Day 08 (2019-01-30)
1. Springi MVC 기본

(1) Spring MVC 무작정 실행하기

1) 프로젝트 생성

이전시간에 진행한 내용을 참조하여 Maven 프로젝트 생성후 아래와 같이 pom.xml 파일 작성

pom.xml

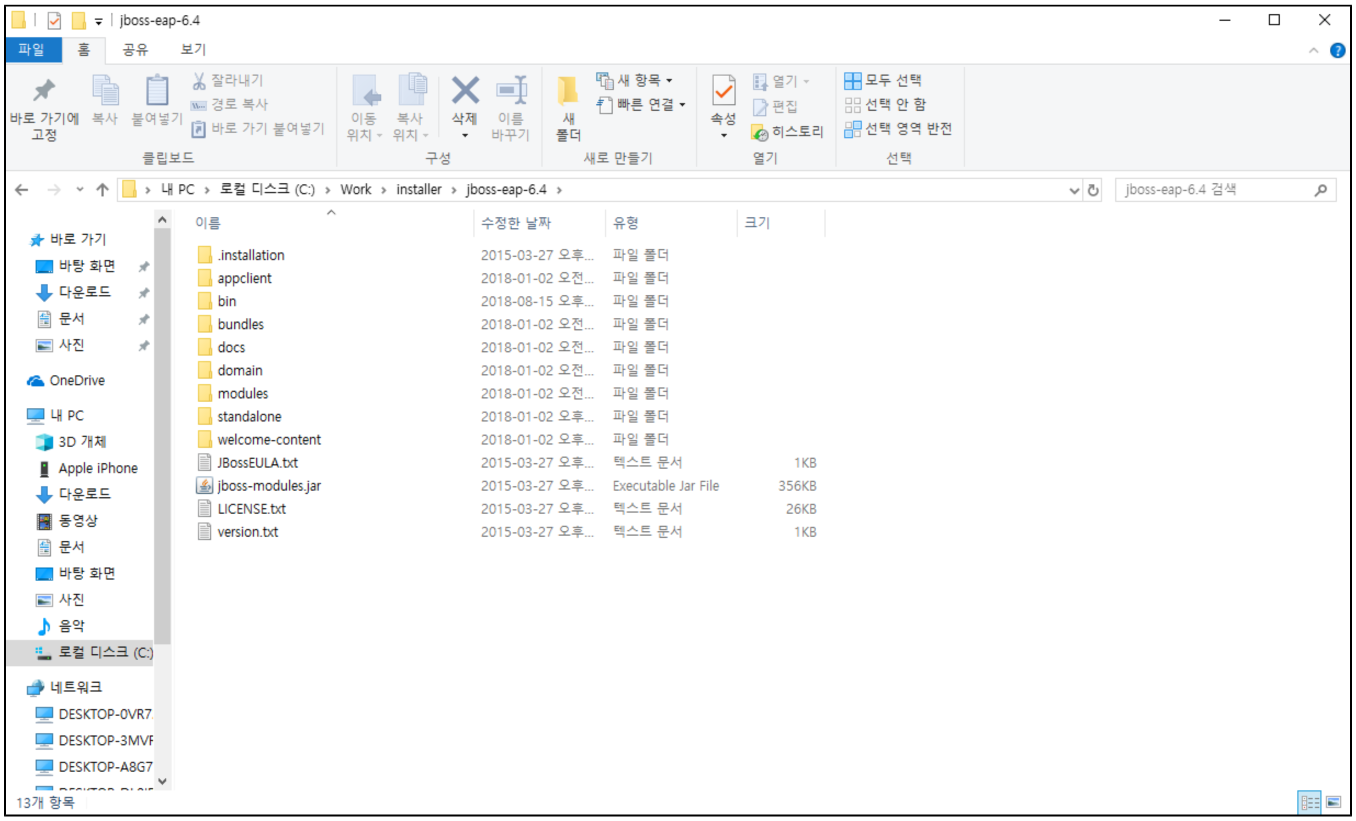
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>

   <groupId>io.crscube</groupId>
   <artifactId>semina</artifactId>
   <name>semina</name>
   <packaging>war</packaging>
   <version>1.0-SNAPSHOT</version>
   <build>
      <finalName>semina</finalName>
      <plugins>
         <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-war-plugin</artifactId>
            <version>2.4</version>
         </plugin>
         <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <configuration>
               <source>${java.version}</source>
               <target>${java.version}</target>
            </configuration>
         </plugin>
      </plugins>
   </build>

   <properties>
      <java.version>1.8</java.version>
      <org.springframework.version>4.3.22.RELEASE</org.springframework.version>
      <javax.inject.javax.inject.version>1</javax.inject.javax.inject.version>
      <org.aspectj.aspectjrt.version>1.7.4</org.aspectj.aspectjrt.version>
      <org.aspectj.aspectjweaver.version>1.7.4</org.aspectj.aspectjweaver.version>
      <com.fasterxml.jackson.core.jackson-databind.version>2.5.2</com.fasterxml.jackson.core.jackson-databind.
version>
      <javax.servlet.jsp.jsp-api.version>2.2</javax.servlet.jsp.jsp-api.version>
      <javax.servlet.javax.servlet-api.version>3.1.0</javax.servlet.javax.servlet-api.version>
      <javax.servlet.jstl.version>1.2</javax.servlet.jstl.version>
      <junit.junit.version>4.12</junit.junit.version>
      <org.mokito.mokito-all.version>1.9.5</org.mokito.mokito-all.version>
      <org.assertj.assertj-core.version>3.4.1</org.assertj.assertj-core.version>
   </properties>

   <dependencies>
      <!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->
      <dependency>
         <groupId>org.springframework</groupId>
         <artifactId>spring-core</artifactId>
         <version>${org.springframework.version}</version>
      </dependency>
      <dependency>
```

```
        <groupId>org.springframework</groupId>
        <artifactId>spring-context-support</artifactId>
        <version>${org.springframework.version}</version>
     </dependency>
     <!-- https://mvnrepository.com/artifact/org.springframework/spring-web -->
     <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-web</artifactId>
        <version>${org.springframework.version}</version>
     </dependency>
     <!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->
     <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>${org.springframework.version}</version>
     </dependency>
     <!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->
     <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-test</artifactId>
        <version>${org.springframework.version}</version>
        <scope>test</scope>
     </dependency>

     <dependency>
        <groupId>javax.inject</groupId>
        <artifactId>javax.inject</artifactId>
        <version>${javax.inject.javax.inject.version}</version>
     </dependency>
     <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-aspects</artifactId>
        <version>${org.springframework.version}</version>
     </dependency>

     <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>javax.servlet-api</artifactId>
        <version>${javax.servlet.javax.servlet-api.version}</version>
     </dependency>

     <dependency>
        <groupId>javax.servlet.jsp</groupId>
        <artifactId>jsp-api</artifactId>
        <version>${javax.servlet.jsp.jsp-api.version}</version>
     </dependency>

     <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>jstl</artifactId>
        <version>${javax.servlet.jstl.version}</version>
     </dependency>

     <dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-core</artifactId>
        <version>${com.fasterxml.jackson.core.jackson-databind.version}</version>
     </dependency>
     <dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-databind</artifactId>
        <version>${com.fasterxml.jackson.core.jackson-databind.version}</version>
     </dependency>
     <dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-annotations</artifactId>
        <version>${com.fasterxml.jackson.core.jackson-databind.version}</version>
     </dependency>
     <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
```

```
            <version>${junit.junit.version}</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.mockito</groupId>
            <artifactId>mockito-all</artifactId>
            <version>${org.mokito.mokito-all.version}</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.assertj</groupId>
            <artifactId>assertj-core</artifactId>
            <version>${org.assertj.assertj-core.version}</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

2) WAS 설정

JBOSS를 다운로드하여 특정 디렉토리에 위치



이름	수정한 날짜	유형	크기
.installation	2015-03-27 오후...	파일 폴더	
appclient	2018-01-02 오전...	파일 폴더	
bin	2018-08-15 오후...	파일 폴더	
bundles	2018-01-02 오전...	파일 폴더	
docs	2018-01-02 오전...	파일 폴더	
domain	2018-01-02 오전...	파일 폴더	
modules	2018-01-02 오전...	파일 폴더	
standalone	2018-01-02 오후...	파일 폴더	
welcome-content	2018-01-02 오후...	파일 폴더	
JBossEULA.txt	2015-03-27 오후...	텍스트 문서	1KB
jboss-modules.jar	2015-03-27 오후...	Executable Jar File	356KB
LICENSE.txt	2015-03-27 오후...	텍스트 문서	26KB
version.txt	2015-03-27 오후...	텍스트 문서	1KB

C:₩Work₩installer₩jboss-eap-6.4₩standalone₩configuration₩crscube 위치에 아래 설정파일 저장

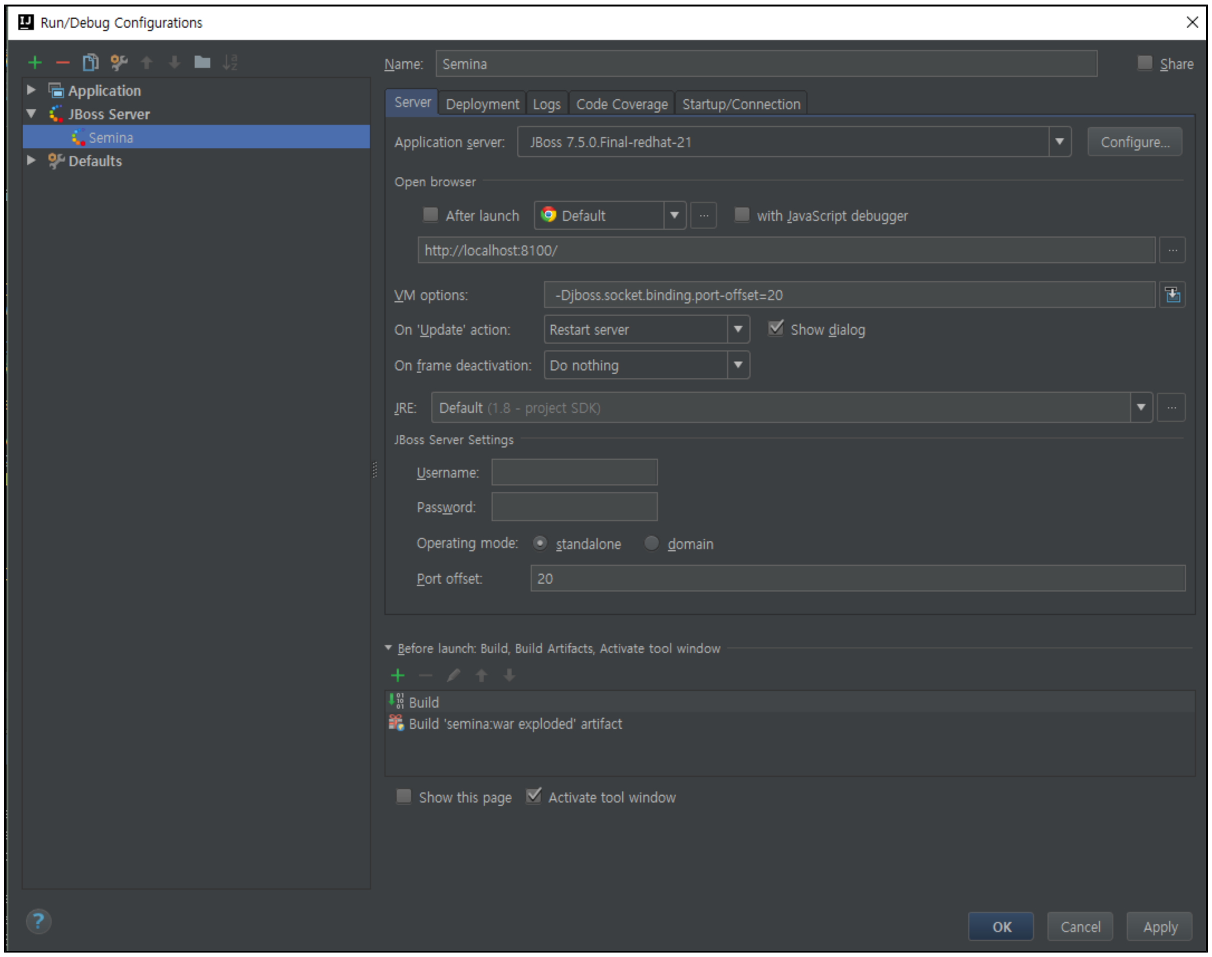
standalone-semina.xml

```
<?xml version='1.0' encoding='UTF-8'?>
```

```
<server xmlns="urn:jboss:domain:1.5">

    <extensions>
        <extension module="org.jboss.as.clustering.infinispan"/>
        <extension module="org.jboss.as.connector"/>
        <extension module="org.jboss.as.deployment-scanner"/>
        <extension module="org.jboss.as.ee"/>
        <extension module="org.jboss.as.ejb3"/>
        <extension module="org.jboss.as.jaxrs"/>
        <extension module="org.jboss.as.jdr"/>
        <extension module="org.jboss.as.jmx"/>
        <extension module="org.jboss.as.jpa"/>
        <extension module="org.jboss.as.jsf"/>
        <extension module="org.jboss.as.logging"/>
        <extension module="org.jboss.as.mail"/>
        <extension module="org.jboss.as.naming"/>
        <extension module="org.jboss.as.pojo"/>
        <extension module="org.jboss.as.remoting"/>
        <extension module="org.jboss.as.sar"/>
        <extension module="org.jboss.as.security"/>
        <extension module="org.jboss.as.threads"/>
        <extension module="org.jboss.as.transactions"/>
        <extension module="org.jboss.as.web"/>
        <extension module="org.jboss.as.webservices"/>
        <extension module="org.jboss.as.weld"/>
    </extensions>


    <management>
        <security-realms>
            <security-realm name="ManagementRealm">
                <authentication>
                    <local default-user="$local"/>
                    <properties path="mgmt-users.properties" relative-to="jboss.server.config.dir"/>
                </authentication>
                <authorization map-groups-to-roles="false">
                    <properties path="mgmt-groups.properties" relative-to="jboss.server.config.dir"/>
                </authorization>
            </security-realm>
            <security-realm name="ApplicationRealm">
                <authentication>
                    <local default-user="$local" allowed-users="*"/>
                    <properties path="application-users.properties" relative-to="jboss.server.config.dir"/>
                </authentication>
                <authorization>
                    <properties path="application-roles.properties" relative-to="jboss.server.config.dir"/>
                </authorization>
            </security-realm>
        </security-realms>
        <audit-log>
            <formatters>
                <json-formatter name="json-formatter"/>
            </formatters>
            <handlers>
                <file-handler name="file" formatter="json-formatter" path="audit-log.log" relative-to="jboss.
server.data.dir"/>
            </handlers>
            <logger log-boot="true" log-read-only="false" enabled="false">
                <handlers>
                    <handler name="file"/>
                </handlers>
            </logger>
        </audit-log>
        <management-interfaces>
            <native-interface security-realm="ManagementRealm">
                <socket-binding native="management-native"/>
            </native-interface>
            <http-interface security-realm="ManagementRealm">
                <socket-binding http="management-http"/>
            </http-interface>
        </management-interfaces>
        <access-control provider="simple">
            <role-mapping>
                <role name="SuperUser">
                    <include>
                        <user name="$local"/>
                    </include>
                </role>
            </role-mapping>
        </access-control>
    </management>

    <profile>
        <subsystem xmlns="urn:jboss:domain:logging:1.3">
            <console-handler name="CONSOLE">
                <level name="DEBUG"/> <!--level name="INFO"/-->
                <formatter>
                    <!--pattern-formatter pattern="%K{level}%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n"/-->
                    <pattern-formatter pattern="%d{HH:mm:ss,SSS} %s%n"/>
                </formatter>
            </console-handler>
            <periodic-rotating-file-handler name="FILE" autoflush="true">
                <formatter>
                    <pattern-formatter pattern="%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n"/>
                </formatter>
                <file relative-to="jboss.server.log.dir" path="server.log"/>
                <suffix value=".yyyy-MM-dd"/>
                <append value="true"/>
            </periodic-rotating-file-handler>
            <logger category="com.arjuna">
                <level name="WARN"/>
            </logger>
            <logger category="org.apache.tomcat.util.modeler">
                <level name="WARN"/>
            </logger>
            <logger category="org.jboss.as.config">
                <level name="DEBUG"/>
            </logger>
            <logger category="sun.rmi">
                <level name="WARN"/>
            </logger>
            <logger category="jacorb">
                <level name="WARN"/>
            </logger>
                        <logger category="kr.co.crscube.ctms">
                <level name="DEBUG"/>
            </logger>
            <logger category="org.springframework.core">
               <level name="ERROR" />
            </logger>
            <logger category="org.springframework.beans">
                <level name="ERROR" />
            </logger>
            <logger category="org.springframework.context">
                <level name="ERROR" />
            </logger>
            <logger category="org.springframework.web">
                <level name="ERROR" />
            </logger>
            <logger category="org.springframework.jdbc">
               <level name="ERROR" />
            </logger>
            <logger category="jacorb.config">
                <level name="ERROR"/>
            </logger>
            <root-logger>
                <level name="INFO"/> <!--level name="INFO"/-->
                <handlers>
                    <handler name="CONSOLE"/>
                    <handler name="FILE"/>
                </handlers>
            </root-logger>
        </subsystem>
        <subsystem xmlns="urn:jboss:domain:datasources:1.1">
            <datasources>
                <drivers>
                    <driver name="oracle" module="com.oracle">
                        <xa-datasource-class>oracle.jdbc.xa.client.OracleXADataSource</xa-datasource-class>
                    </driver>
                </drivers>
            </datasources>
        </subsystem>
        <subsystem xmlns="urn:jboss:domain:deployment-scanner:1.1">
            <deployment-scanner path="deployments" relative-to="jboss.server.base.dir" scan-interval="5000"/>
        </subsystem>
        <subsystem xmlns="urn:jboss:domain:ee:1.1">
            <spec-descriptor-property-replacement>false</spec-descriptor-property-replacement>
            <jboss-descriptor-property-replacement>true</jboss-descriptor-property-replacement>
        </subsystem>
        <subsystem xmlns="urn:jboss:domain:ejb3:1.4">
            <session-bean>
                <stateless>
                    <bean-instance-pool-ref pool-name="slsb-strict-max-pool"/>
                </stateless>
                <stateful default-access-timeout="5000" cache-ref="simple"/>
                <singleton default-access-timeout="5000"/>
            </session-bean>
            <pools>
                <bean-instance-pools>
                    <strict-max-pool name="slsb-strict-max-pool" max-pool-size="20" instance-acquisition-
timeout="5" instance-acquisition-timeout-unit="MINUTES"/>
                    <strict-max-pool name="mdb-strict-max-pool" max-pool-size="20" instance-acquisition-
timeout="5" instance-acquisition-timeout-unit="MINUTES"/>
                </bean-instance-pools>
            </pools>
            <caches>
                <cache name="simple" aliases="NoPassivationCache"/>
                <cache name="passivating" passivation-store-ref="file" aliases="SimpleStatefulCache"/>
            </caches>
            <passivation-stores>
                <file-passivation-store name="file"/>
            </passivation-stores>
            <async thread-pool-name="default"/>
            <timer-service thread-pool-name="default">
                <data-store path="timer-service-data" relative-to="jboss.server.data.dir"/>
            </timer-service>
            <remote connector-ref="remoting-connector" thread-pool-name="default"/>
            <thread-pools>
                <thread-pool name="default">
                    <max-threads count="10"/>
                    <keepalive-time time="100" unit="milliseconds"/>
                </thread-pool>
            </thread-pools>
            <default-security-domain value="other"/>
            <default-missing-method-permissions-deny-access value="true"/>
        </subsystem>
        <subsystem xmlns="urn:jboss:domain:infinispan:1.4">
            <cache-container name="web" aliases="standard-session-cache" default-cache="local-web" module="org.
jboss.as.clustering.web.infinispan">
                <local-cache name="local-web" batching="true">
                    <file-store passivation="false" purge="false"/>
                </local-cache>
            </cache-container>
            <cache-container name="hibernate" default-cache="local-query" module="org.jboss.as.jpa.hibernate:4">
                <local-cache name="entity">
                    <transaction mode="NON_XA"/>
                    <eviction strategy="LRU" max-entries="10000"/>
                    <expiration max-idle="100000"/>
                </local-cache>
                <local-cache name="local-query">
                    <transaction mode="NONE"/>
                    <eviction strategy="LRU" max-entries="10000"/>
                    <expiration max-idle="100000"/>
                </local-cache>
                <local-cache name="timestamps">
                    <transaction mode="NONE"/>
                    <eviction strategy="NONE"/>
                </local-cache>
            </cache-container>
            <cache-container name="application" default-cache="local-acl" start="EAGER" module="org.jboss.as.
clustering.web.infinispan">
                <local-cache name="local-acl">
                    <transaction mode="NONE"/>
                    <eviction strategy="LRU" max-entries="10000"/>
                </local-cache>
            </cache-container>
        </subsystem>
        <subsystem xmlns="urn:jboss:domain:jaxrs:1.0"/>
        <subsystem xmlns="urn:jboss:domain:jca:1.1">
            <archive-validation enabled="true" fail-on-error="true" fail-on-warn="false"/>
            <bean-validation enabled="true"/>
            <default-workmanager>
                <short-running-threads>
                    <core-threads count="50"/>
                    <queue-length count="50"/>
                    <max-threads count="50"/>
                    <keepalive-time time="10" unit="seconds"/>
                </short-running-threads>
                <long-running-threads>
                    <core-threads count="50"/>
                    <queue-length count="50"/>
                    <max-threads count="50"/>
                    <keepalive-time time="10" unit="seconds"/>
                </long-running-threads>
            </default-workmanager>
            <cached-connection-manager/>
        </subsystem>
        <subsystem xmlns="urn:jboss:domain:jdr:1.0"/>
        <subsystem xmlns="urn:jboss:domain:jmx:1.3">
            <expose-resolved-model/>
            <expose-expression-model/>
            <remoting-connector/>
        </subsystem>
        <subsystem xmlns="urn:jboss:domain:jpa:1.1">
            <jpa default-datasource="" default-extended-persistence-inheritance="DEEP"/>
        </subsystem>
        <subsystem xmlns="urn:jboss:domain:jsf:1.0"/>
        <subsystem xmlns="urn:jboss:domain:mail:1.1">
            <mail-session jndi-name="java:jboss/mail/Default">
                <smtp-server outbound-socket-binding-ref="mail-smtp"/>
            </mail-session>
        </subsystem>
        <subsystem xmlns="urn:jboss:domain:naming:1.4">
            <remote-naming/>
        </subsystem>
        <subsystem xmlns="urn:jboss:domain:pojo:1.0"/>
        <subsystem xmlns="urn:jboss:domain:remoting:1.1">
            <connector name="remoting-connector" socket-binding="remoting" security-realm="ApplicationRealm"/>
        </subsystem>
        <subsystem xmlns="urn:jboss:domain:resource-adapters:1.1"/>
        <subsystem xmlns="urn:jboss:domain:sar:1.0"/>
        <subsystem xmlns="urn:jboss:domain:security:1.2">
            <security-domains>
                <security-domain name="other" cache-type="default">
                    <authentication>
                        <login-module code="Remoting" flag="optional">
                            <module-option name="password-stacking" value="useFirstPass"/>
                        </login-module>
                        <login-module code="RealmDirect" flag="required">
                            <module-option name="password-stacking" value="useFirstPass"/>
                        </login-module>
                    </authentication>
                </security-domain>
                <security-domain name="jboss-web-policy" cache-type="default">
```

```
                    <authorization>
                        <policy-module code="Delegating" flag="required"/>
                    </authorization>
                </security-domain>
                <security-domain name="jboss-ejb-policy" cache-type="default">
                    <authorization>
                        <policy-module code="Delegating" flag="required"/>
                    </authorization>
                </security-domain>
            </security-domains>
        </subsystem>
        <subsystem xmlns="urn:jboss:domain:threads:1.1"/>
        <subsystem xmlns="urn:jboss:domain:transactions:1.4">
            <core-environment>
                <process-id>
                    <uuid/>
                </process-id>
            </core-environment>
            <recovery-environment socket-binding="txn-recovery-environment" status-socket-binding="txn-status-
manager"/>
            <coordinator-environment default-timeout="300"/>
        </subsystem>
        <subsystem xmlns="urn:jboss:domain:web:1.5" default-virtual-server="default-host" native="false">
            <connector name="http" protocol="HTTP/1.1" scheme="http" socket-binding="http"/>
            <virtual-server name="default-host" enable-welcome-root="false">
                <alias name="localhost"/>
                <alias name="example.com"/>
            </virtual-server>
        </subsystem>
        <subsystem xmlns="urn:jboss:domain:webservices:1.2">
            <modify-wsdl-address>true</modify-wsdl-address>
            <wsdl-host>${jboss.bind.address:127.0.0.1}</wsdl-host>
            <endpoint-config name="Standard-Endpoint-Config"/>
            <endpoint-config name="Recording-Endpoint-Config">
                <pre-handler-chain name="recording-handlers" protocol-bindings="##SOAP11_HTTP
##SOAP11_HTTP_MTOM ##SOAP12_HTTP ##SOAP12_HTTP_MTOM">
                    <handler name="RecordingHandler" class="org.jboss.ws.common.invocation.
RecordingServerHandler"/>
                </pre-handler-chain>
            </endpoint-config>
            <client-config name="Standard-Client-Config"/>
        </subsystem>
        <subsystem xmlns="urn:jboss:domain:weld:1.0"/>
    </profile>

    <interfaces>
        <interface name="management">
            <inet-address value="${jboss.bind.address.management:127.0.0.1}"/>
        </interface>
        <interface name="public">
            <inet-address value="${jboss.bind.address:127.0.0.1}"/>
        </interface>
        <interface name="unsecure">
            <inet-address value="${jboss.bind.address.unsecure:127.0.0.1}"/>
        </interface>
    </interfaces>

    <socket-binding-group name="standard-sockets" default-interface="public" port-offset="${jboss.socket.
binding.port-offset:20}">
        <socket-binding name="management-native" interface="management" port="${jboss.management.native.port:
9999}"/>
        <socket-binding name="management-http" interface="management" port="${jboss.management.http.port:9990}"
/>
        <socket-binding name="management-https" interface="management" port="${jboss.management.https.port:
9443}"/>
        <socket-binding name="ajp" port="8009"/>
        <socket-binding name="http" port="8080"/>
        <socket-binding name="https" port="8443"/>
        <socket-binding name="remoting" port="4447"/>
        <socket-binding name="txn-recovery-environment" port="4712"/>
        <socket-binding name="txn-status-manager" port="4713"/>
        <outbound-socket-binding name="mail-smtp">
            <remote-destination host="localhost" port="25"/>
        </outbound-socket-binding>
    </socket-binding-group>

</server>
```

IntelliJ Run/Debug Configurations 메뉴의 Server탭에서 JBoss Server 추가하여 아래와 같이 설정



Name: Semina

Application server: JBoss 7.5.0.Final-redhat-21

After launch: Default; with JavaScript debugger

http://localhost:8100/

VM options: -Djboss.socket.binding.port-offset=20

On 'Update' action: Restart server ✓ Show dialog

On frame deactivation: Do nothing

JRE: Default (1.8 - project SDK)

JBoss Server Settings

Username:

Password:

Operating mode: [● standalone] domain

Port offset: 20

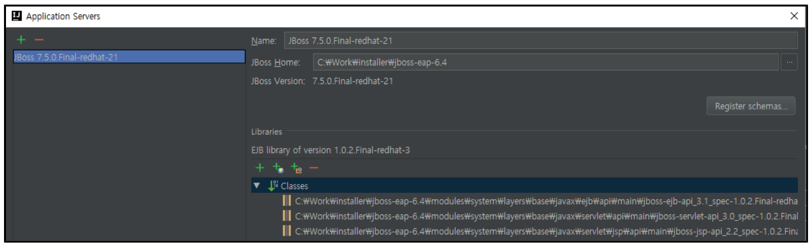
Before launch: Build, Build Artifacts, Activate tool window

Build

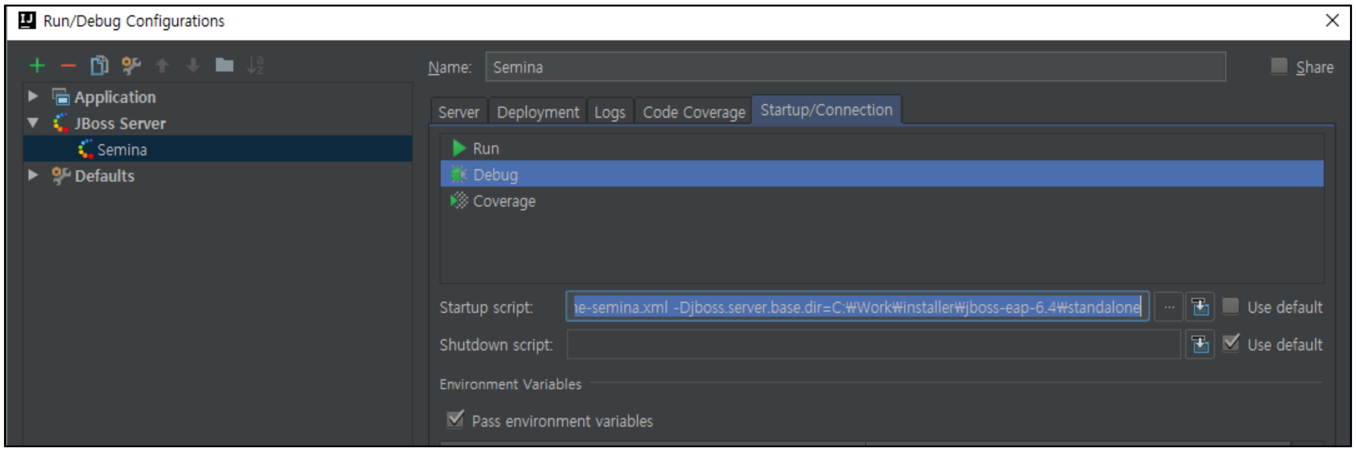
Build 'semina:war exploded' artifact

Show this page ✓ Activate tool window

Attribute	Value
Name	Configuration name

Application server	Jboss home 위치 입력  Name: JBoss 7.5.0.Final-redhat-21 JBoss Home: C:₩Work₩installer₩jboss-eap-6.4 JBoss Version: 7.5.0.Final-redhat-21 EJB library of version 1.0.2.Final-redhat-3 Classes C:₩Work₩installer₩jboss-eap-6.4₩modules₩system₩layers₩base₩javax₩ejb₩api₩main₩jboss-ejb-api_3.1_spec-1.0.2.Final-redha C:₩Work₩installer₩jboss-eap-6.4₩modules₩system₩layers₩base₩javax₩servlet₩api₩main₩jboss-servlet-api_3.0_spec-1.0.2.Final C:₩Work₩installer₩jboss-eap-6.4₩modules₩system₩layers₩base₩javax₩servlet₩jsp₩api₩main₩jboss-jsp-api_2.2_spec-1.0.2.Fin
Port offset	20으로 설정
Operating mode	Standalone 선택

Startup/Connection 탭에서 Debug 항목에 Startup script 입력



Name: Semina

Startup script: ıe-semina.xml -Djboss.server.base.dir=C:₩Work₩installer₩jboss-eap-6.4₩standalone

Shutdown script:

Use default

Pass environment variables

[JBOSS Home]₩bin₩standalone.bat -b [바인딩할 IP주소] --server-config=crscube/standalone-semina.xml -Djboss.server.base.dir= [JBOSS Home]

3) Spring MVC를 위한 설정

src/main/webapp/WEB-INF 디렉토리에 web.xml 작성

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0"
         xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.
xsd">

    <display-name>cubeCTMS Application</display-name>


    <filter>
        <filter-name>characterSetEncodingFilter</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>
        <init-param>
            <param-name>forceEncoding</param-name>
            <param-value>true</param-value>
        </init-param>
    </filter>
    <filter-mapping>
        <filter-name>characterSetEncodingFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>

    <servlet>
        <servlet-name>dispatcher</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
            <param-name>throwExceptionIfNoHandlerFound</param-name>
            <param-value>true</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
        <multipart-config>
            <max-file-size>1073741824</max-file-size>
            <max-request-size>1073741824</max-request-size>
            <file-size-threshold>0</file-size-threshold>
        </multipart-config>
    </servlet>

    <servlet-mapping>
        <servlet-name>dispatcher</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>

    <distributable/>

   <session-config>
      <session-timeout>60</session-timeout>
   </session-config>

</web-app>
```

src/main/webapp/WEB-INF 디렉토리에 dispatcher-servlet.xml 작성

dispatcher-servlet.xml

```
<?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:context="http://www.springframework.org/schema
/context"
      xmlns:aop="http://www.springframework.org/schema/aop" xmlns:mvc="http://www.springframework.org/schema
/mvc"
      xsi:schemaLocation="http://www.springframework.org/schema/beans   http://www.springframework.org/schema
/beans/spring-beans.xsd http://www.springframework.org/schema/context http://www.springframework.org/schema
/context/spring-context.xsd   http://www.springframework.org/schema/aop http://www.springframework.org/schema
/aop/spring-aop.xsd http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-
mvc.xsd">

   <mvc:annotation-driven />
   <context:component-scan base-package="io.crscube.semina" />
</beans>
```

4) 코드 구현 및 실행

HelloController 작성

HelloController.java

```
@RestController
public class HelloController {
   @GetMapping("/")
   public ResponseEntity<String> getHelloMessage() {
      return ResponseEntity.ok().body("hello");
   }
}
```

Application 실행하여 http:// [바인드 한 IP] :8100으로 접속하여 확인

(2) Deployment Descriptor 설정 파헤치기

1) XML 기반 설정

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0"
         xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.
xsd">

    <display-name>cubeCTMS Application</display-name>

    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>
         classpath:spring/root-context.xml
        </param-value>
    </context-param>

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


    <filter>
        <filter-name>characterSetEncodingFilter</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>
        <init-param>
            <param-name>forceEncoding</param-name>
            <param-value>true</param-value>
        </init-param>
    </filter>
    <filter-mapping>
        <filter-name>characterSetEncodingFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>

    <servlet>
        <servlet-name>dispatcher</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
            <param-name>throwExceptionIfNoHandlerFound</param-name>
            <param-value>true</param-value>
        </init-param>
                <init-param>
                   <param-name>contextConfigLocation</param-name>
                   <param-value>WEB-INF/dispatcher-servlet.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
        <multipart-config>
            <max-file-size>1073741824</max-file-size>
            <max-request-size>1073741824</max-request-size>
            <file-size-threshold>0</file-size-threshold>
        </multipart-config>
    </servlet>

    <servlet-mapping>
        <servlet-name>dispatcher</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>

    <distributable/>

   <session-config>
      <session-timeout>60</session-timeout>
   </session-config>

</web-app>
```

XML Element	Description
display-name	Application 이름
context-param	Application context의 초기화에 사용할 parameter 정보 전달 contextConfigLocation 파라미터에 Application context를 설정 할 설정파일 위치를 전달
listener	Application context와 관련된 listener 등록
filter	Servlet의 실행 양끝단에서 위치할 Filter 설정
servlet	Servlet 설정 (Spring에서는 DispatcherServlet) contextConfigLocation 파라미터에 Servlet Context를 설정 할 설정파일 위치를 전달
servlet-mapping	Servlet의 URL 매핑

※ classpath: prefix는 어디일까?

Spring - The ResourceLoader

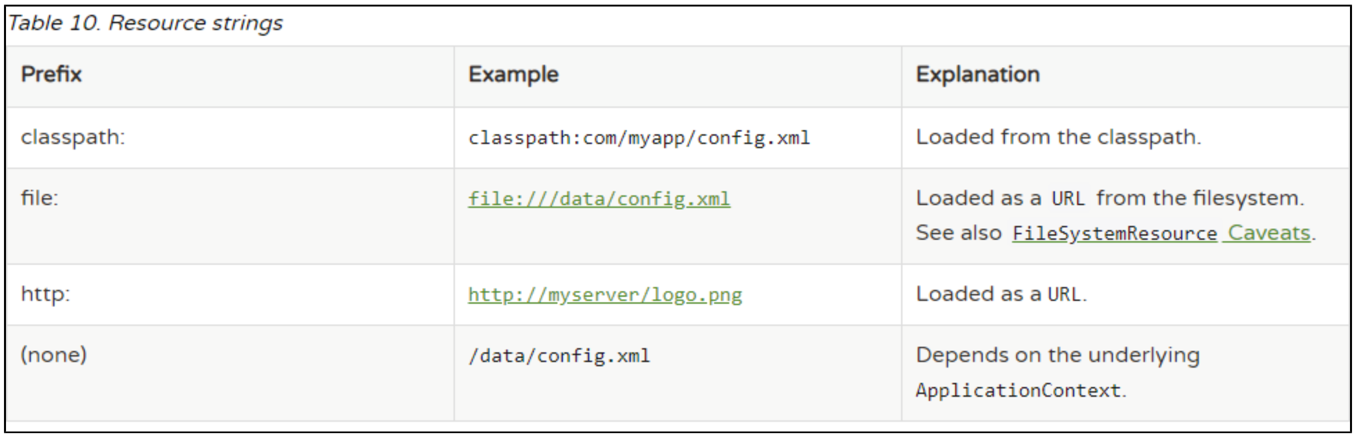
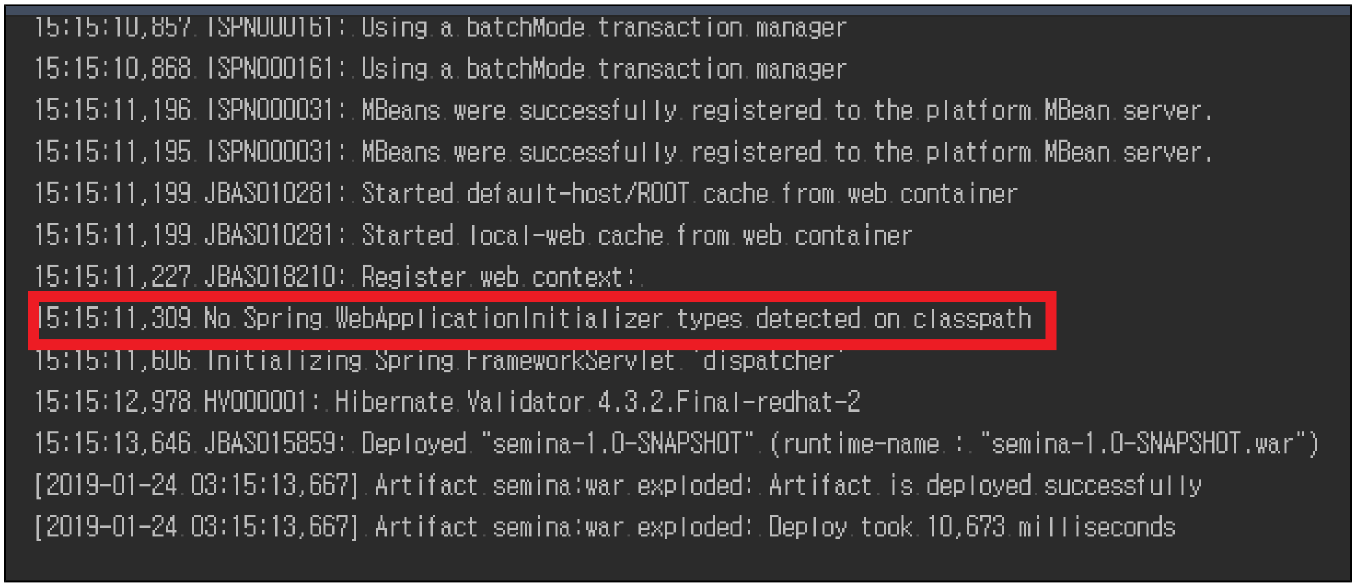


Table 10. Resource strings

Prefix	Example	Explanation
classpath:	`classpath:com/myapp/config.xml`	Loaded from the classpath.
file:	`file:///data/config.xml`	Loaded as a `URL` from the filesystem. See also `FileSystemResource` Caveats.
http:	`http://myserver/logo.png`	Loaded as a URL.
(none)	`/data/config.xml`	Depends on the underlying `ApplicationContext`.

2) Java 기반 설정

Application을 실행하면 아래와 같은 로그를 볼 수 있음



```
15:15:10,857 ISPN000161: Using a batchMode transaction manager
15:15:10,868 ISPN000161: Using a batchMode transaction manager
15:15:11,196 ISPN000031: MBeans were successfully registered to the platform MBean server.
15:15:11,195 ISPN000031: MBeans were successfully registered to the platform MBean server.
15:15:11,199 JBAS010281: Started default-host/ROOT cache from web container
15:15:11,199 JBAS010281: Started local-web cache from web container
15:15:11,227 JBAS018210: Register web context:
15:15:11,309 No Spring WebApplicationInitializer types detected on classpath
15:15:11,606 Initializing Spring FrameworkServlet 'dispatcher'
15:15:12,978 HV000001: Hibernate Validator 4.3.2.Final-redhat-2
15:15:13,646 JBAS015859: Deployed "semina-1.0-SNAPSHOT" (runtime-name : "semina-1.0-SNAPSHOT.war")
[2019-01-24 03:15:13,667] Artifact semina:war exploded: Artifact is deployed successfully
[2019-01-24 03:15:13,667] Artifact semina:war exploded: Deploy took 10,673 milliseconds
```

→ **WebApplicationInitializer**를 이용하면 web.xml 설정을 대체할 수 있음

WebApplicationInitializer 인터페이스 구현하여 설정하기

WebApplicationInitializer 인터페이스 구현

```
public class ApplicationInitializer implements WebApplicationInitializer {
   @Override
   public void onStartup(ServletContext container) {

      XmlWebApplicationContext appContext = new XmlWebApplicationContext();
      appContext.setConfigLocations("/WEB-INF/dispatcher-servlet.xml", "classpath:spring/root-context.xml");


            container.addListener(new ContextLoaderListener(appContext));

      ServletRegistration.Dynamic dispatcher =
            container.addServlet("dispatcher", new DispatcherServlet(appContext));
      dispatcher.setLoadOnStartup(1);
      dispatcher.addMapping("/");


          container.addFilter("My filter", MyFilter.class).addMappingForServletNames(null, false, "dispatcher");
   }
}
```

좀 더 편리하게 WebApplicationInitializer를 구현하고 있는 구현체를 이용하기

AbstractAnnotationConfigDispatcherServletInitializer 상속하여 설정

```
public class ApplicationInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {

   @Override
   protected Class<?>[] getRootConfigClasses() {
      return new Class[]{ApplicationConfig.class};
   }

   @Override
   protected Class<?>[] getServletConfigClasses() {
      return new Class[]{DispatcherServletConfig.class};
   }

   @Override
   protected String[] getServletMappings() {
      return new String[]{"/"};
   }

   @Override
   protected DispatcherServlet createDispatcherServlet(WebApplicationContext servletAppContext) {
      final DispatcherServlet dispatcherServlet = (DispatcherServlet) super.createDispatcherServlet
(servletAppContext);
      dispatcherServlet.setThrowExceptionIfNoHandlerFound(true);
      return dispatcherServlet;
   }

   @Override
   protected Filter[] getServletFilters() {
      return new Filter[] {new CharacterEncodingFilter("UTF-8", true, true)};
   }
}
```

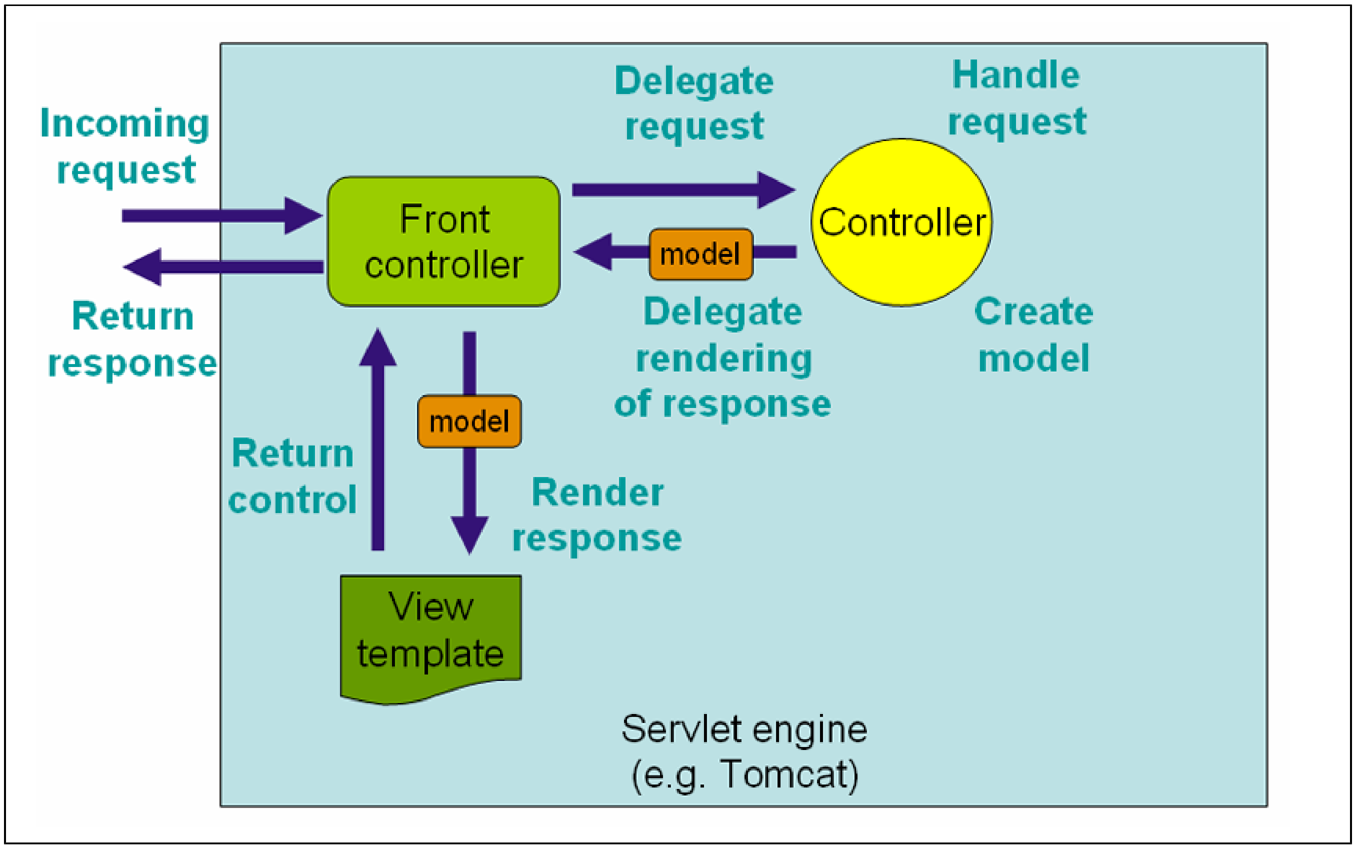
DispatcherServletConfig.java

```
@Configuration        //<beans>~</beans>
@EnableWebMvc        //<mvc:annotation-driven />
@ComponentScan("io.crscube.semina.day07")     //<context:component-scan base-package="io.crscube.semina.day07"
/>
public class DispatcherServletConfig extends WebMvcConfigurerAdapter {
   @Override
   public void addResourceHandlers(ResourceHandlerRegistry registry) {
      registry.addResourceHandler("/resources/**")
            .addResourceLocations("/resources/");
   }

   @Bean
   public InternalResourceViewResolver viewResolver() {
      InternalResourceViewResolver viewResolver
            = new InternalResourceViewResolver();
      viewResolver.setViewClass(JstlView.class);
      viewResolver.setPrefix("/WEB-INF/views/jsp/");
      viewResolver.setSuffix(".jsp");
      return viewResolver;
   }
}
```

(3) 핵심 구성 요소

1) DispatcherServlet

스프링 MVC 구조에서 Front controller를 담당하고 있다



주입받은 여러 개의 **Strategy Bean**들을 통해서 기능을 수행한다

2) @Controller를 위한 HandlerMapping과 HandlerAdapter

HandlerMapping

우리가 작성한 Controller에 기술한 RequestMapping(GetMapping, PostMapping...)정보를 인스턴스화 한 목록이다

DispatcherServlet은 HttpServletRequest를 HandlerMapping 객체에 넘겨 적합한 Handler인지 판단한다

```
protected HandlerExecutionChain getHandler(HttpServletRequest request) throws Exception {
   for (HandlerMapping hm : this.handlerMappings) {
      if (logger.isTraceEnabled()) {
         logger.trace(
               "Testing handler map [" + hm + "] in DispatcherServlet with name '" + getServletName() + "'");
      }
      HandlerExecutionChain handler = hm.getHandler(request);
      if (handler != null) {
         return handler;
      }
   }
   return null;
}
```

HandlerAdapter

DispatcherServlet은 HandlerMapping을 통해 얻은 HandlerExecutionChain 객체를 통해 HandlerAdapter를 반환받는다

직접적인 Handler를 받는 것이 아니라 HandlerAdapter을 반환 받는 이유는 우리가 요청에 따라 작성한 코드의 실행을 handle에서 실행하기 전 후에
필요한 작업을 DispatcherServlet에서 대신 해주기 위함이다

3) HandlerExceptionResolver

Controller 실행 중 발생하는 UnCheckedException에 대해서 처리한다. (Spring에서 무엇인가를 처리하는 클래스의 네이밍 패턴은 XXXResolver 이다)

HandlerExceptionResolver는 예외가 던져졌을 때 어떤 핸들러를 실행할 지를 정하는데 Default로 DispatcherServlet에 등록되 것은 DefaultHandlerExceptionResolver이다.

예외	HTTP 상태 코드
ConversionNotSupportedException	500 (Internal Server Error)
HttpMediaTypeNotAcceptableException	406 (Not Acceptable)
HttpMediaTypeNotSupportedException	415 (Unsupported Media Type)
HttpMessageNotReadableException	400 (Bad Request)
HttpMessageNotWritableException	500 (Internal Server Error)
HttpRequestMethodNotSupportedException	405 (Method Not Allowed)
MissingServletRequestParameterException	400 (Bad Request)
NoSuchRequestHandlingMethodException	404 (Not Found)
TypeMismatchException	400 (Bad Request)

4) ViewResolver, LocaleResolver, MultipartResolver

ViewResolver

Controller에서 Request를 처리하고 생성한 결과물에 대한 View 처리를 담당하는 Strategy Interface이다.

이전에 Spring을 개발할 때 Controller에서 ModelAndView라는 객체를 반환했어야 했다.

```
@Controller
public class ViewController {
   @GetMapping("/home")
   public ModelAndView getHomeModelAndView(ModelAndView modelAndView){
      modelAndView.setViewName("home");
      modelAndView.addObject("user", "Lee");
      return modelAndView;
   }
}
```

따라서 Model은 Model대로 생성하고 View 이름과 관련해서도 별도로 설정이 필요했다.

하지만 최근에는 Model만 설정하고 View 이름만 반환하는 형태의 개발도 가능한데
이것은 InternalResourceViewResolver 라는 Strategy Class에서 컨트롤러가 반환한 View 이름을 가지고
ModelAndView를 알아서 생성하도록 처리를 위임받기 때문이다

```
@GetMapping("/home-view")
public String getHomeView(Model model){
   model.addAttribute("user", "Lee");
   return "home";
}
```

LocaleResolver

웹 기반의 Locale을 결정하는 Strategy Interface이다. Spring에서는 Request, Session, Cookie 등을 기반으로
Locale 관련 정보를 가져오는 구현체가 있다.

AccpetHeaderLocaleResolver, CookieLocaleResolver, FixedLocaleResolver, SessionLocaleResolver 등이 구현체이며
Locale 정보는 LocaleContextHolder라는 ThreadLocal에 담긴다

MultipartResolver

RFC 1867에 따라 multipart 파일 업로드 요청에 대해 해석하여 변환하는 Strategy Interface이다

Spring에서 제공하는 구현체로는 CommonsMultipartResolver, StandardServletMultipartResolver(Servlet 3.0+) 구현체가 있다

(CommonsMultipartResolver는 Apache Commons FileUpload 라이브러리에 의존하고 있다)

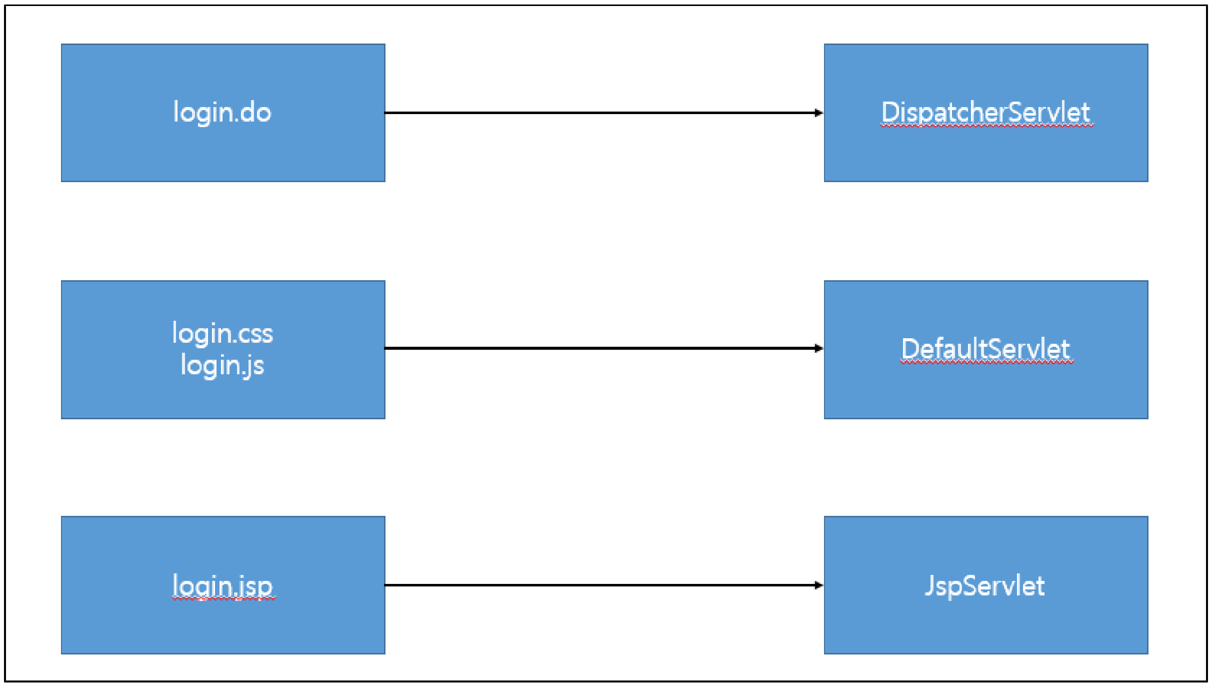
4) DefaultServletHandler

이전에 Java 기반 웹 개발에선 URL에 확장자를 꼭 붙여왔다. 예를들어서 login.do, login.action과 같은 패턴이다.

이는 xxx.jsp를 직접적으로 노출시키지 않기 위한 패턴인데 그에 맞춰서 Spring에서도 사용자의 요청인 *.do url 매핑을 DispatcherServlet에 하였다

보통 Servlet Container는 Static resource(HTML, JS, CSS, Image) 등을 처리하는 Default Servlet과 JSP를 처리하는 JSP Servlt이 기본으로 존재한다.

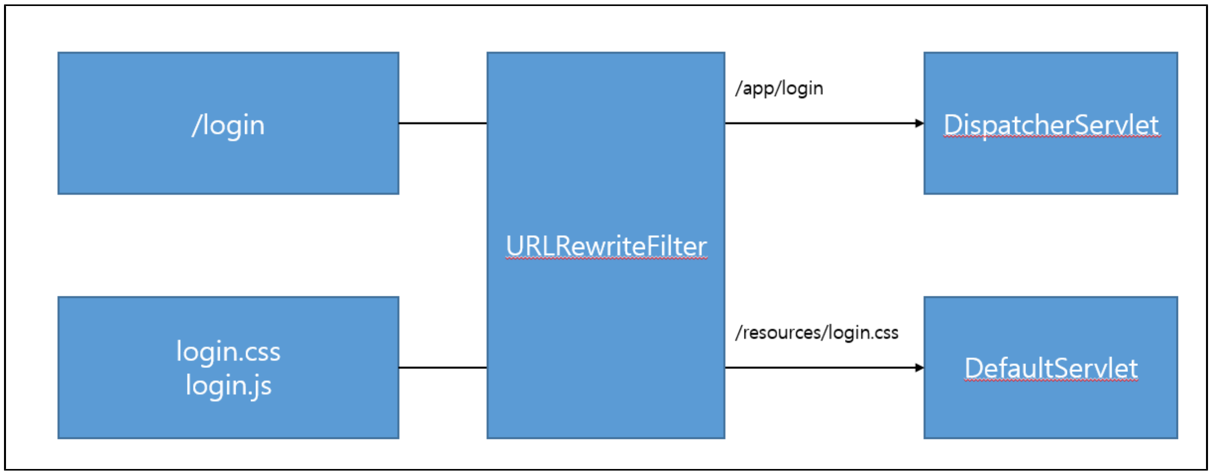
DefaultServlet은 /에 매핑되어있고 jsp는 *.jsp로 되어있는데 여기에 우리가 개발한 DispatcherServlet이 등록되어 URL 우선순위에 따라 요청을 담당하게 된다.



하지만 최근에는 RESTful 스타일이 등장하면서 확장자를 사용하지 않는 추세이다. 즉 login.do, login.action처럼 요청한 것을 /login으로 요청하는 것인데
이렇게 하려면 DispatcherServlet을 /에 매핑해야한다.

즉, DefaultServlet과 DispatcherServlet이 매핑된 URL이 중복되어 문제가 발생하게되고 이에대한 해결법으로 DispatcherServlet을 /app과 같이 매핑하는 꼼수를 사용한다

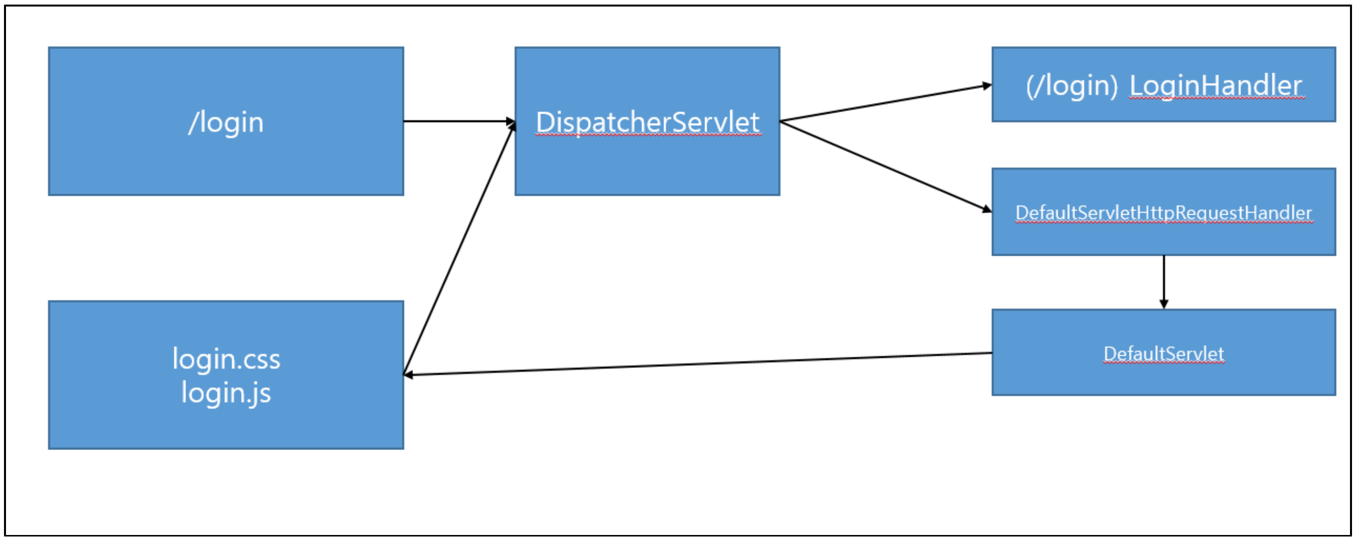
하지만 /app/login, /app/user와 같이 app이 붙는 패턴조차 싫어 UrlRewriteFilter와 같은 것을 사용하여 서버로 온 URL을 애플리케이션에 넘기기 전에 변경하여 넘기도록 하는 것이다.



그런데 **UrlRewriteFilter는 구현체가 복잡하고 사용법도 별도로 학습하여 적용해야 한다.**

Spring에서는 이 문제를 해결하기 위해 DefaultServletHandler를 Spring 3.0.4 버전부터 지원하였는데 DispatcherServlet이 모든 요청을 받아 처리하되
Controller에 매핑되지 않는 URL이 나오면 DefaultServletHttpRequestHandler가 담당하도록 한것이다.

이 핸들러는 /로 매핑되어있으며 대신 우선순위가 가장 낮다. 그리고 넘어온 요청을 자신이 직접 Static resource를 읽어 처리하는 것이 아니라 원래 서버가 제공하는 DefaultServlet으로 넘겨버린다. 그러면 Servlet의 DefaultServlet이 동작하여 Static resource를 처리하도록 깔끔하게 정리가 된다.



DefaultServletHandler는 아래와 같이 resource handler를 설정하면 자동으로 등록된다.

```
<mvc:resources mapping="/favicon.ico" location="/favicon.ico" cache-period="2592000"/>
<mvc:resources mapping="/resources/javascript/library/**" location="/resources/javascript/library/" cache-
period="2592000"/>
<mvc:resources mapping="/resources/css/library/**" location="/resources/css/library/" cache-period="2592000"/>
<mvc:resources mapping="/resources/fusionCharts/**" location="/resources/fusionCharts/" cache-period="2592000"/>
<mvc:resources mapping="/resources/fullCalendar/**" location="/resources/fullCalendar/" cache-period="2592000"/>
<mvc:resources mapping="/resources/image/**" location="/resources/image/" cache-period="36000" />
<mvc:resources mapping="/resources/**" location="/resources/" cache-period="0"/>
```

```
@Override
public void addResourceHandlers(ResourceHandlerRegistry registry) {
   registry.addResourceHandler("/resources/**")
         .addResourceLocations("/resources/");
}
```

(4) Filter

1) 필터의 설정과 커스텀 구현

XML을 이용한 필터 설정

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0"
         xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.
xsd">
    <filter>
        <filter-name>characterSetEncodingFilter</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>
        <init-param>
            <param-name>forceEncoding</param-name>
            <param-value>true</param-value>
        </init-param>
    </filter>
    <filter-mapping>
        <filter-name>characterSetEncodingFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
</web-app>
```

Java를 이용한 필터 설정

```
public class ApplicationInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {
   @Override
   protected Filter[] getServletFilters() {
      return new Filter[] {new CharacterEncodingFilter("UTF-8", true, true)};
   }
}
```

커스텀 필터 구현하기

javax.servlet.Filter 인터페이스를 구현하여 위와같이 등록하면 쉽게 사용 가능하며 구현해야 할 메소드는 아래와 같다

init(FilterConfig)

필터 객체가 생성될 때 호출되는 메소드이다. 필처 객체는 웹 애플리케이션 서비스가 올라가면서, 즉 웹서버가 시작될 때 한번만 되어 생성되어 한번만 호출되며, init( )메소드에는 주로 초기화 기능을 구현한다.

doFilter(ServletRequest request, ServletResponse response)

doFilter( ) 메소드는 필터링 설정한 서블릿을 실행할 때마다 호출되는 메소드로서 실제 필터링 기능을 구현하는 메소드.

destory( )

필터 객체가 삭제 될 때 호출되는 메소드 따라서 destory( ) 메소드에는 주로 자원 해제 기능을 구현.

2) 주 사용처

Character Encoding 설정, XSS 공격 방지 (Lucy-Xss-Servlet Filter), CORS Filter, MultipartFilter 등

DispatcherServlet의 실행 전후로 어떤 처리가 필요할 경우

2. Spring MVC 테스트

우리가 구현한 서비스에 대한 Unit test와 컨트롤러에 대해 Integrated test는 필수이며 여기서는 MockMvc를 통해

우리가 작성한 컨트롤러에 대해 테스트를 진행한다

아래와 같이 간단한 컨트롤러가 있다고 할 때 우리가 테스트할 항목들은 요청을 진행했을 때

응답 상태, 예외 발생 여부, 응답 body의 Assertion이다.

```
@RestController
public class HelloController {
   @GetMapping("/hellos")
   public ResponseEntity<String> getHelloMessage() {
      return ResponseEntity.ok().body("hello");
   }

   @PostMapping("/hellos")
   public ResponseEntity<Map<String, Object>> createHello() {
      Map<String, Object> resultMap = new HashMap<>();
      resultMap.put("hello", "Lee");
      resultMap.put("message", "Request processed successfully");

      return ResponseEntity.status(HttpStatus.CREATED)
                      .body(resultMap);
   }
}
```

따라서 아래와 같이 테스트를 작성할 수 있다.

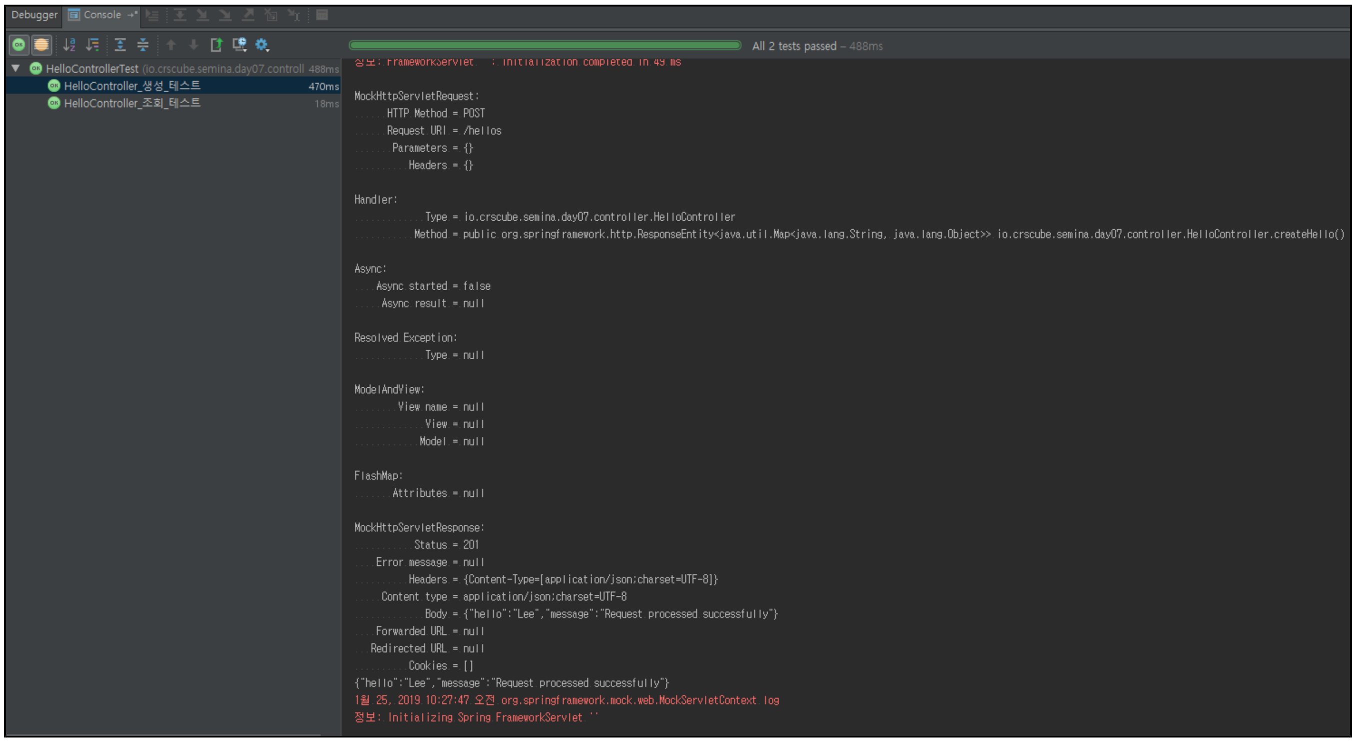
```
@RunWith(SpringJUnit4ClassRunner.class)
@WebAppConfiguration
@ContextConfiguration(classes = {ApplicationConfig.class, DispatcherServletConfig.class})
public class HelloControllerTest {

   @Autowired
   private WebApplicationContext webApplicationContext;

   private ObjectMapper objectMapper;

   private MockMvc mockMvc;

        // @Test
   @Before
   public void onBeforeTest() {
      this.mockMvc = MockMvcBuilders.webAppContextSetup(this.webApplicationContext).build();
      this.objectMapper = new ObjectMapper();
   }

   @Test
   public void HelloController__() throws Exception {
      //Given
      MockHttpServletRequestBuilder requestBuilder = MockMvcRequestBuilders.get("/hellos");

      //When
      ResultActions perform = mockMvc.perform(requestBuilder);

      //Then
      //Check Response status
      MockHttpServletResponse response = perform
            .andExpect(status().isOk())          //HttpStatus 200
            .andDo(print())
            .andReturn()
            .getResponse();


      //Check response body
      String bodyToText = response.getContentAsString();
      assertThat(bodyToText).isEqualTo("hello");

      System.out.println(bodyToText);
   }

   @Test
   public void HelloController__() throws Exception {
      //Given
      MockHttpServletRequestBuilder requestBuilder = MockMvcRequestBuilders.post("/hellos");

      //When
      ResultActions perform = mockMvc.perform(requestBuilder);

      //Then
      //Check Response status
      MockHttpServletResponse response = perform
            .andExpect(status().isCreated())      //HttpStatus 201
            .andDo(print())
            .andReturn()
            .getResponse();

      //Check response body
      String bodyToText = response.getContentAsString();
      System.out.println(bodyToText);

      Map map = objectMapper.readValue(bodyToText, Map.class);

      assertThat(map.get("hello")).isEqualTo("Lee");
      assertThat(map.get("message")).isEqualTo("Request processed successfully");
   }
}
```

- 테스트 메소드 이름은 한글로 작성
- 테스트 메소드 내에 구현은 **Given, When, Then** 패턴을 따를 것
    - Given, When, Then 템플릿에 익숙하다면 **Spock**의 사용을 고려

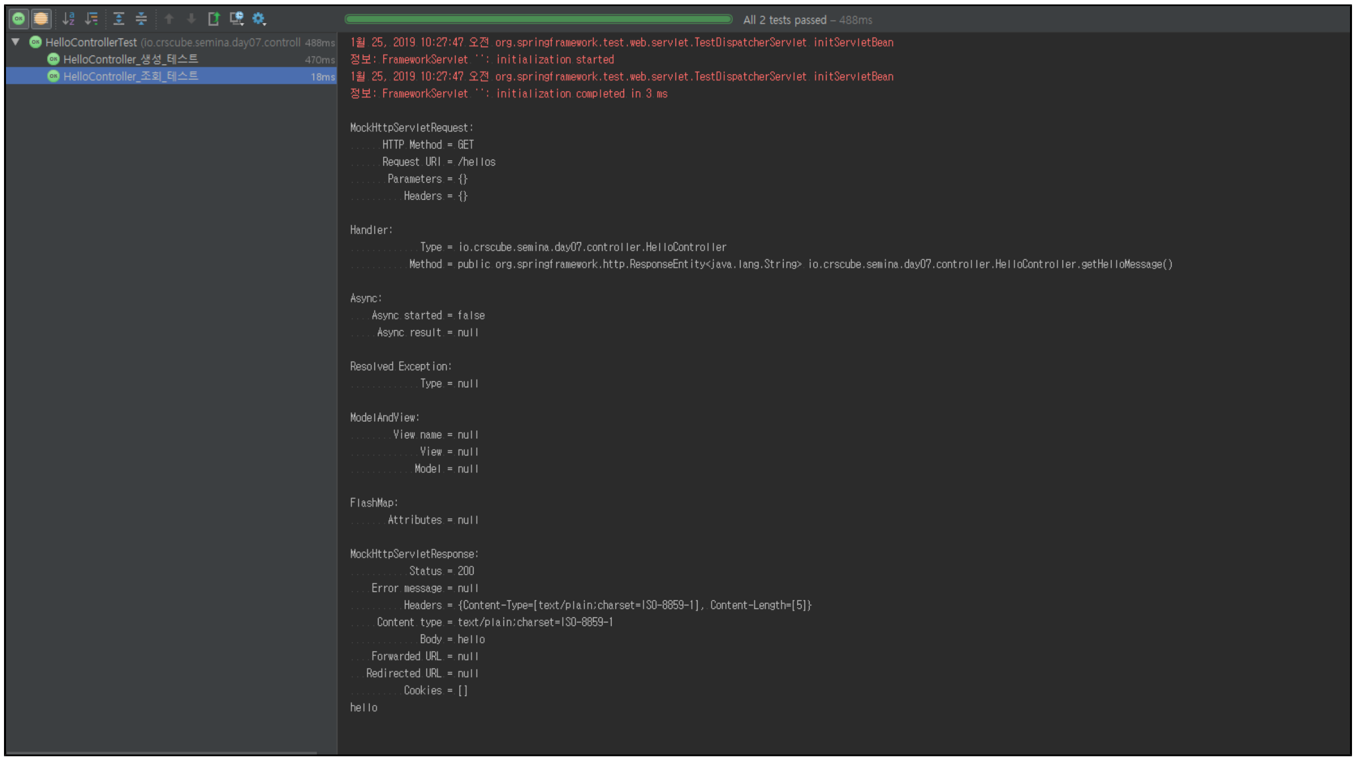
```
def "          "() {
    setup:
    def email = "test@mail"
    def user = new User()
    user.email = email


    def mockUserRepo = Mock(UserRepository.class)
    def service = new PylonUserDetailsService(mockUserRepo)


    when:
    def userDetails = service.loadUserByUsername(email)


    then:
    mockUserRepo.findByEmail(email) >> user
    userDetails instanceof PylonUser
    userDetails.username == "whiteship"
}
```

- 주로 체크할 항목은 Response의 Http Status, Expected Exception, Response body의 검증
- Java8 이상을 사용한다면 JUnit 5이상의 버전을 사용
- Assertion 관련 AssertJ 라이브러리의 사용을 고려

테스트 결과


Debugger
Console
HelloControllerTest (io.crscube.semina.day07.controll 488ms
HelloController_생성_테스트 470ms
HelloController_조회_테스트 18ms
All 2 tests passed – 488ms

```
정보: FrameworkServlet '': initialization completed in 49 ms

MockHttpServletRequest:
      HTTP Method = POST
      Request URI = /hellos
       Parameters = {}
          Headers = {}

Handler:
             Type = io.crscube.semina.day07.controller.HelloController
           Method = public org.springframework.http.ResponseEntity<java.util.Map<java.lang.String, java.lang.Object>> io.crscube.semina.day07.controller.HelloController.createHello()

Async:
    Async started = false
     Async result = null

Resolved Exception:
             Type = null

ModelAndView:
        View name = null
             View = null
            Model = null

FlashMap:
       Attributes = null

MockHttpServletResponse:
           Status = 201
    Error message = null
          Headers = {Content-Type=[application/json;charset=UTF-8]}
     Content type = application/json;charset=UTF-8
             Body = {"hello":"Lee","message":"Request processed successfully"}
    Forwarded URL = null
   Redirected URL = null
          Cookies = []
{"hello":"Lee","message":"Request processed successfully"}
1월 25, 2019 10:27:47 오전 org.springframework.mock.web.MockServletContext log
정보: Initializing Spring FrameworkServlet ''
```


All 2 tests passed – 488ms
HelloControllerTest (io.crscube.semina.day07.controll 488ms
HelloController_생성_테스트 470ms
HelloController_조회_테스트 18ms

```
1월 25, 2019 10:27:47 오전 org.springframework.test.web.servlet.TestDispatcherServlet initServletBean
정보: FrameworkServlet '': initialization started
1월 25, 2019 10:27:47 오전 org.springframework.test.web.servlet.TestDispatcherServlet initServletBean
정보: FrameworkServlet '': initialization completed in 3 ms

MockHttpServletRequest:
      HTTP Method = GET
      Request URI = /hellos
       Parameters = {}
          Headers = {}

Handler:
             Type = io.crscube.semina.day07.controller.HelloController
           Method = public org.springframework.http.ResponseEntity<java.lang.String> io.crscube.semina.day07.controller.HelloController.getHelloMessage()

Async:
    Async started = false
     Async result = null

Resolved Exception:
             Type = null

ModelAndView:
        View name = null
             View = null
            Model = null

FlashMap:
       Attributes = null

MockHttpServletResponse:
           Status = 200
    Error message = null
          Headers = {Content-Type=[text/plain;charset=ISO-8859-1], Content-Length=[5]}
     Content type = text/plain;charset=ISO-8859-1
             Body = hello
    Forwarded URL = null
   Redirected URL = null
          Cookies = []
hello
```

막간의 이야기

Spring Framework의 구조와 구현체들을 살펴보면 Adapter Pattern, Front Controller Pattern, Strategy Pattern, Delegate Pattern 등

여러 디자인 패턴을 살펴볼 수 있습니다.

이러한 패턴은 Back-end framework 뿐 아니라 Front-end framework에서도 살펴볼 수 있는데요

Javascript에서 주로 사용하는 Design pattern 중 Revealing Module Pattern 패턴이 있습니다

Javascript에 Class라는 개념이 없는 시절 많은 개발자들이 전역변수로와 복잡한 호이스팅, Closure 등으로 인해 ~~고통받..~~

힘든 유지보수를 이어갈 때 등장한 패턴으로 Javascript의 Object에 private, public 등의 개념을 부여하는 패턴입니다

설명하기에 앞서 몇가지 살펴볼 개념이 있습니다

Object Literals(객체 리터럴)이란 Javascript에서 사용하는 키/값 형태로 중괄호에 선언하는 객체입니다

```
var User = {
    userKey: undefined,
    userName: undefined,

    getUserName: function(){
        return this.userName;
    }
};

User.userName = "Lee";
User.getUserName();         //Lee
```

이러한 객체리터럴을 이용해서 ES6 이하에서 지원하지 않는 클래스의 컨셉을 모방할 수 있습니다.

prototype이라는 것을 통해서 상속을 모방하듯이...

Module pattern은 이러한 객체리터럴을 통해 코드의 캡슐화, 구조화를 이루고자 하는 방식입니다.

```
var Renderer = (function () {
        'use strict';

    function rendering() {
        console.log("rendering");
    }

    return {
        rendering: rendering
    };
})();

var CommonGrid = (function (renderer) {
    'use strict';

    var config = {
        supplyPaging: true,
        defaultPageNum: 1,
        defaultRowSize: 10
    };

    var context = {
        currentPageNum: config.defaultPageNum,
        currentRowSize: config.defaultRowSize
    };

    var events = {
        onInit: function (config) {
            if (config['pageNum']) {
                context.currentPageNum = config.pageNum;
            }
            if (config['rowSize']) {
                context.currentRowSize = config.rowSize;
            }

            //Call api...
            rendering();
        },
        onPaging: function (pageNum) {
            context.currentPageNum = pageNum;
            rendering();
        },
        onRowSizeChanged: function (rowSize) {
            context.currentRowSize = rowSize;
            rendering();
        }
    };

    var init = function (config) {
        events.onInit(config);
    };

    var rendering = function () {
        //Before rendering logic
        renderer.rendering();
        //After rendering logic
    };

    return {
        init: init,
        appliedConfig: {
            defaultPageNum: config.defaultPageNum,
            defaultRowSize: config.defaultRowSize
        }
    };
})(Renderer);

CommonGrid.init({
    pageNum: 1,
    rowSize: 100
});
```

Import

위의 코드를 잠시 살펴보면 CommonGrid에 Renderer를 주입하는 것을 볼 수 있습니다.

이처럼 우리가 의존할 대상을 전달하고 모듈 내에서 지역화된 이름으로 사용할 수있습니다.

버그로 인한 라이브러리 교체에 따른 Site effect를 줄일 수 있는 것이죠

대표적인 예로 아래와 같이 jQuery, Underscore를 주입하는 코드입니다.

```
var myModule = (function ($, _) {
    function privateMethod1() {
        $(".container").html("test");
    }

    function privateMethod2() {
        console.log(_.min([10, 5, 100, 2, 1000]));
    }

    return {
        publicMethod: function () {
            privateMethod1();
        }
    }; // Pull in jQuery and Underscore
})( jQuery, _ );
```

Export

예제코드를 다시 살펴보면 모듈에서 return문에 명시한 것만 접근이 가능합니다.

따라서 Global scope를 어지럽히지 않고 Global variable을 선언하는 방법이라고 할 수 있습니다.

각 변수와 함수가 캡슐화 된다는 것이죠

이러한 모듈패턴의 단점은 private 멤버에 대한 자동화 테스트를 진행할 수 없고 그에따라 버그가 있을 수 있다라는 정도의 단점이 있습니다.

대표적으로 잘 사용하고 있는 예는 Mozilla의 pdf.js입니다.