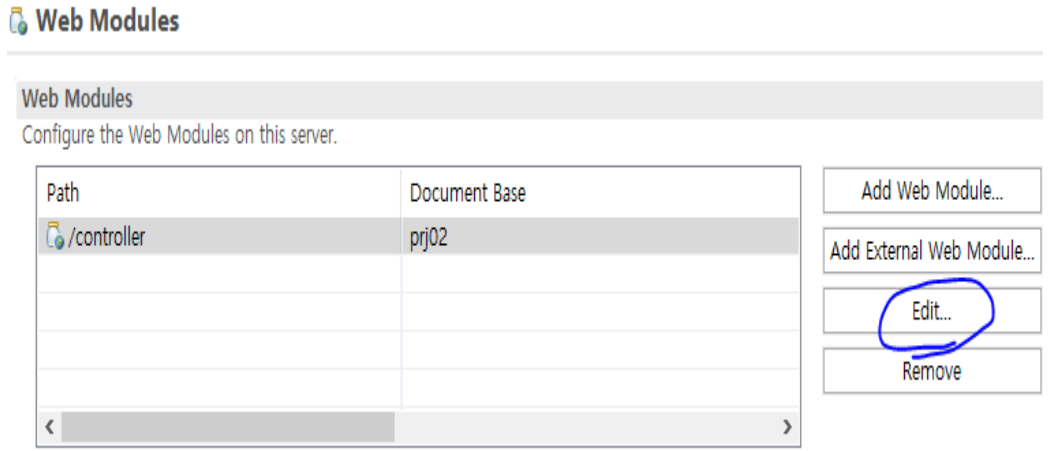


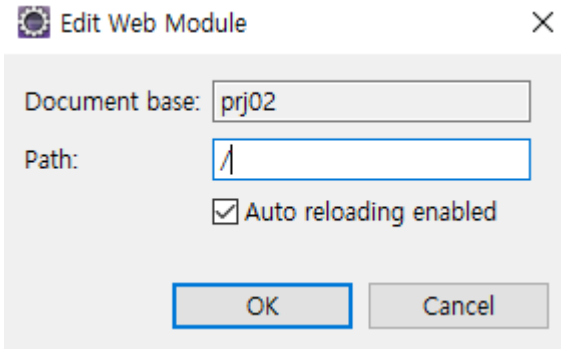
먼저 기본 주소는 localhost:8181/ 까지입니다.

Path 항목은 기본주소 뒤에 붙을 추가 주소를 의미하며

현재 /controller로 기술되어 있어서 메인페이지가
localhost:8181/controller라는 주소를 가집니다.

서버와 프로젝트

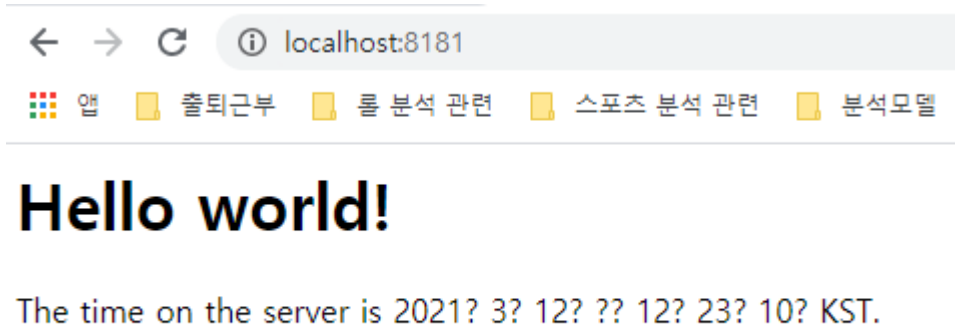




프로젝트 설정시 기본 경로 이외에 다른 경로를 추가로 붙이지 않는것이 유리하기 때문에 위와 같이 Edit를 눌러 기본 주소를 /로 설정합니다.

추가로 prj02이외의 다른 프로젝트가 있다면 Remove를 눌러 제거합니다.

서버와 프로젝트



수정 후엔 보시는 바와 같이 기본 주소만으로 메인주소에 접속이 가능합니다.

초기 상태 확인


```
6    <!-- The definition of the Root Spring Container sh
7    <context-param>
8        <param-name>contextConfigLocation</param-name>
9        <param-value>/WEB-INF/spring/root-context.xml</
10   </context-param>
11
```

프로그램이 시작되면 web.xml에서 root-context.xml을 읽으며 시작합니다.
위와 같이 web.xml에 우선적으로 읽어오는 설정이 되어 있기 때문입니다.

초기 상태 확인


```
17      <!-- Processes application requests -->
18      <servlet>
19          <servlet-name>appServlet</servlet-name>
20          <servlet-class>org.springframework.web.servlet.DispatcherServle
21          <init-param>
22              <param-name>contextConfigLocation</param-name>
23              <param-value>/WEB-INF/spring/appServlet/servlet-context.xml
24          </init-param>
25          <load-on-startup>1</load-on-startup>
26      </servlet>
```

다음 라인인 17번라인을 보면 servlet-context.xml을 읽어오도록 되어있으며 이 파일은 기본적으로 view쪽을 담당합니다.

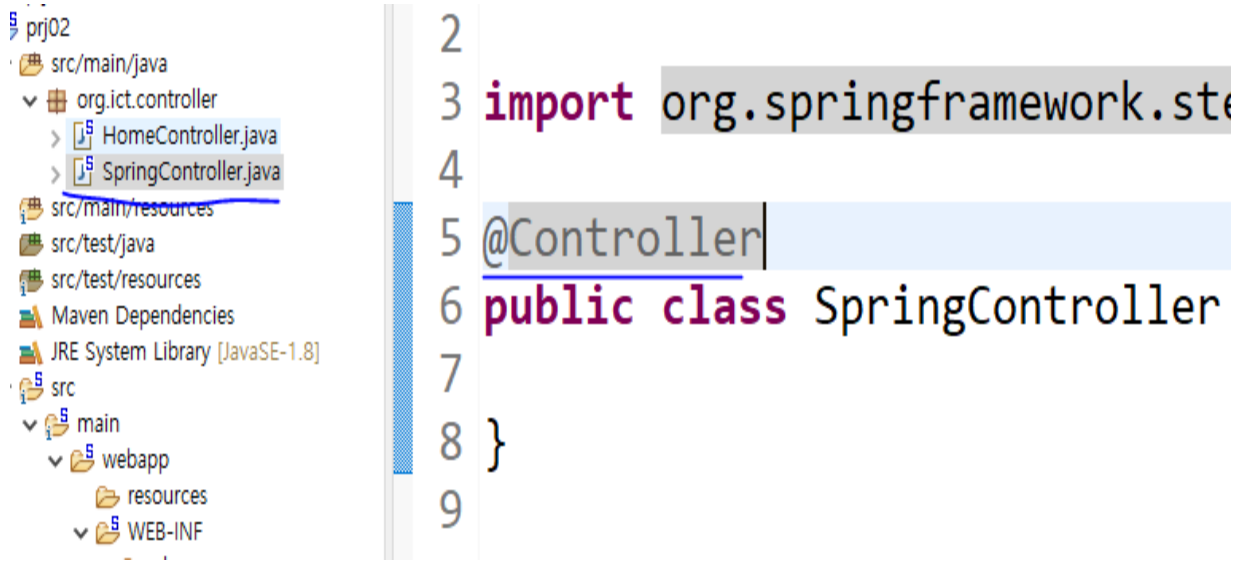
초기 상태 확인


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

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

<context:component-scan base-package="org.ict.controller" />
```

Servlet-context.xml파일을 다음으로 보면 위와 같이
정적 자원의 위치는 /resources/ 폴더에서 읽어오도록

View파일은 /WEB-INF/views/ 폴더 하위에 있는 .jsp로
끝나는 파일을 읽도록 설정되어있습니다.

컨트롤러 관련 어노테이션 이해하기



Src/main/java의 org.ict.controller에 SpringController.java를 생성합니다.

생성 후 이 클래스파일에 @Controller 어노테이션을 달면
Servlet-context.xml 가장 하단의 컴포넌트 스캔이 있어서
이 파일을 컨트롤러로 인식하게 됩니다.

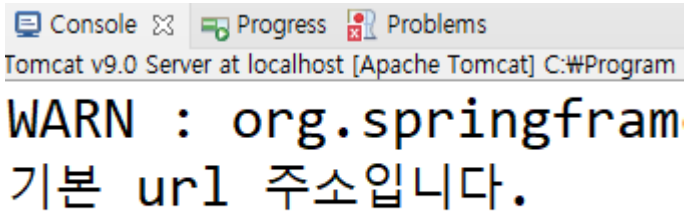
컨트롤러 관련 어노테이션 이해하기


```
7 @Controller
8 @RequestMapping("/spring/*")
9 public class SpringController {
0
1⊖    @RequestMapping("")
2     public void basic() {
3         System.out.println("기본 url 주소입니다.");
4     }
5 }
6
```

다음은 @RequestMapping() 어노테이션입니다.
이 어노테이션은 컨트롤러에 붙으면 하위 전체 메서드 이전에 공통주소를 부여합니다.

메서드에 붙으면 기본url 이후 주소를 정의하며 접속시 작동하게 합니다.
가령 위 basic() 메서드 실행주소는 localhost:8181/spring/ 입니다.

컨트롤러 관련 어노테이션 이해하기



기본 주소 + /spring/ 조합 접속의 결과로
Basic()이 호출되는것을 콘솔창에서 확인할 수 있습니다.

컨트롤러 관련 어노테이션 이해하기


```
16          System.out.println("기본 url 주소입니다.");
17      }
18
19      @RequestMapping(value = "/base",
20              method = {RequestMethod.GET, RequestMethod.POST})
21      public void baseGet() {
22
23          System.out.println("base get");
24      }
```

다음으로 @RequestMapping에 추가로 넣을 수 있는 정보는

Value = 주소패턴(문자열만 입력시 주소패턴으로 자동 간주)
Method = get, post방식 구분을 위해 넣는 파라미터
위와 같이 array로 RequestMethod를 묶어서 GET, POST를 전부 처리 가능.

컨트롤러 관련 어노테이션 이해하기


```java
@GetMapping("/baseGet")
public void baseOnlyGet() {

    System.out.println("base only get");
}
```

그리고 Get, Post 지정이 귀찮다면 위와같이
get방식만 전제로 하는 @GetMapping() 이나
Post만을 전제로 하는 @PostMapping()을 쓸 수도 있습니다.