<http://huskdoll.tistory.com/3>

## [Gradle 설치 (1.11버전 기준)](http://huskdoll.tistory.com/3)

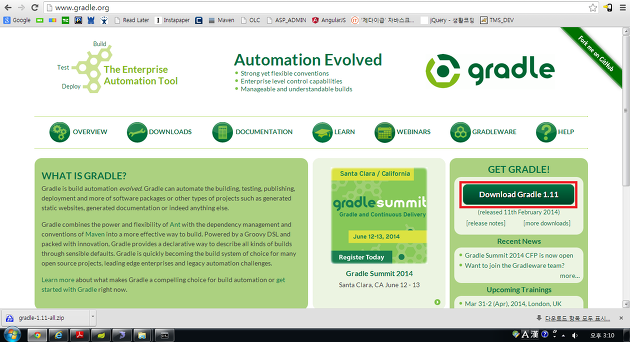
[Spring](http://huskdoll.tistory.com/category/Spring) 2014.03.07 10:36

|  |
| --- |
|  |

\* 결론부터 말하면 설치가 간단하다.

download 받아서 압축 풀고 해당 경로를 환경변수에 등록해주면 끝.

1. <http://www.gradle.org/>  => Download Gradle 1.11



2. 다운받은 파일 (gradle-1.11-all.zip) 을 설치 하려는 위치에 놓고 압축을 풀어준다.

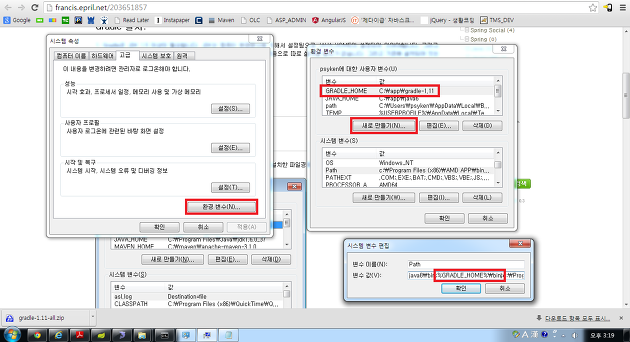


3. 설정한 gradle위치를 환경 변수에 등록 하여 준다.

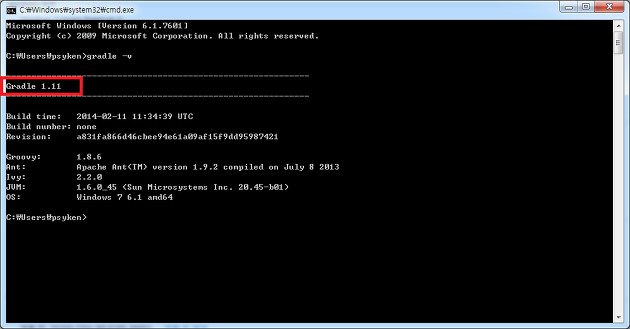
GRADLE\_HOME : C:\app\gradle-1.11

Path : %GRADLE\_HOME%\bin;

\* 그런데 저는 실제적으로 잘 안되서 Path쪽에 C:\app\gradle-1.11\bin; 를 입력해줬습니다. (전체 경로)



4. cmd창 열어서 **gradle -v** 명령어를 실행할 경우 해당 설치 버전의 정보가 보이면 성공



## [1탄. gradle 로 이클립스 Spring MVC 프로젝트 생성 (1.11버전 기준)](http://huskdoll.tistory.com/8)

[Spring](http://huskdoll.tistory.com/category/Spring) 2014.03.07 14:54

|  |
| --- |
|  |

**\* 댓글을 달아주신 분이 Gradle 2.0 이후 버전부터는 설정이 변경되어 정삭적으로 프로젝트가 생성 되지 않는다고 알려주셨습니다.**

**해당 포스팅은 1.11 버전을 기준으로 작성 되었습니다.**

**2.0 이후 버전은 추가로 작성 하려고 예정중입니다.**

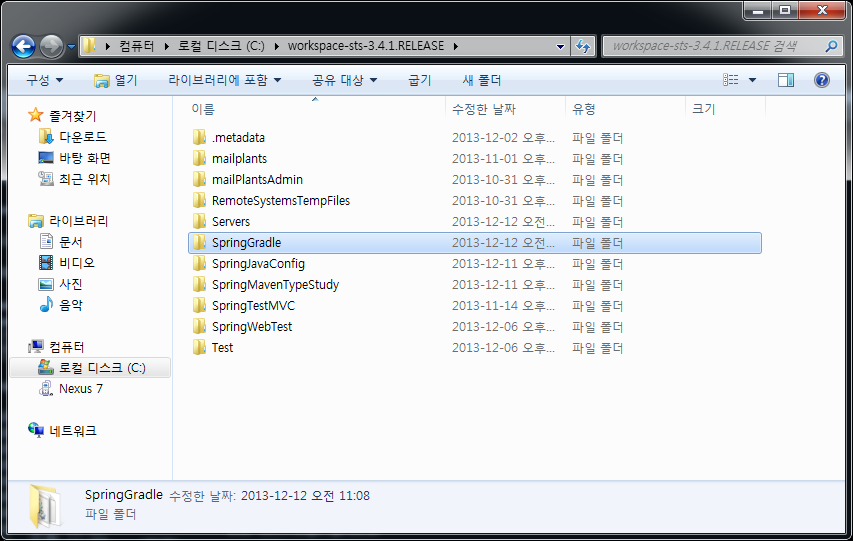
**감사합니다.**

gradle 설치 하셔야 가능 합니다.

설치는 <http://huskdoll.tistory.com/3> 를 참고하여 주세요.

**1. 프로젝트 폴더 생성**

프로젝트 명으로 폴더를 하나 생성합니다. (저 같은 경우는 이클립스 workspce 밑에 SpringGradle 로 생성 하였습니다.)

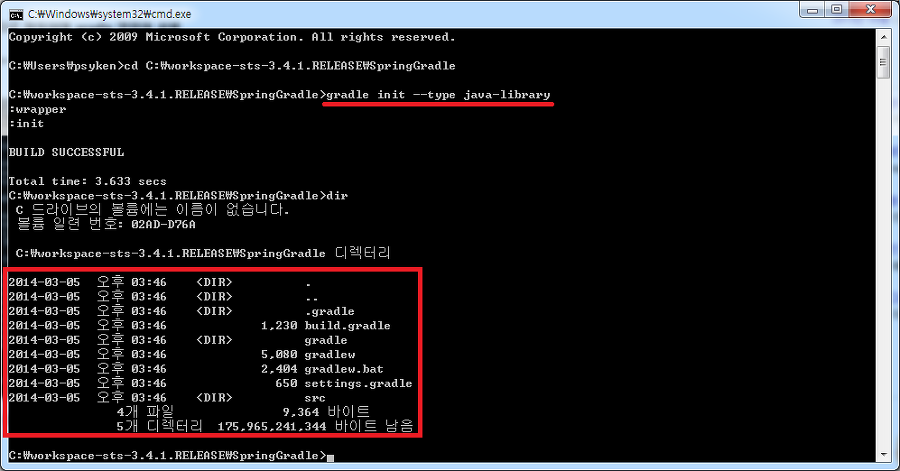


**2. CMD 창에서 생성한 폴더로 이동하여 gradle 명령어 실행**

cmd 창을 열고 위에서 생성한 폴더로 이동합니다.

그리고 아래와 같이 명령어를 입력 하세요.

gradle init --type java-library



**3. build.gradle 내용 수정**

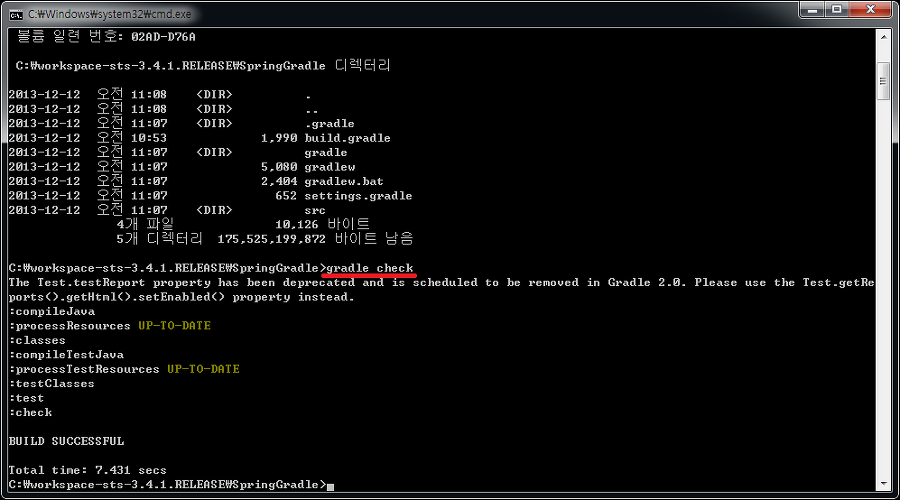
2번을 실행하면 해당 프로젝트 폴더에 build.gradle 이 생성 되는데 해당 파일 열어서 아래 내용 작성 후 저장 합니다.

\* build.gradle 파일 작성 시 해당 사이트 참고 하였습니다. <http://arawn.github.io/blog/2012/08/28/gradle-springmvc-project/>

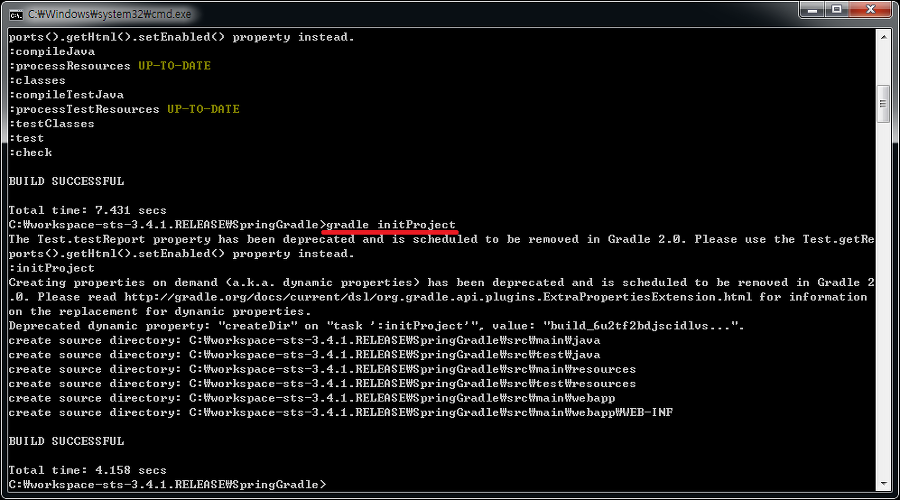
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83 | /\*   \* This build file was auto generated by running the Gradle 'init' task   \* by 'psyken' at '14. 3. 5 오후 3:46' with Gradle 1.11   \*   \* This generated file contains a sample Java project to get you started.   \* For more details take a look at the Java Quickstart chapter in the Gradle   \* user guide available at <http://gradle.org/docs/1.11/userguide/tutorial_java_projects.html>   \*/    // Apply the java plugin to add support for Java  apply plugin: 'java'  apply plugin: 'war'  apply plugin: 'eclipse'  apply plugin: 'eclipse-wtp'    // JAVA Version 1.6  sourceCompatibility = 1.6  // 개발한 애플리케이션 버전  version = '1.0'    // 메이븐 Central 저장소 사용  repositories {      mavenCentral()  }    // dependency 버전 정보  def version = [  spring: '3.2.2.RELEASE'  , junit: '4.11'  , servletApi: '3.0.1'  , jstl: '1.2'  , slf4j: '1.7.6'  , mockito: '1.9.0'  , cglib: '2.2.2'  , logback: '1.0.6'  ]    // In this section you declare the dependencies for your production and test code  // 의존성 설정  dependencies {      compile "org.springframework:spring-webmvc:${version.spring}",            "cglib:cglib-nodep:${version.cglib}",            "ch.qos.logback:logback-classic:${version.logback}",                      "org.slf4j:slf4j-api:${version.slf4j}"      providedCompile "javax.servlet:javax.servlet-api:${version.servletApi}"      testCompile "org.springframework:spring-test:${version.spring}",                "junit:junit:${version.junit}",                "org.mockito:mockito-core:${version.mockito}"      runtime "javax.servlet:jstl:${version.jstl}"  }    // logback(slf4j)를 사용하기 때문에 모든 의존성에서 commons-logging는 제외  [configurations.runtime, configurations.default]\*.exclude(module: 'commons-logging')    // JAVA 컴파일시 인코딩 설정  [compileJava, compileTestJava]\*.options\*.encoding = 'UTF-8'    // TEST 설정  test {      jvmArgs = ['-ea', '-Xmx256m']      logging.captureStandardOutput(LogLevel.INFO)      //testReport = false      reports.html.enabled = false  }    // 프로젝트 초기화  // 1. java source directory 생성 : src/main/java, src/test/java  // 2. resource directory 생성    : src/main/resource, src/test/resource  // 3. web source directory 생성  : src/main/webapp, src/main/webapp/WEB-INF  task initProject(description: 'initialize project') << {      createDir = {          println "create source directory: $it"          // it.mkdirs() //유닉스용          mkdir(it) //윈도우용      }      sourceSets\*.java.srcDirs\*.each createDir      sourceSets\*.resources.srcDirs\*.each createDir      createDir webAppDir      createDir new File(webAppDir, '/WEB-INF')  } |

**4. gradle로 eclipse 프로젝트 작성**

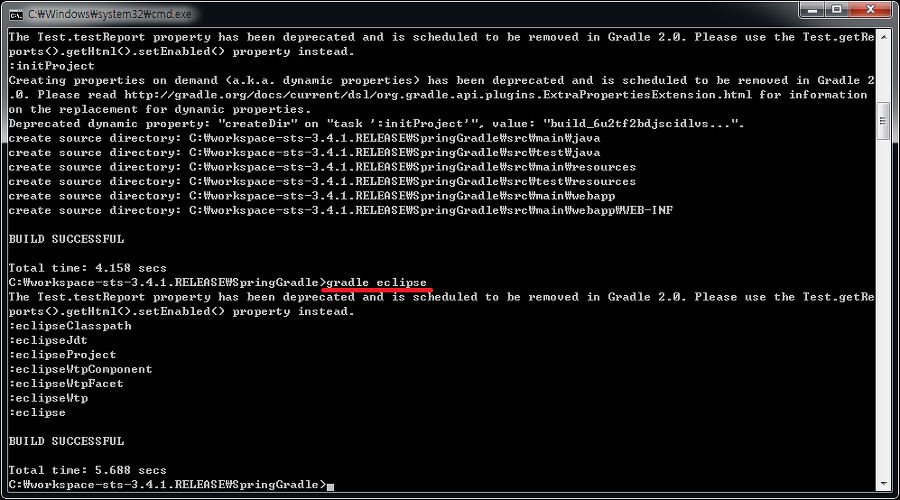
4-1. gradle check (build.gradle 이 정상적으로 작성 되었는지 확인합니다.)



4-2. gradle initProject (프로젝트 디렉토리 생성)



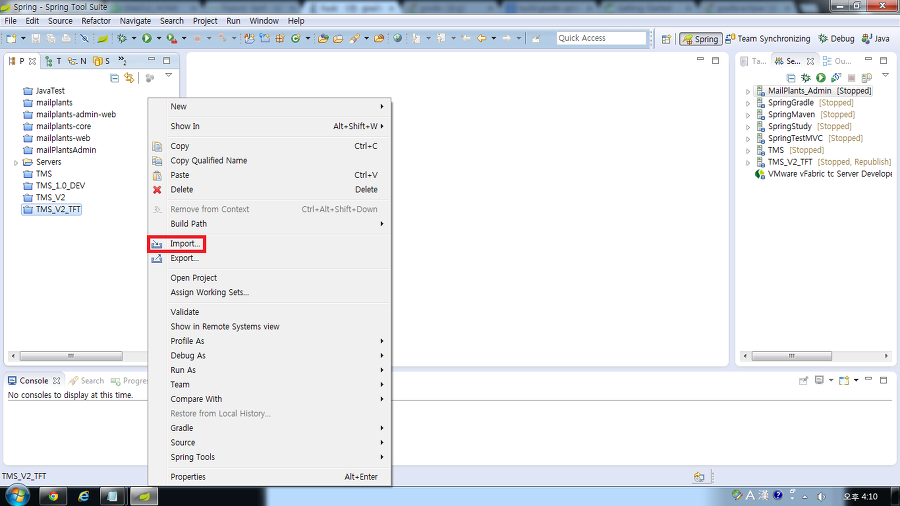
4-3. gradle eclipse (이클립스 프로젝트로 생성)



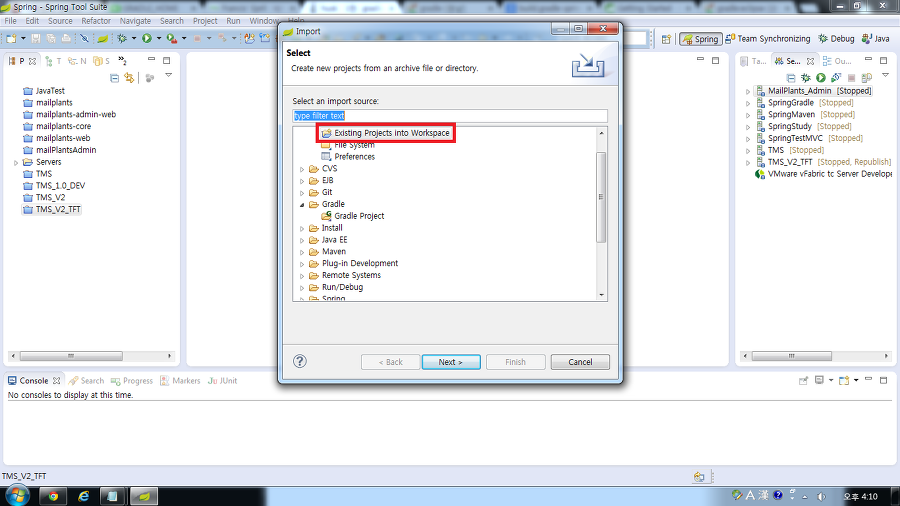
**5. 이클립스(또는 STS)에서 import project**

Package Explorer >> import >> General >> Existing Projects into Workspace >> Select root directory 에서 프로젝트 폴더 선택

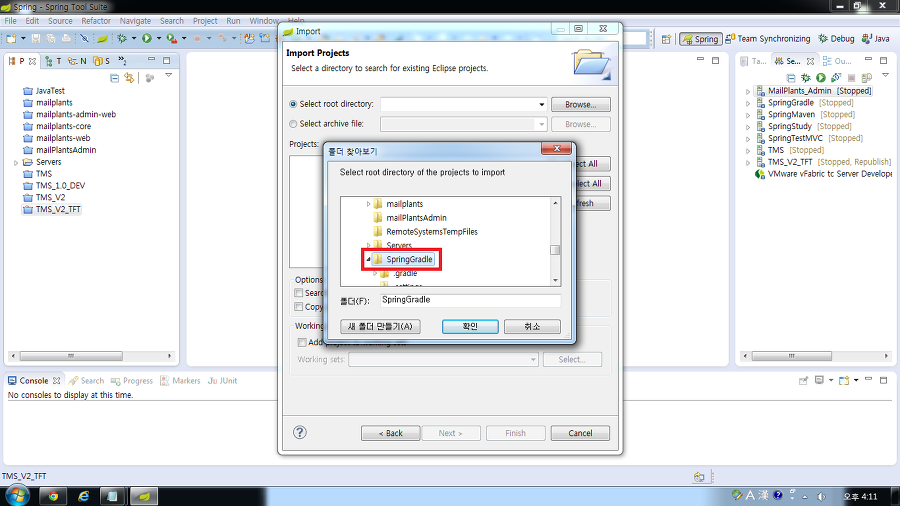
5-1. 우클릭 Import



5-2. General >> Existing Projects into Workspace



5-3. Select root directory (아까 생성한 프로젝트 폴더를 선택한다.)



**6. Gradle 프로젝트로 convert**

프로젝트 우 클릭 후

Configure >> Convert to Gradle Project

프로젝트 위에 G라고 표시가 되면 Gradle 프로젝트 생성 완료

\* 처음 생성하면 왼쪽과 같은 프로젝트 구성이 됩니다.

우측과 같은 폴더로 하기 위해 패키지도 만들어 주고 폴더 및 파일도 오른쪽과 같이 만들어 줬습니다. (자신의 스타일 대로 생성 하시면 됩니다.)

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | http://cfile25.uf.tistory.com/image/27020F445319618F05AD83 | | |  | | --- | | http://cfile7.uf.tistory.com/image/231CD7445319618F2BFDE1 | |

## [2탄. Spring xml 설정을 java설정으로 변경 (Spring JavaConfig)](http://huskdoll.tistory.com/9)

[Spring](http://huskdoll.tistory.com/category/Spring) 2014.03.07 15:14

|  |
| --- |
|  |

- <http://huskdoll.tistory.com/8> (1탄. gradle 로 이클립스 MVC 프로젝트 생성)

참고: <http://docs.spring.io/spring/docs/3.1.x/javadoc-api/org/springframework/web/WebApplicationInitializer.html>

위의 단계를 완료한 후에 설정 하시면 됩니다.

**1. Spring MVC Project 생성 (\* gradle로 프로젝트 생성 하였다면 해당 부분은 패스 하셔도 됩니다.)**

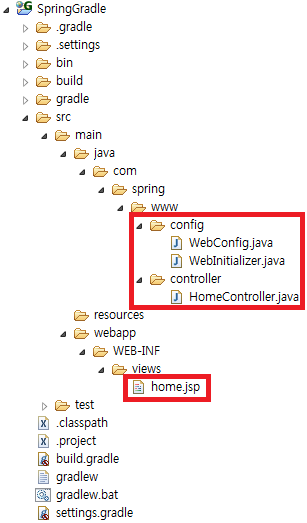
**2. build.gradle 의 dependencies 에 Spring Version 설정**

java config 설정은 Spring 3.2 이상부터 지원 (이미 프로젝트 생성시 3.2 버전 이상이면 해당 부분 필요 없음 제 블로그에서 gradle로 생성 하셨다면 build.gradle 작성서 3.2.2 로 셋팅하였기 때문에 해당 부분 패스 하셔도 됩니다.)

|  |  |
| --- | --- |
| 1  2  3 | dependencies {      org.springframework:spring-webmvc:3.2.2.RELEASE //추가  } |

**3. java 설정**

config 패키지 하나 만들어 주고 해당 부분에 java 설정 파일 작성 (만드려는 위치의 폴더에서 우클릭 후 패키지나 폴더 생성)



\* 해당 폴더를 참고 하시고 취향대로 생성 하시면 됩니다.

3-1. WebConfig.java 작성

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102 | package com.spring.www.config;    import org.springframework.context.MessageSource;  import org.springframework.context.annotation.Bean;  import org.springframework.context.annotation.ComponentScan;  import org.springframework.context.annotation.Configuration;  import org.springframework.context.support.ReloadableResourceBundleMessageSource;  import org.springframework.util.StringUtils;  import org.springframework.web.servlet.LocaleResolver;  import org.springframework.web.servlet.ViewResolver;  import org.springframework.web.servlet.config.annotation.EnableWebMvc;  import org.springframework.web.servlet.config.annotation.InterceptorRegistry;  import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;  import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;  import org.springframework.web.servlet.i18n.CookieLocaleResolver;  import org.springframework.web.servlet.i18n.LocaleChangeInterceptor;  import org.springframework.web.servlet.view.InternalResourceViewResolver;  import org.springframework.web.servlet.view.JstlView;    @Configuration  @EnableWebMvc  @ComponentScan(basePackages = { "com.spring.www.controller" })  public class WebConfig extends WebMvcConfigurerAdapter {        /\*\*       \* CSS / JavaScript / Image 등의 정적 리소스를 처리해주는 핸들러를 등록       \*/      @Override      public void addResourceHandlers(ResourceHandlerRegistry registry) {          registry.addResourceHandler("/resources/\*\*").addResourceLocations("/resources/");      }        /\*\*       \* 인터셉터 (요청을 가로챔)       \*/      @Override      public void addInterceptors(InterceptorRegistry registry) {            LocaleChangeInterceptor localeChangeInterceptor = new LocaleChangeInterceptor();          localeChangeInterceptor.setParamName("lang");          registry.addInterceptor(localeChangeInterceptor);        }        /\*\*       \* locale resolver       \*       \* @return       \*/      @Bean      public LocaleResolver localeResolver() {            CookieLocaleResolver cookieLocaleResolver = new CookieLocaleResolver();          cookieLocaleResolver.setDefaultLocale(StringUtils.parseLocaleString("en"));          return cookieLocaleResolver;      }        /\*\*       \* JSP를 뷰로 사용하는 뷰 리졸버 등록       \*/      @Bean      public ViewResolver viewResolver() {            InternalResourceViewResolver viewResolver = new InternalResourceViewResolver();          viewResolver.setViewClass(JstlView.class);          viewResolver.setPrefix("/WEB-INF/views/");          viewResolver.setSuffix(".jsp");            return viewResolver;      }        @Override      public void addViewControllers(org.springframework.web.servlet.config.annotation.ViewControllerRegistry registry) {            // 특별히 controller 를 타지 않아도 되는 뷰만 있는 경우 등록          // ex) 디자인만 입힌 것들.          registry.addViewController("/simpleView").setViewName("/simpleView");              // 404 오류가 발생했을때 보여줄 뷰를 등록          // registry.addViewController("/page-not-found").setViewName("errors/404");      }        /\*\*       \* message source 들을 등록함       \*       \* @return       \*/      @Bean      public MessageSource messageSource() {            ReloadableResourceBundleMessageSource messageSource = new ReloadableResourceBundleMessageSource();          messageSource.setBasenames("classpath:messages/messages", "classpath:messages/validation");          // if true, the key of the message will be displayed if the key is not          // found, instead of throwing a NoSuchMessageException          messageSource.setUseCodeAsDefaultMessage(true);          messageSource.setDefaultEncoding("UTF-8");          // # -1 : never reload, 0 always reload          messageSource.setCacheSeconds(0);          return messageSource;      }  } |

3-2. WebInitializer.java 작성

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | package com.spring.www.config;    import javax.servlet.Filter;    import org.springframework.web.filter.CharacterEncodingFilter;  import org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;    public class WebInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {      @Override    protected Class<?>[] getRootConfigClasses() {        return null;    }      @Override    protected Class<?>[] getServletConfigClasses() {        return new Class<?>[] { WebConfig.class };    }      @Override    protected String[] getServletMappings() {        return new String[] { "/" };    }      @Override    protected Filter[] getServletFilters() {          CharacterEncodingFilter characterEncodingFilter = new CharacterEncodingFilter();        characterEncodingFilter.setEncoding("UTF-8");          return new Filter[] { characterEncodingFilter};      }  } |

**4. 해당 부분 설정 하고 xml 삭제 (\* gradle로 프로젝트 생성 하였다면 해당 부분은 패스 하셔도 됩니다.)**

web.xml, root-context.xml, servlet-context.xml 다 삭제해도 톰켓에서 돌아감.

**5. controller 설정 (HomeController.java)**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | package com.spring.www.controller;    import org.springframework.stereotype.Controller;  import org.springframework.ui.Model;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestMethod;    /\*\*   \* Handles requests for the application home page.   \*/  @Controller  public class HomeController {          /\*\*       \* Simply selects the home view to render by returning its name.       \*/      @RequestMapping(value = "/", method = RequestMethod.GET)      public String home(Model model) {          String name = "Psyken";              model.addAttribute("name", name );            return "home";      }    } |

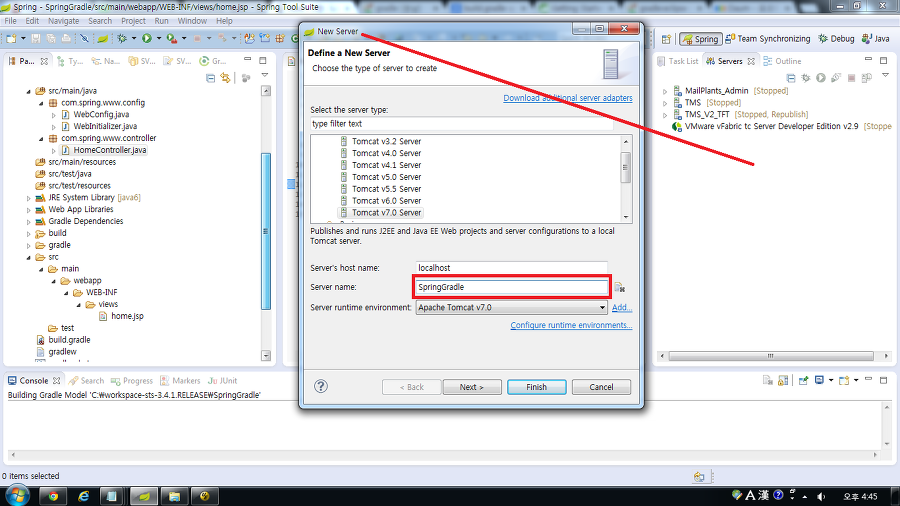
**6. view 설정 (home.jsp)**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | <%@ taglib uri="<http://java.sun.com/jsp/jstl/core>" prefix="c" %>  <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  <%@ page session="false" %>  <html>  <head>    <title>Home</title>  </head>  <body>  <P>  My name is ${name}. </P>  </body>  </html> |

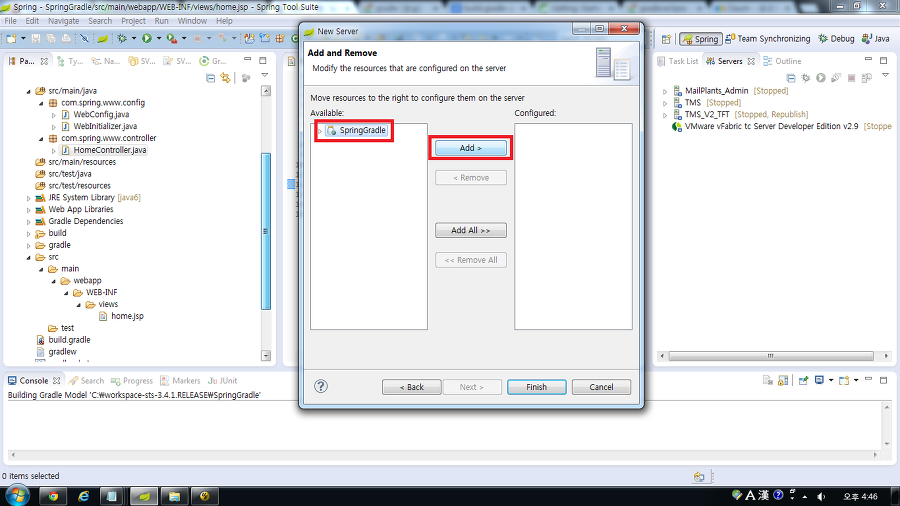
**7. 톰켓 설정**

7-1. Server 에서 우클릭 하여 New >> Server 을 선택 합니다.

해당 Apache >> 톰켓 버전 을 선택하고 서버 이름을 정하시고 그후에 Next 클릭 하세요.

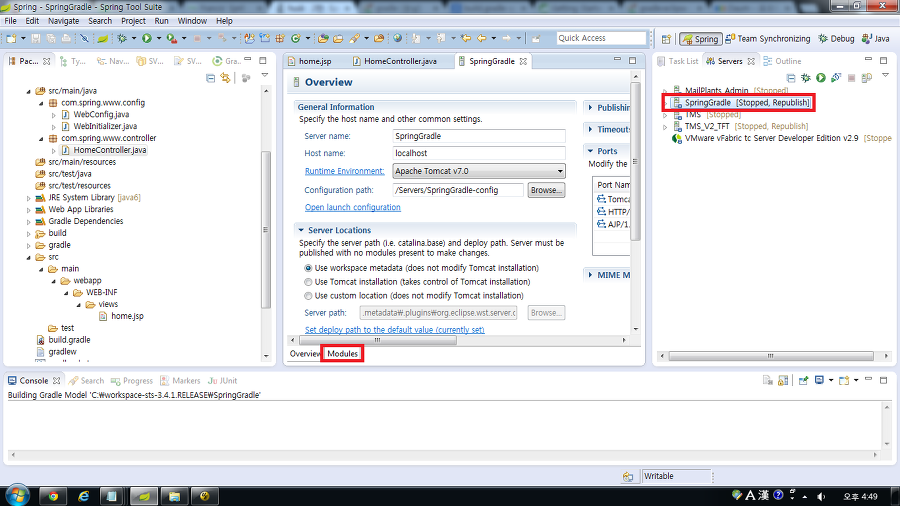


7-2. 왼쪽의 해당 프로젝트를 선택하고 Add로 추가 시킵니다.

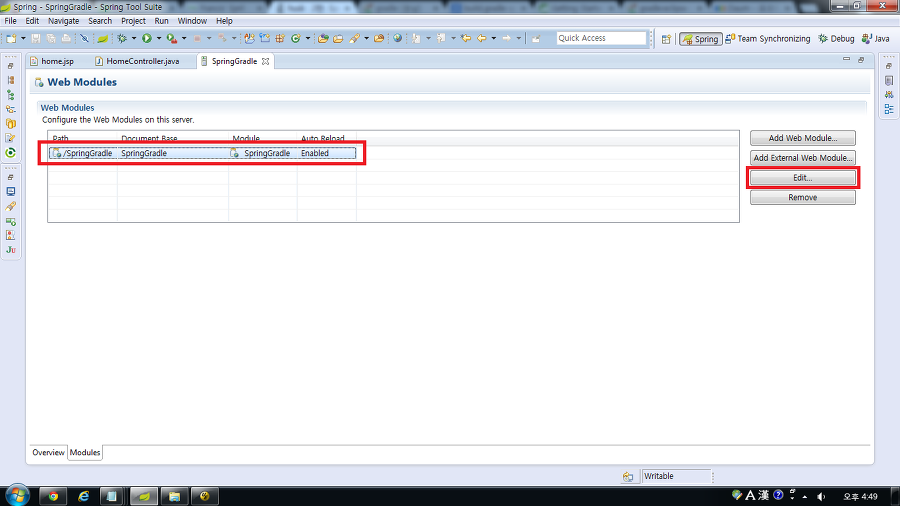


7-3. 등록한 Server를 더블클릭 하면 Server에 대한 설정을 수정 할 수 있는 페이지가 보이게 됩니다.

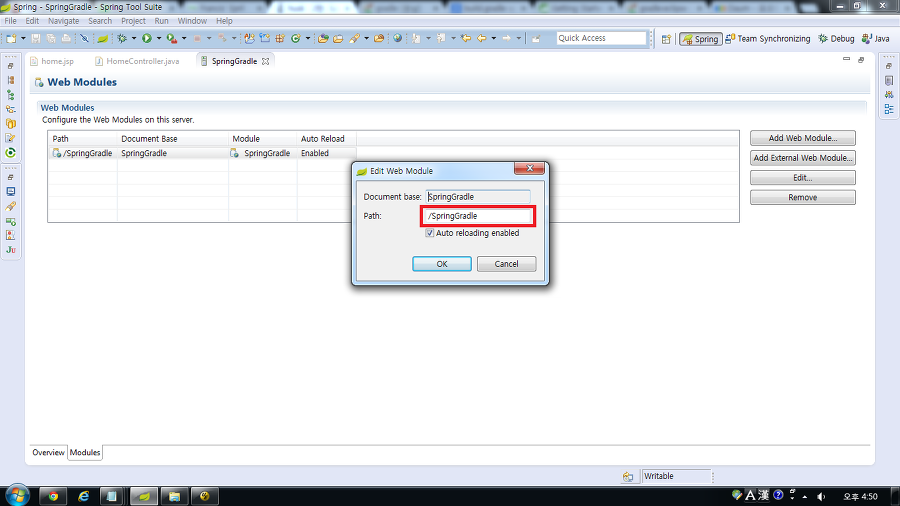
하단의 Modules 탭을 클릭하세요.

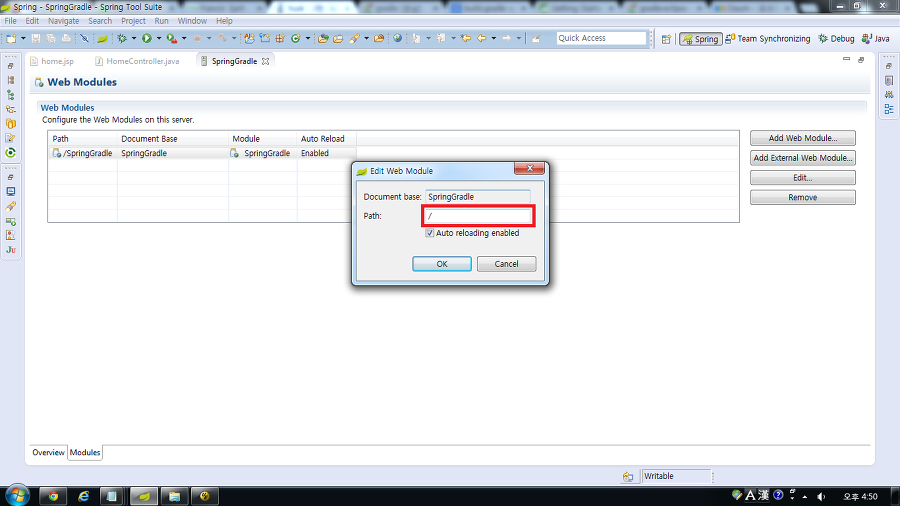


7-4. Modules 탭에서 해당 Web Modules를 선택하고 Edit버튼을 누릅니다.

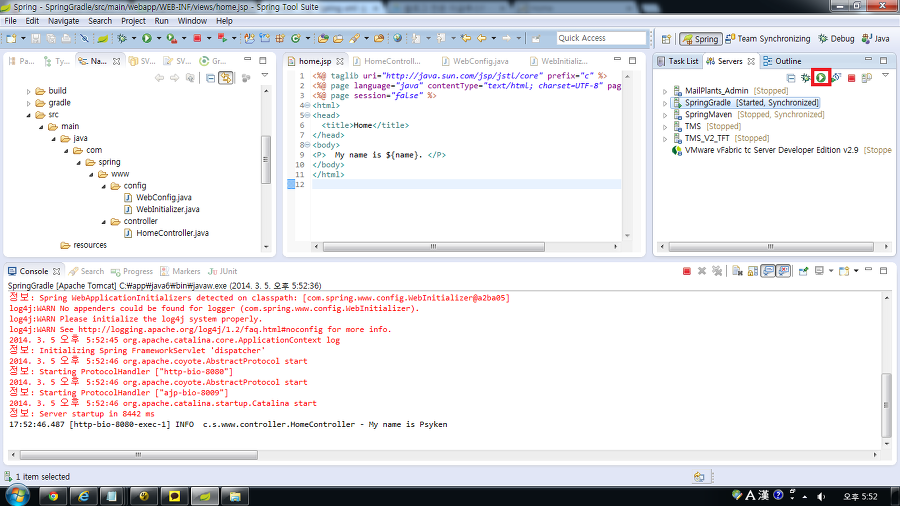


7-5. Path의 기존 경로를 지우고 / 만 남겨 놓고 저장합니다.

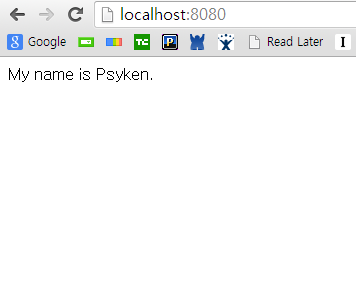




7-6. Servers쪽의 Play 버튼을 눌려서 서버를 구동 시킵니다.



7-7. <http://localhost:8080/> 눌러서 웹페이지에서 확인 하셔서 해당 페이지가 보인다면 성공.



## [3탄. MyBatis 설정](http://huskdoll.tistory.com/19)

[Spring](http://huskdoll.tistory.com/category/Spring) 2014.03.10 16:24

|  |
| --- |
|  |

- <http://huskdoll.tistory.com/8> (1탄. gradle 로 이클립스 MVC 프로젝트 생성)

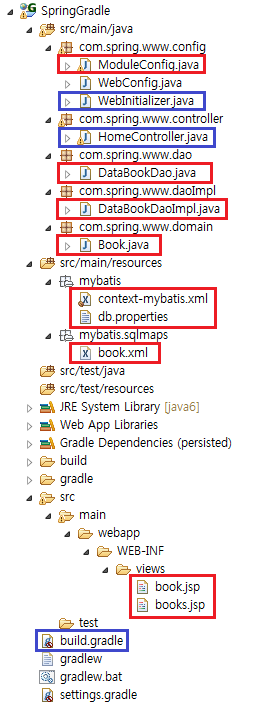
- <http://huskdoll.tistory.com/9> (2탄. Spring xml 설정을 java설정으로 변경 (Spring JavaConfig))

위의 단계를 완료한 후에 설정 하시면 됩니다.

MyBatis 설정을 하기위한 소스 예제를 작성 하였습니다.

아래에 디렉토리를 참고하셔서 순서대로 따라해 보시길 바랍니다. ^^

소스는 <https://github.com/psyken/SpringGradle> 에 올려 놓았습니다.



**※디렉토리 참고**

**- 파란색 (수정)**

**- 빨간색 (추가)**

**1. 조회 할 Table 및 데이터 입력**

1.1 테이블 생성

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | CREATE TABLE BOOKS (    ID VARCHAR(5) NOT NULL,    NAME VARCHAR(20) NOT NULL,    WRITER VARCHAR(20) NOT NULL,    PRICE DECIMAL(10,2) DEFAULT 0,    GENRE VARCHAR(20) NOT NULL,    PUBLISHER VARCHAR(20) NOT NULL,    CNT DECIMAL(10,0) DEFAULT 0,    PRIMARY KEY  (ID)  ) ENGINE=InnoDB DEFAULT CHARSET=utf8 |

1.2 데이터 입력

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35 | INSERT INTO BOOKS(    ID,    NAME,    WRITER,    PRICE,    GENRE,    PUBLISHER  ) VALUES (    'A1',    '홍길동전',    '허균',    300,    '소설',    '허균 출판사'  )   ON DUPLICATE KEY UPDATE   CNT = CNT+1;     INSERT INTO BOOKS(    ID,    NAME,    WRITER,    PRICE,    GENRE,    PUBLISHER  ) VALUES (    'B1',    '레미제라블',    '빅토르 위고',    900,    '소설',    '빅토르 위고 출판사'  )   ON DUPLICATE KEY UPDATE   CNT = CNT+1; |

**2. build.gradle 내용 수정**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89 | /\*   \* This build file was auto generated by running the Gradle 'init' task   \* by 'psyken' at '14. 3. 5 오후 3:46' with Gradle 1.11   \*   \* This generated file contains a sample Java project to get you started.   \* For more details take a look at the Java Quickstart chapter in the Gradle   \* user guide available at <http://gradle.org/docs/1.11/userguide/tutorial_java_projects.html>   \*/    // Apply the java plugin to add support for Java  apply plugin: 'java'  apply plugin: 'war'  apply plugin: 'eclipse'  apply plugin: 'eclipse-wtp'    // JAVA Version 1.6  sourceCompatibility = 1.6  // 개발한 애플리케이션 버전  version = '1.0'    // 메이븐 Central 저장소 사용  repositories {      mavenCentral()  }    // dependency 버전 정보  def version = [  spring: '3.2.2.RELEASE'  , junit: '4.11'  , servletApi: '3.0.1'  , jstl: '1.2'  , slf4j: '1.7.6'  , mockito: '1.9.0'  , cglib: '2.2.2'  , logback: '1.0.6'  , mybatis: '3.2.2' //MyBatis 추가  , mybatisSpring: '1.2.0' //MyBatis 추가  , mysql: '5.1.26' //MyBatis 추가  ]    // In this section you declare the dependencies for your production and test code  // 의존성 설정  dependencies {      compile "org.springframework:spring-webmvc:${version.spring}",            "cglib:cglib-nodep:${version.cglib}",            "ch.qos.logback:logback-classic:${version.logback}",            "org.slf4j:slf4j-api:${version.slf4j}",            "org.mybatis:mybatis:${version.mybatis}", //MyBatis 추가            "org.mybatis:mybatis-spring:${version.mybatisSpring}", //MyBatis 추가            "mysql:mysql-connector-java:${version.mysql}", //MyBatis 추가            "org.springframework:spring-jdbc:${version.spring}", //MyBatis 추가            "org.springframework:spring-tx:${version.spring}" //MyBatis 추가      providedCompile "javax.servlet:javax.servlet-api:${version.servletApi}"      testCompile "org.springframework:spring-test:${version.spring}",                "junit:junit:${version.junit}",                "org.mockito:mockito-core:${version.mockito}"      runtime "javax.servlet:jstl:${version.jstl}"  }    // logback(slf4j)를 사용하기 때문에 모든 의존성에서 commons-logging는 제외  [configurations.runtime, configurations.default]\*.exclude(module: 'commons-logging')    // JAVA 컴파일시 인코딩 설정  [compileJava, compileTestJava]\*.options\*.encoding = 'UTF-8'    // TEST 설정  test {      jvmArgs = ['-ea', '-Xmx256m']      logging.captureStandardOutput(LogLevel.INFO)      testReport = false  }    // 프로젝트 초기화  // 1. java source directory 생성 : src/main/java, src/test/java  // 2. resource directory 생성    : src/main/resource, src/test/resource  // 3. web source directory 생성  : src/main/webapp, src/main/webapp/WEB-INF  task initProject(description: 'initialize project') << {      createDir = {          println "create source directory: $it"          it.mkdirs()      }      sourceSets\*.java.srcDirs\*.each createDir      sourceSets\*.resources.srcDirs\*.each createDir      createDir webAppDir      createDir new File(webAppDir, '/WEB-INF')  } |

36 ~ 38 라인 추가

48 ~ 52 라인 추가

**3. ModuleConfig.java 생성**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | package com.spring.www.config;    import org.springframework.context.annotation.Configuration;  import org.springframework.context.annotation.ImportResource;    /\*\*   \* 웹쪽 설정이 아닌 공통으로 쓰일수 있는 부분의 설정   \*/  @Configuration  @ImportResource("classpath:mybatis/context-mybatis.xml")  public class ModuleConfig {    } |

**4. WebInitializer.java 내용 수정**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35 | package com.spring.www.config;    import javax.servlet.Filter;    import org.springframework.web.filter.CharacterEncodingFilter;  import org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;    public class WebInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {      @Override    protected Class<?>[] getRootConfigClasses() {      return new Class<?>[]{ ModuleConfig.class };//ModuleConfig 웹쪽 설정이 아닌 공통으로 쓰일수 있는 부분의 설정      //return null; //주석처리    }      @Override    protected Class<?>[] getServletConfigClasses() {        return new Class<?>[] { WebConfig.class };    }      @Override    protected String[] getServletMappings() {        return new String[] { "/" };    }      @Override    protected Filter[] getServletFilters() {          CharacterEncodingFilter characterEncodingFilter = new CharacterEncodingFilter();        characterEncodingFilter.setEncoding("UTF-8");          return new Filter[] { characterEncodingFilter};      }  } |

12 라인 추가

**5. db.properties 생성**

|  |  |
| --- | --- |
| 1  2  3  4 | jdbc.driver=com.mysql.jdbc.Driver  jdbc.url=jdbc:<mysql://localhost/db>  jdbc.user=username  jdbc.pwd=password |

**6. context-mybatis.xml 생성**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52 | <?xml version="1.0" encoding="UTF-8"?>  <beans xmlns="<http://www.springframework.org/schema/beans>" xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>"    xmlns:p="<http://www.springframework.org/schema/p>"    xmlns:context="<http://www.springframework.org/schema/context>"    xmlns:tx="<http://www.springframework.org/schema/tx>"    xsi:schemaLocation="<http://www.springframework.org/schema/beans> <http://www.springframework.org/schema/beans/spring-beans-3.2.xsd>  <http://www.springframework.org/schema/context> <http://www.springframework.org/schema/context/spring-context-3.2.xsd>  <http://www.springframework.org/schema/tx> <http://www.springframework.org/schema/tx/spring-tx-3.2.xsd>">     <context:component-scan base-package="com.spring.www" />     <context:property-placeholder location="classpath:mybatis/db.properties"/>     <!-- commons dbcp : 커넥션 풀 사용 (commons-dbcp을 pom.xml에 dependencies 한다.)-->   <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource">    <property name="driverClassName" value="${jdbc.driver}"/>    <property name="url" value="${jdbc.url}"/>    <property name="username" value="${jdbc.user}"/>    <property name="password" value="${jdbc.pwd}"/>      <property name="maxActive" value="50"/>    <property name="maxIdle" value="30"/>    <property name="minIdle" value="30"/>    <property name="maxWait" value="5"/>     </bean>     <!-- MyBatis 맵퍼의 SQL 문장에 대한 파라미터 및 리턴 타입을 담을 클래스를 검색 할 패키지 지정 -->   <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">    <property name="dataSource" ref="dataSource" />    <property name="mapperLocations" value="classpath:mybatis/sqlmaps/\*.xml" />   </bean>     <bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate">    <constructor-arg index="0" ref="sqlSessionFactory" />   </bean>     <!-- 맵퍼 인터페이스와 맵퍼 XML 파일에 의해 생성되는 맵퍼 클래스를 빈에 주입 할 수 있도록 설정 -->   <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">     <property name="basePackage" value="com.spring.www" />   </bean>     <!-- 트랜잭션 관리자. id가 transactionManager이면, 아래 tx:annotation-driven에서 transactionManager 속성을 지정할 필요 없음 -->   <bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">    <property name="dataSource" ref="dataSource"/>   </bean>     <!-- 애노테이션으로 트랜잭션 설정 할 수 있도록 -->   <tx:annotation-driven />   <!--tx:annotation-driven transaction-manager="transactionManager" /-->    </beans> |

**7. Domain 생성 (Book.java)**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56 | package com.spring.www.domain;    public class Book {        private String id;      private String name;      private String writer;      private String price;      private String genre;      private String publisher;      private String cnt;        public String getId() {          return id;      }      public void setId(String id) {          this.id = id;      }      public String getName() {          return name;      }      public void setName(String name) {          this.name = name;      }      public String getWriter() {          return writer;      }      public void setWriter(String writer) {          this.writer = writer;      }      public String getPrice() {          return price;      }      public void setPrice(String price) {          this.price = price;      }      public String getGenre() {          return genre;      }      public void setGenre(String genre) {          this.genre = genre;      }      public String getPublisher() {          return publisher;      }      public void setPublisher(String publisher) {          this.publisher = publisher;      }      public String getCnt() {          return cnt;      }      public void setCnt(String cnt) {          this.cnt = cnt;      }    } |

**8. Dao 생성 (DataBookDao.java)**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | package com.spring.www.dao;    import java.util.List;    import com.spring.www.domain.Book;    public interface DataBookDao {        List<Book> selectAllBook();        String selectBookName(String id);    } |

**9. DaoImpl 생성 (DataBookDaoImpl.java)**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38 | package com.spring.www.daoImpl;    import java.util.List;    import org.apache.ibatis.session.SqlSession;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Repository;    import com.spring.www.dao.DataBookDao;  import com.spring.www.domain.Book;    @Repository("DataBookDao")  public class DataBookDaoImpl implements DataBookDao{        @Autowired      private SqlSession sqlSession;        public void setSqlSession(SqlSession sqlSession){          this.sqlSession = sqlSession;      }        @Override      public List<Book> selectAllBook() {          // TODO Auto-generated method stub          List<Book> users = sqlSession.selectList("getBookInfo");            return users;      }        @Override      public String selectBookName(String id) {          // TODO Auto-generated method stub          String name = sqlSession.selectOne("getBookName", id);            return name;      }    } |

**10. 쿼리 xml 생성 (book.xml)**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | <?xml version="1.0" encoding="UTF-8" ?>  <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"    "<http://mybatis.org/dtd/mybatis-3-mapper.dtd>">    <mapper namespace="com.spring.www.dao.DataBookDao"><!-- namespace에서 interface로 정의된 DAO를 정확히 명시해주어야 한다 -->    <select id="getBookInfo" resultType="com.spring.www.domain.Book">      SELECT        ID,        NAME,        WRITER,        PRICE,        GENRE,        PUBLISHER,        CNT      FROM        BOOKS    </select>    <select id="getBookName" parameterType="string" resultType="string">      SELECT        NAME      FROM        BOOKS      WHERE        ID = #{id}    </select>  </mapper> |

**11. controller 생성 (HomeController.java)**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44 | package com.spring.www.controller;    import javax.annotation.Resource;    import org.slf4j.Logger;  import org.slf4j.LoggerFactory;  import org.springframework.stereotype.Controller;  import org.springframework.ui.Model;  import org.springframework.web.bind.annotation.PathVariable;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestMethod;    import com.spring.www.dao.DataBookDao;    /\*\*   \* Handles requests for the application home page.   \*/  @Controller  public class HomeController {        private static final Logger logger = LoggerFactory.getLogger(HomeController.class);        @Resource(name="DataBookDao")      private DataBookDao bookDao;        @RequestMapping(value = "/", method = RequestMethod.GET)      public String home(Model model) {            model.addAttribute("books", bookDao.selectAllBook());            return "books";        }        @RequestMapping(value = "/{id}", method = RequestMethod.GET)      public String home(@PathVariable("id") String id, Model model) {            model.addAttribute("bookName", bookDao.selectBookName(id));            return "book";        }    } |

**12. view 생성 [단건 출력] (book.jsp)**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | <%@ taglib uri="<http://java.sun.com/jsp/jstl/core>" prefix="c" %>  <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  <%@ page session="false" %>  <html>  <head>    <title>Home</title>  </head>  <body>  <h1>    Book Name.  </h1>    <P>  This book's name is [${bookName}]. </P>  </body>  </html> |

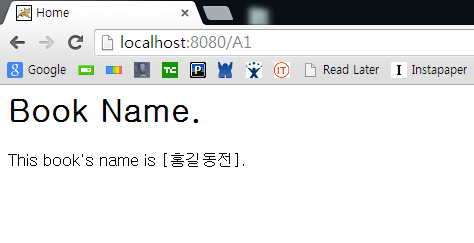
**13. view 생성 [리스트 출력] (books.jsp)**

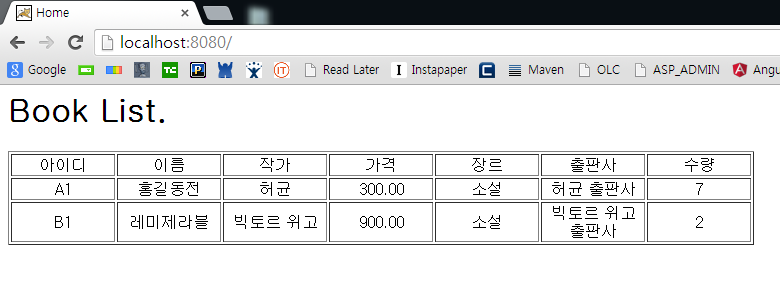
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35 | <%@ taglib uri="<http://java.sun.com/jsp/jstl/core>" prefix="c" %>  <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  <%@ page session="false" %>  <html>  <head>    <title>Home</title>  </head>  <body>  <h1>    Book List.  </h1>  <table border="1">    <tr align="center">      <td width="100px">아이디</td>      <td width="100px">이름</td>      <td width="100px">작가</td>      <td width="100px">가격</td>      <td width="100px">장르</td>      <td width="100px">출판사</td>      <td width="100px">수량</td>    </tr>  <c:forEach var="book" items="${books}">    <tr align="center">      <td width="100px">${book.id}</td>      <td width="100px">${book.name}</td>      <td width="100px">${book.writer}</td>      <td width="100px">${book.price}</td>      <td width="100px">${book.genre}</td>      <td width="100px">${book.publisher}</td>      <td width="100px">${book.cnt}</td>    </tr>  </c:forEach>  </table>  </body>  </html> |

**14. 페이지 호출**

URL 창에 http://localhost:8080/ 입력 하시면 전체 책 목록이 나옵니다.

http://localhost:8080/책아이디 를 입력하시면 해당 아이디의 책 이름이 보이게 됩니다.





## [4탄. STS 에서 gradle 로 war 배포](http://huskdoll.tistory.com/22)

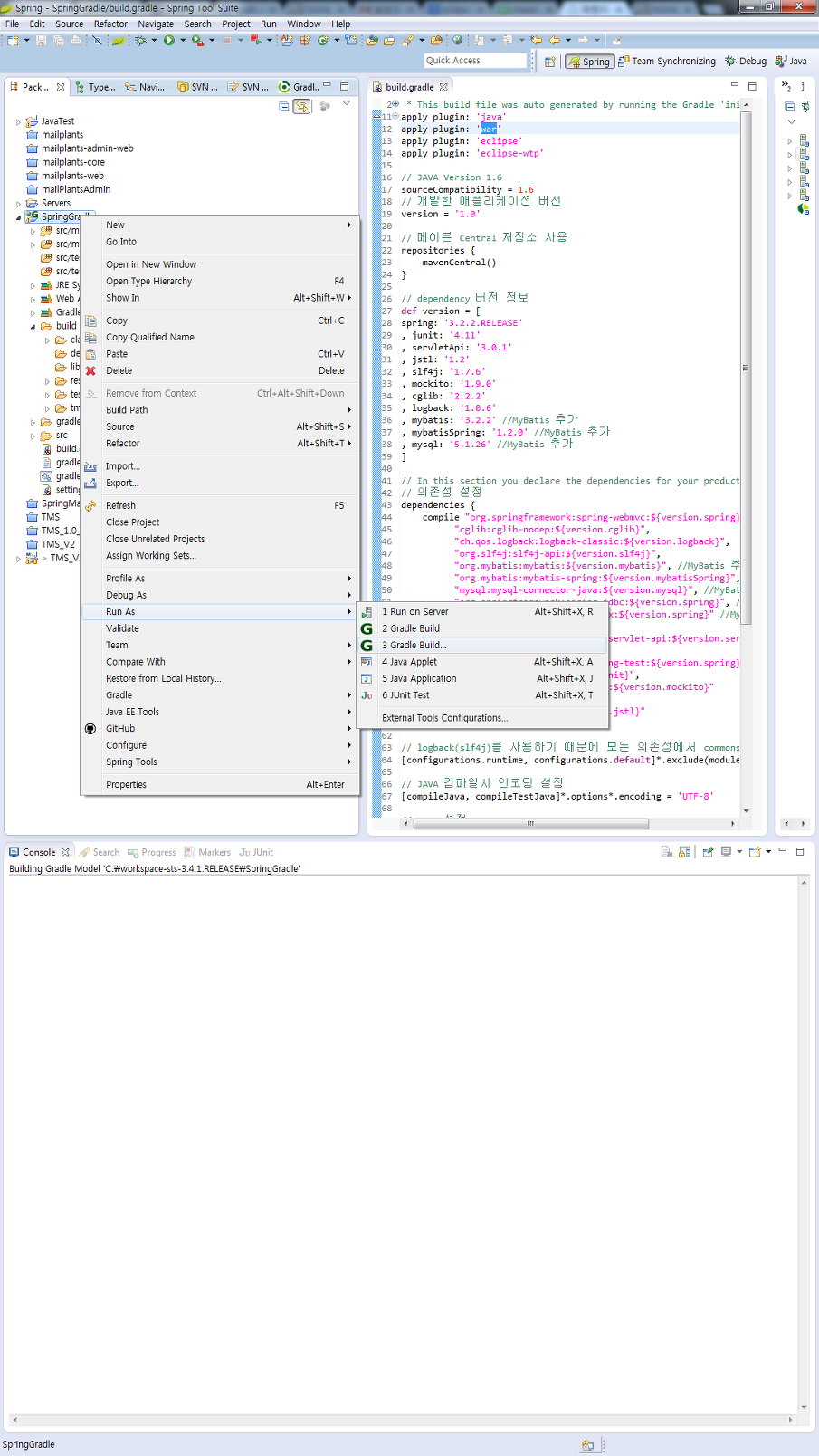
[Spring](http://huskdoll.tistory.com/category/Spring) 2014.03.11 15:08

|  |
| --- |
|  |

개발 완료한 프로젝트를 Tomcat에 올리기 위해 war파일로 배포 하는 방식 입니다.

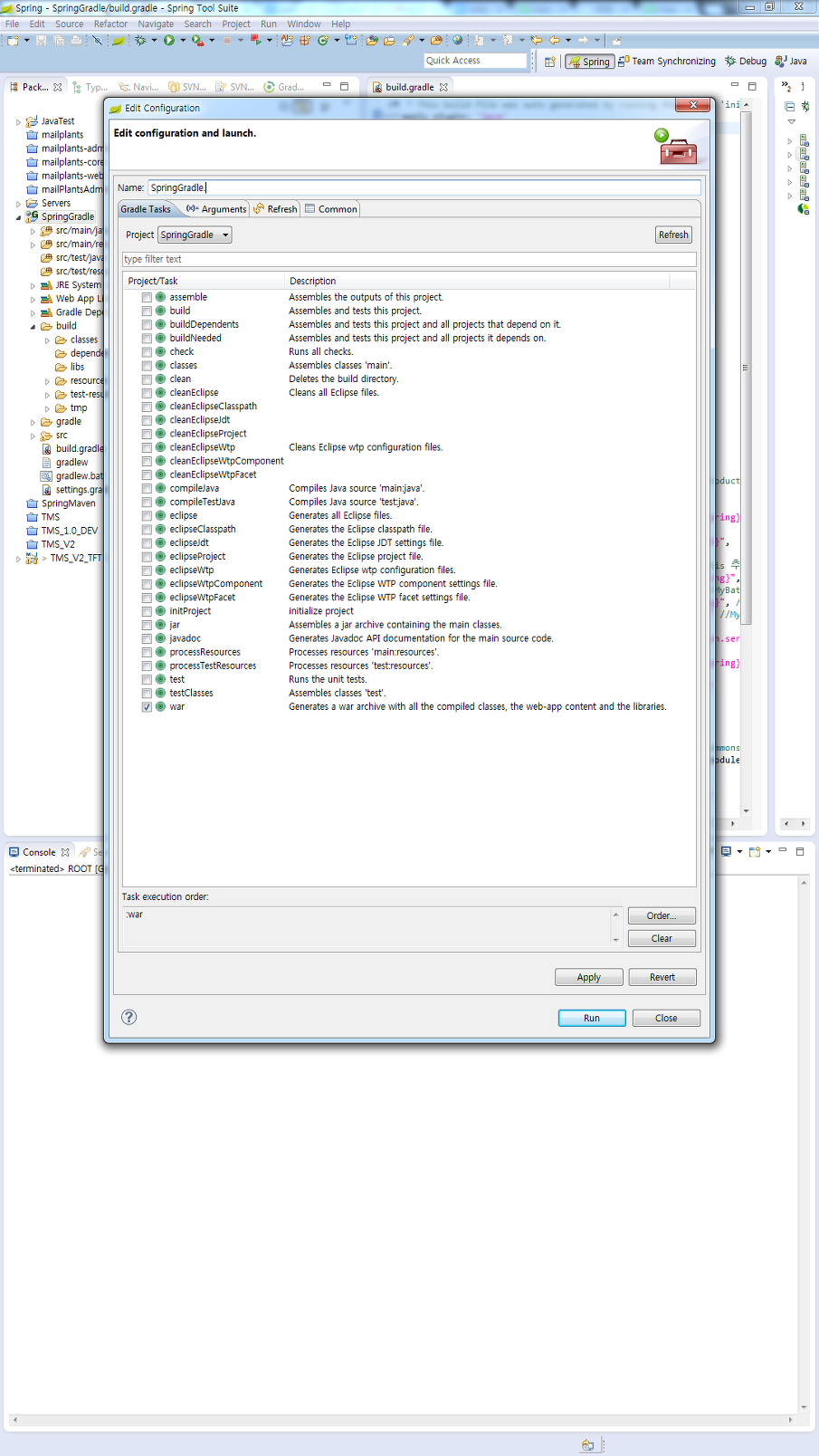
**1. 프로젝트에서 Run As >> Gradle Build...**

해당 프로젝트에서 우클릭 한후 Run As >> Gradle Build... 을 선택 합니다.



**2. war 선택 후 Run**

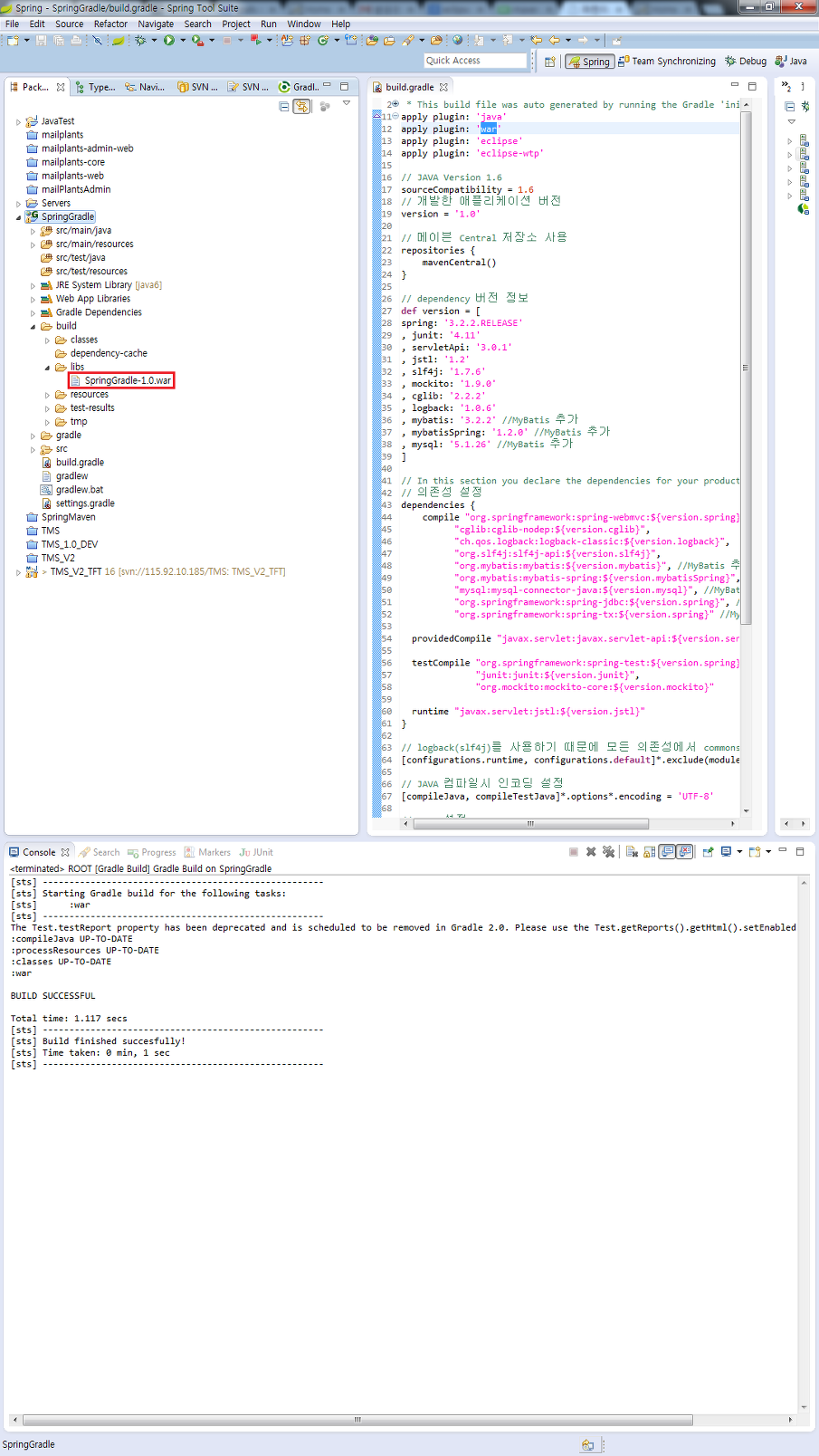
체크박스에서 war를 선택하고 Run을 실행 합니다.



**3. war 파일 확인**

build 폴더 하위 폴더인 libs 에 war 파일이 생성 됩니다. (확인이 안 되시면 프로젝트에서 Refresh를 해주세요.)

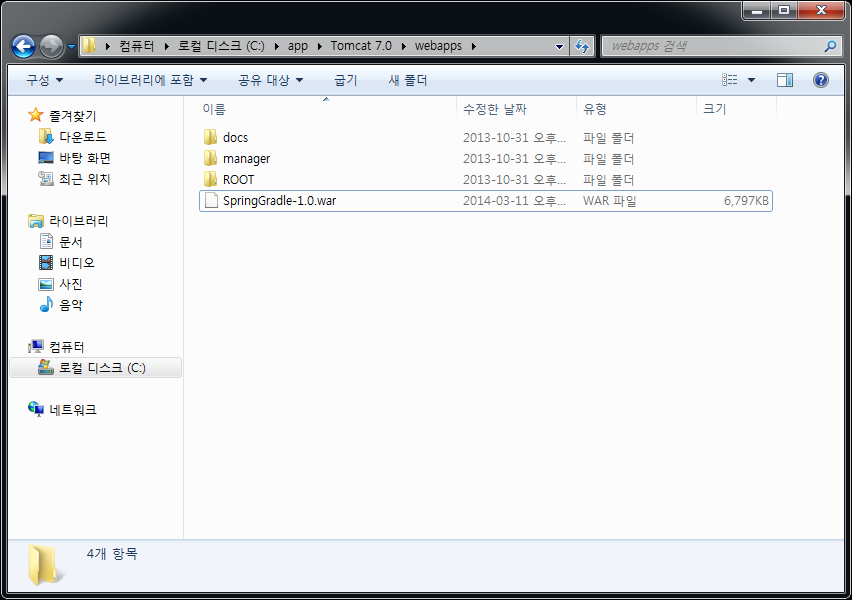
Console창에서 BUILD SUCCESSFUL 확인 하신다면 성공 입니다.



**4. Tomcat에 war를 올림**

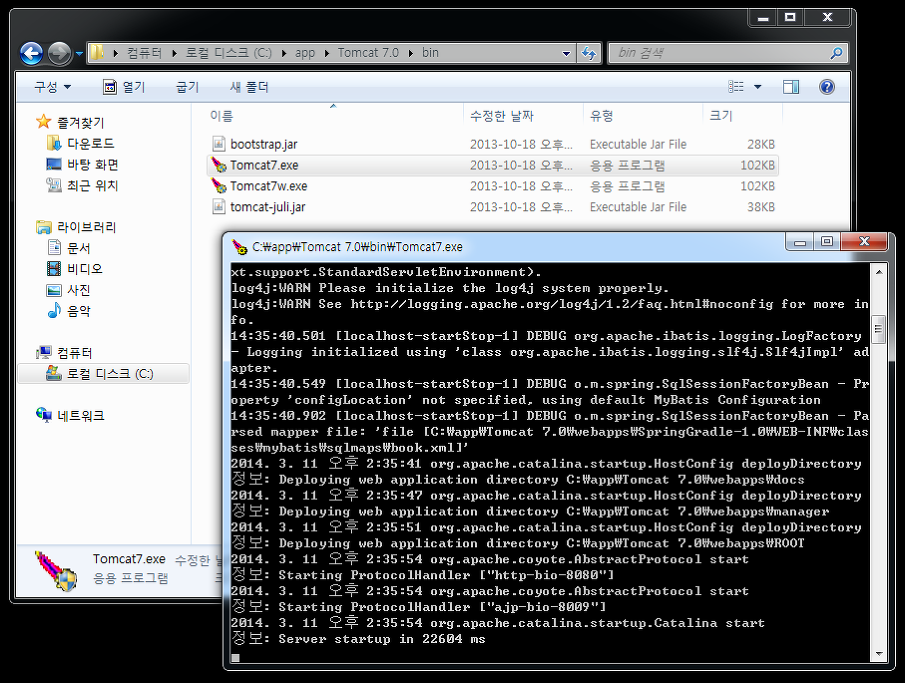
Tomcat이 설치된 위치의 webapps에 war파일을 올려줍니다. (톰켓설치위치\Tomcat 7.0\webapps => war파일)

\* 해당 부분은 간단하게 하기 위해서 tomcat의 webapps에 넣었습니다. 경로를 변경 하고 싶으시면 올리시려는 위치네 놓으시고 tomcat의 conf폴더에서 server.xml의 Host 부분에 지정해 주세요.



**5. Tomcat 실행**

Tomcat이 설치된 위치의 bin에서 Tomcat7.exe를 실행 시켜주세요. (톰켓설치위치\Tomcat 7.0\bin\Tomcat7.exe)



**6. 웹페이지 확인**

톰켓이 올라가면서 war파일이 풀리게 됩니다.

주소창에 (http://localhost:8080/war폴더명) 을 실행시키시면 확인 가능합니다.

경로를 수정하시려면 4번을 확인하시여 server.xml을 수정하시여 host를 설치한 경로로 변경하시면 됩니다. 설명이 되어있는 블로그가 있어서 링크를 추가 합니다. (<http://blog.naver.com/jwlee0208/10125127966>)

