1. Spring 설정

1. Spring Tools 4
2. Spring Tools 4 for Eclipse 설치
3. `java -jar spring-tool-suite-4-4.XX.XXXXX.jar`
4. *SpringToolSuite4.ini* 파일 수정

   ```
   -vm
   C:\Program Files\Java\jdk-버전\bin
   -vmargs
   ```

5. *SpringToolSuite4.exe* 실행 후 작업폴더 설정
6. Help => Eclipse Marketplace => sts 검색 => Spring Tools 3 Add-On for Spring Tools 4 설치
7. Window => preference => General => Workspace => UTF-8
8. Window => preference => General => WebBrowser=> chrome
9. File => new => other => Spring Legacy => Spring MVC Proejct 템플릿 선택 & 프로젝트명
10. ProjectFacet => java 11 dynamic web module 4.0
11. Run/Debug => Console => 버퍼사이즈 1,000,000
12. 톰캣서버 생성 및 프로젝트 연동

2. Maven (새로 문서로 빠지고 보강될 예정)

- 컴파일과 동시에 패키징까지 해주는 빌드 도구이다.
- 라이브러리(`.jar`)를 자동으로 관리해주는 도구이다.
  - 라이브러리의 의존성을 고려하여 다른 라이브러리들도 mvn repository에서 다운로드 한다.

2.1 POM.xml

Preference > User Settings > Local Repository 에 저장한다.

2.1.1 헤더

```
<modelVersion>4.0.0</modelVersion>
<!-- 회사명 -->
<groupId>com.kosmo</groupId>
<!-- 어플리케이션명 -->
<artifactId>springapp</artifactId>
<name>SpringProj</name>
<packaging>war</packaging>
<!-- 어플리케이션명 -->
<version>1.0.0-BUILD-SNAPSHOT</version>
<!-- dependencies에서 적용될 버전 정의 -->
<properties>
    <!-- JDK 버전 -->
    <java-version>1.11</java-version>
    <!-- 스프링 프레임워크 버전 -->
    <org.springframework-version>4.3.30.RELEASE</org.springframework-version>
```

```
    <!-- 그대로 써도 무방 -->
    <org.aspectj-version>1.6.10</org.aspectj-version>
    <org.slf4j-version>1.6.6</org.slf4j-version>
</properties>
```

2.1.2 dependencies

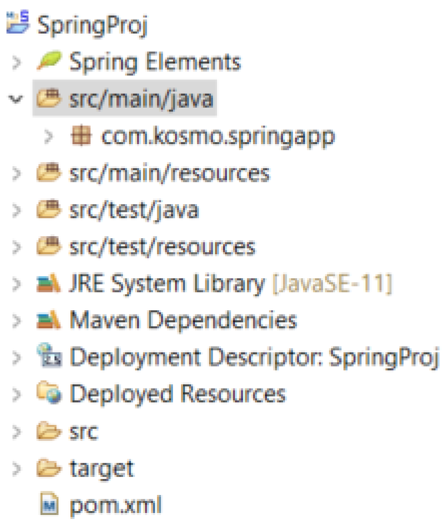
```
<dependencies>
    <!-- Spring -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>${org.springframework-version}</version>
        <exclusions>
            <!-- Exclude Commons Logging in favor of SLF4j -->
            <exclusion>
                <groupId>commons-logging</groupId>
                <artifactId>commons-logging</artifactId>
            </exclusion>
        </exclusions>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>${org.springframework-version}</version>
    </dependency>

    <!-- AspectJ -->
    <dependency>
        <groupId>org.aspectj</groupId>
        <artifactId>aspectjrt</artifactId>
        <version>${org.aspectj-version}</version>
    </dependency>

    <!-- Logging -->
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-api</artifactId>
        <version>${org.slf4j-version}</version>
    </dependency>
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>jcl-over-slf4j</artifactId>
        <version>${org.slf4j-version}</version>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-log4j12</artifactId>
        <version>${org.slf4j-version}</version>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>log4j</groupId>
        <artifactId>log4j</artifactId>
        <version>1.2.15</version>
        <exclusions>
```

```xml
                <exclusion>
                    <groupId>javax.mail</groupId>
                    <artifactId>mail</artifactId>
                </exclusion>
                <exclusion>
                    <groupId>javax.jms</groupId>
                    <artifactId>jms</artifactId>
                </exclusion>
                <exclusion>
                    <groupId>com.sun.jdmk</groupId>
                    <artifactId>jmxtools</artifactId>
                </exclusion>
                <exclusion>
                    <groupId>com.sun.jmx</groupId>
                    <artifactId>jmxri</artifactId>
                </exclusion>
            </exclusions>
            <scope>runtime</scope>
        </dependency>

        <!-- @Inject -->
        <dependency>
            <groupId>javax.inject</groupId>
            <artifactId>javax.inject</artifactId>
            <version>1</version>
        </dependency>

        <!-- Servlet -->
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>servlet-api</artifactId>
            <version>2.5</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>javax.servlet.jsp</groupId>
            <artifactId>jsp-api</artifactId>
            <version>2.1</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>jstl</artifactId>
            <version>1.2</version>
        </dependency>

        <!-- Test -->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.7</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
```

2.1.3 build

```
<build>
    <plugins>
        <plugin>
            <artifactId>maven-eclipse-plugin</artifactId>
            <version>2.9</version>
            <configuration>
                <additionalProjectnatures>

 <projectnature>org.springframework.ide.eclipse.core.springnature</projectnature
>
                </additionalProjectnatures>
                <additionalBuildcommands>

 <buildcommand>org.springframework.ide.eclipse.core.springbuilder</buildcommand>
                </additionalBuildcommands>
                <downloadSources>true</downloadSources>
                <downloadJavadocs>true</downloadJavadocs>
            </configuration>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>2.5.1</version>
            <configuration>
                <source>1.6</source>
                <target>1.6</target>
                <compilerArgument>-Xlint:all</compilerArgument>
                <showWarnings>true</showWarnings>
                <showDeprecation>true</showDeprecation>
            </configuration>
        </plugin>
        <plugin>
            <groupId>org.codehaus.mojo</groupId>
            <artifactId>exec-maven-plugin</artifactId>
            <version>1.2.1</version>
            <configuration>
                <mainClass>org.test.int1.Main</mainClass>
            </configuration>
        </plugin>
    </plugins>
</build>
```

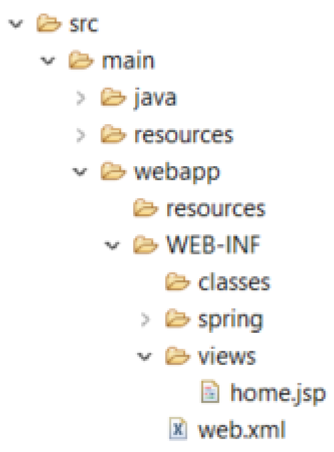
3. 스프링 파일 구조


SpringProj
Spring Elements
src/main/java
com.kosmo.springapp
src/main/resources
src/test/java
src/test/resources
JRE System Library [JavaSE-11]
Maven Dependencies
Deployment Descriptor: SpringProj
Deployed Resources
src
target
pom.xml

- `src/main/java`
    - *com.kosmo.springapp*
      탑레벨 패키지이며 모든 자바 파일은 여기에 저장하게 된다.
- HomeController.java
  index.jsp 뷰 페이지로 이동시켜준다.

```
@Controller
public class HomeController {

    private static final Logger logger =
LoggerFactory.getLogger(HomeController.class);

    @RequestMapping(value = "/index.do", method = RequestMethod.GET)
    public String home(Locale locale, Model model) {
        logger.info("Welcome home! The client locale is {}.", locale);

        Date date = new Date();
        DateFormat dateFormat =
DateFormat.getDateTimeInstance(DateFormat.LONG, DateFormat.LONG, locale);

        String formattedDate = dateFormat.format(date);
        model.addAttribute("serverTime", formattedDate );
        return "index";
    }
}
```

- `src/main/resources`
  컨트롤러에서 사용할 데이터베이스 셋팅 및 쿼리 등을 저장
- `src/test/java`
    - *com.kosmo.springapp*
      탑레벨 패키지이며 모든 유닛테스트 파일은 여기에 저장


```
src
  main
    java
    resources
    webapp
      resources
      WEB-INF
        classes
        spring
        views
          home.jsp
        web.xml
```

- `src/webapp/WEB-INF`
  - `web.xml`
    컨텍스트(컨테이너) 파라미터 설정
    순수 서블릿API로 구현했던 방식과 달리 디스패처 서블릿만 등록한다.

```
<!-- 컨텍스트(~컨테이너) 초기화 파라미터 -->
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/spring/root-context.xml</param-value>
</context-param>


<!-- 컨테이너 생성 -->
<listener>
    <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-
class>
</listener>


<!-- *.do URL 요청을 처리하는 디스패처서블릿 생성 -->
<servlet>
    <servlet-name>appServlet</servlet-name>
    <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <!-- 디스패처 초기화 파라미터 (빈 설정) -->
    <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>*.do</url-pattern>
</servlet-mapping>


<!-- index.do를 시작요청으로 설정 -->
<welcome-file-list>
    <welcome-file>index.do</welcome-file>
</welcome-file-list>
```

```
<!-- *.kosmo URL 요청을 처리하는 디스패처서블릿 생성 -->
<servlet>
    <servlet-name>myServlet</servlet-name>
    <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>myServlet</servlet-name>
    <url-pattern>*.kosmo</url-pattern>
</servlet-mapping>


<!-- 인코딩타입 -->
<filter>
    <filter-name>CharacterEncoding8888888888</filter-name>
    <filter-
class>org.springframework.web.filter.CharacterEncodingFilter</filter-
class>
    <init-param>
        <param-name>encoding</param-name>
        <param-value>UTF-8</param-value>
    </init-param>
</filter>
<filter-mapping>
    <filter-name>CharacterEncoding8888888888</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

- *spring/appServlet/* `servlet-context.xml`
  컨텍스트에서 정의한 디스패처 서블릿에 대한 설정
  디스패처에 종속된 커넥션, 자바빈, 등을 생성

```
<!-- Enables the Spring MVC @Controller programming model -->
<annotation-driven />


<!--
    핸들러매핑
        ${mapping} 패턴의 URL로 접근 시 ${location} 내의 파일 위치로 접근
-->
<!--
<resources mapping="/resources/**" location="/resources/" />
-->

<!--
    뷰 리졸버 빈으로 등록
    오더는 3으로 두는 이유를 미리 말하자면...
        1 : 스프링에서 제공하는 API 사용해서 다운로드 구현
        2 : tiles3
        3 : 뷰리졸버
-->
<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>


<!--
    ${base-package}의 하위 패키지에 있는
    @Component, @Controller, @Service, @Repository 붙은 클래스들을 객체로 생성 후 빈으로 등록
    그러므로, 설정파일에 직접 등록할 필요가 없어졌다.

    @Controller : 사용자 요청을 처리하는 클래스
    @Service    : 서비스 역할을 하는 클래스
    @Repository : DAO
    @Component  : 기타클래스
-->
<context:component-scan base-package="com.kosmo.springapp" />
```

- *spring/root-context.xml*
  여러 디스패처가 공유하는 커넥션풀 등을 빈으로 등록

소스파일