|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| http://www.kame.co.kr/board/down/book/1408698160_spirng4-web.jpg | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **웹 개발자를 위한 Spring 4.0 프로그래밍**  http://www.kame.co.kr/images/subtitle/sub01_bt04.gif   |  |  | | --- | --- | | **▶** 저자 | - 최범균 | | **▶** 서적구분 | - 국내서 | | **▶** 발행일 | - 2014-07-30 | | **▶** 정가 | - **32,000** | | **▶** 페이지수 | - 806 | | **▶** 분류 | - 언어/프로그래밍 | | **▶** 판매척도 | - | |  | [http://www.kame.co.kr/images/subtitle/sub01_bt03.gif](http://www.kame.co.kr/book_main.html?board=kame_book&work=read&tcode=280&tbook_top=3&tbook_jong=#bottom3) | | |
| http://www.kame.co.kr/images/subtitle/sub04_line01.gif | |
| 웹 개발자를 위한 Spring 4.0 프로그래밍  - 스프링 4의 새로운 특징 포함  - 스프링 DI, AOP  - 스프링 MVC  - 스프링 웹소켓  - JDB, 하이버네이트, JPA, MyBatis 연동  - 스프링 데이터 JPA  - 일반적인 웹 어플리케이션 구조 소개  - 스프링 시큐리티를 이용한 웹 보안 구현  - 메일 발송, 스케쥴링, RestTemplates  - 스프링을 이용한 JUnit 테스트  ■ 소스 자료는 가메출판사 홈페이지http://www.kame.co.kr 자료실에서다운로드합니다.    상세 목차    PART 01 스프링 기초  Chapter 01 스프링 시작하기  01 스프링 프레임워크  02 설치 및 주요 모듈  03 첫 번째 스프링 프로그램  Chapter 02 스프링 DI를 이용한 객체 생성  01 DI(Dependency Injection)와 스프링  02 스프링 컨테이너 종류  03 스프링 DI 설정  04 팩토리 방식의 스프링 빈 설정  05 애노테이션을 이용한 객체 간 의존 자동 연결 1  06 컴포넌트 스캔을 이용한 빈 자동 등록 1  07 스프링 컨테이너 추가 설명  Chapter 03 빈 라이프사이클과 빈 범위  01 빈 객체의 라이프사이클  02 빈 객체 범위(scope)  Chapter 04 Environment, 프로퍼티, 프로필, 메시지  01 Environment 소개  02 Environment와 PropertySource  03 Environment를 스프링 빈에서 사용하기  04 프로퍼티 파일을 이용한 프로퍼티 설정  05 프로필을 이용한 설정  06 MessageSource를 이용한 메시지 국제화 처리  Chapter 05 확장 포인트와 PropertyEditor/ConversionService  01 스프링 확장 포인트  02 PropertyEditor와 ConversionService  Chapter 06 스프링 AOP  01 AOP 소개  02 스프링에서의 AOP  03 XML 스키마 기반 AOP 퀵 스타트  04XML 스키마 기반의 POJO 클래스를 이용한 AOP 구현  05 @Aspect 애노테이션 기반 AOP 퀵 스타트  06 @Aspect 애노테이션을 이용한 AOP  07 JoinPoint 사용  08 타입을 이용한 파라미터 접근  09 AOP 프록시 객체 생성 방식 설정  10 AspectJ의 Pointcut 표현식  11 Advice 적용 순서  PART 02 스프링 웹 개발 1  Chapter 07 스프링 MVC : 기본기  01 스프링 MVC 일단 해보기  02 기본 흐름과 주요 컴포넌트  03 스프링 MVC 설정 기초  04 컨트롤러 구현  05 커맨드 객체 값 검증과 에러 메시지  06 요청 파라미터의 값 변환 처리  07 HTTP 세션 사용하기  08 익셉션 처리  09 컨트롤러 메서드의 파라미터 타입과 리턴 타입  10 스프링 MVC 설정  11 HandlerInterceptor를 이용한 인터셉터 구현  12 WebApplicationContext 계층  13 DelegatingFilterProxy를 이용한 서블릿 필터 등록  14 핸들러, HandlerMapping, HandlerAdapter  Chapter 08 스프링 MVC : 뷰 영역  01 ViewResolver 설정  02 HTML 특수 문자 처리 방식 설정  03 JSP를 이용한 뷰 구현  04 HTML 이외의 뷰 구현  05 Locale 처리  Chapter 09 스프링 MVC : XML/JSON, 파일 업로드, 웹소켓  01 XML/JSON 변환 처리  02 파일 업로드  03 웹소켓 서버 구현 지원  Chapter 10 스프링MVC : 기타 설정  01 서블릿 3 기반 설정  PART 03 데이터베이스 연동  Chapter 11 데이터베이스 연동 지원과 JDBC 지원  01 스프링의 데이터베이스 연동 지원  02 DataSource 설정  03 스프링 JDBC 지원  Chapter 12 스프링의 트랜잭션 관리  01 트랜잭션이란  02 스프링의 트랜잭션 지원  03 TransactionTemplate을 이용한 트랜잭션  04 트랜잭션과 DataSource  05 선언적 트랜잭션 처리  06 TransactionsEssentials를 이용한 분산 트랜잭션  Chapter 13 ORM 연동 지원  01 @Repository 애노테이션을 이용한 익셉션 변환 처리  02 하이버네이트 연동 지원  03 JPA 연동 지원(하이버네이트 4 기준)  04 MyBatis 연동 지원  Chapter 14 스프링 데이터 JPA 소개  01 스프링 데이터 JPA 시작하기  02 리파지터리 인터페이스 메서드 작성 규칙  03 @Query를 이용한 JPQL/네이티브 쿼리 사용  04 Specification을 이용한 검색 조건 표현  05 기본 제공 인터페이스  06 커스텀 구현 추가하기  PART 04 스프링 웹 개발 2  Chapter 15 웹 어플리케이션 구조  01 서비스-DAO 구조  02 어플리케이션-도메인-영속성 구조  Chapter 16 스프링 시큐리티를 이용한 웹 보안  01 웹 보안과 스프링 시큐리티  02 스프링 시큐리티 퀵 스타트  03 스프링 시큐리티 구조 개요  04 웹 요청 인가 설정 표현식  05 상황별 스프링 시큐리티 설정  06 JSP 태그 라이브러리  PART 05 기타 기능  Chapter 17 메일 발송, 작업 실행과 스케줄링, RestTemplate  01 메일 발송  02 작업 실행과 스케줄링  03 RestTemplate을 이용한 HTTP 클라이언트 구현  Chapter 18 스프링 테스트 지원  01 메이븐 의존 설정  02 JUnit4의 스프링 테스트 통합 테스트  03 스프링 MVC 테스트  PART 06 부록  부록 A 로깅 의존 설정  01 JCL과 Log4j 사용 설정  02 JCL 대신 SLF4J 사용하기 | |
|  | |
|  | |
|  | |
|  | |
|  | |
|  | |

<https://github.com/madvirus/spring4>

GenericXmlApplicationContext 통한 일반적인 처리

|  |
| --- |
| **public** **class** Main {  **public** **static** **void** main(String[] args) {  String configLocation = "classpath:applicationContext.xml";  AbstractApplicationContext ctx =  **new** GenericXmlApplicationContext(configLocation);  Project project = ctx.getBean("sampleProject", Project.**class**);  project.build();  ctx.close();  }  } |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans.xsd"*>  <bean id=*"mvnBuildRunner"*  class=*"net.madvirus.spring4.appa.MavenBuildRunner"*>  <property name=*"mavenPath"*>  <value>c:\apache-maven-3.1.1</value>  </property>  </bean>  <bean id=*"sampleProject"*  class=*"net.madvirus.spring4.appa.Project"*>  <property name=*"srcDirs"*>  <list>  <value>src/</value>  <value>srcResources/</value>  </list>  </property>  <property name=*"binDir"* value=*"bin/"* />  <property name=*"buildRunner"*>  <ref bean=*"mvnBuildRunner"* />  </property>  </bean>  </beans> |

|  |  |
| --- | --- |
| BeanFactory | ApplicationContext |
|  |  |
|  | GenericXMLApplicationContext 독립형  AnnotationConfigApplicationContext 어노테이션이영 독립  GenericGroovyApplication 구루비 언어 작성된..독립어플  XMLWebApplicationContet 웹개발  AnnotationConfigWebApplicationContext 웹개발 |

XML설정

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans.xsd"*>  <bean id=*"user1"* class=*"net.madvirus.spring4.chap02.User"*>  <constructor-arg value=*"bkchoi"* />  <constructor-arg value=*"1234"* />  </bean>    <bean id=*"user2"* class=*"net.madvirus.spring4.chap02.User"*>  <constructor-arg value=*"madvirus"* />  <constructor-arg value=*"qwer"* />  </bean>    <bean id=*"userRepository"* class=*"net.madvirus.spring4.chap02.UserRepository"*>  <property name=*"users"*>  **<list>**  **<ref bean=*"user1"* />**  **<ref bean=*"user2"* />**  **</list>**  </property>  </bean>  <bean id=*"pwChangeSvc"*  class=*"net.madvirus.spring4.chap02.PasswordChangeService"*>  <constructor-arg><ref bean=*"userRepository"*/></constructor-arg>  </bean>    <bean id=*"authFailLogger"* class=*"net.madvirus.spring4.chap02.AuthFailLogger"*>  <property name=*"threshold"* value=*"2"* />  </bean>  <bean id=*"authenticationService"*  class=*"net.madvirus.spring4.chap02.AuthenticationService"*>  <property name=*"failLogger"* ref=*"authFailLogger"* />  <property name=*"userRepository"* ref=*"userRepository"* />  </bean>  </beans> |
| <bean id=*"ecFactory"* class=*"net.madvirus.spring4.chap02.erp.ErpClientFactory"*  **factory-method=*"instance"***>  <constructor-arg>  <props>  <prop key=*"server"*>10.50.0.101</prop>  </props>  </constructor-arg>  </bean>  <bean id=*"user1"* class=*"net.madvirus.spring4.chap02.User"*  **c:id=*"bkchoi"* c:password=*"1234"*** />    <bean id=*"user2"* class=*"net.madvirus.spring4.chap02.User"*  **c:\_0=*"madvirus"* c:\_1=*"qwer"*** />  <bean id=*"pwChangeSvc"*  class=*"net.madvirus.spring4.chap02.PasswordChangeService"*  c:userRepository-ref=*"userRepository"* /> |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans.xsd"*>  <bean id=*"sensor1"* class=*"net.madvirus.spring4.chap02.sensor.Sensor"*>  <property name=*"additionalInfo"*>  **<props>**  **<prop key=*"threshold"*>1500</prop>**  **<prop key=*"retry"*>3</prop>**  **</props>**  </property>  </bean>  <bean id=*"sensor2"* class=*"net.madvirus.spring4.chap02.sensor.Sensor"*>  <property name=*"agentCodes"*>  **<set>**  **<value>200</value>**  **<value>300</value>**  **</set>**  </property>  <property name=*"additionalInfo"*>  **<value>**  **threshold = 3000**  **retry = 5**  **</value>**  </property>  </bean>  <bean id=*"monitor"* class=*"net.madvirus.spring4.chap02.sensor.Monitor"*>  <property name=*"sensorMap"*>  **<map>**  **<entry>**  **<key>**  **<value>frontDoor</value>**  **</key>**  **<ref bean=*"sensor1"* />**  **</entry>**  **<entry key=*"backDoor"* value-ref=*"sensor2"* />**  **</map>**  </property>  <property name=*"config"*>  **<map>**  **<entry key=*"interval"* value=*"1000"* />**  **<entry key=*"period"*>**  **<value>2000</value>**  **</entry>**  **</map>**  </property>  </bean>  </beans> |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans.xsd"*>  **<import resource=*"classpath:/config-sensor.xml"* />**  **<import resource=*"classpath:/config.xml"* />**  </beans> |
| @Configuration  @ImportResource("classpath:config-sensor.xml")  **public** **class** ConfigWithXmlImport { |
| GenericXmlApplicationContext ctx =  **new** GenericXmlApplicationContext("classpath:config-with-\*.xml"); |
| AnnotationConfigApplicationContext ctx =  **new** AnnotationConfigApplicationContext("net.madvirus.spring4.chap02.conf");  AnnotationConfigApplicationContext ctx =  **new** AnnotationConfigApplicationContext(Config.**class**);  **@Configuration**  **public** **class** Config {  **@Bean**  **public** User user1() {  **return** **new** User("bkchoi", "1234");  }  @Bean(name = "user2")  **public** User user() {  **return** **new** User("madvirus", "qwer");  }  @Bean  **public** UserRepository userRepository() {  UserRepository userRepo = **new** UserRepository();  userRepo.setUsers(Arrays.*asList*(user1(), user()));  **return** userRepo;  }  @Bean  **public** PasswordChangeService pwChangeSvc() {  **return** **new** PasswordChangeService(userRepository());  }  @Bean  **public** AuthFailLogger authFailLogger() {  AuthFailLogger logger = **new** AuthFailLogger();  logger.setThreshold(2);  **return** logger;  }  @Bean  **public** AuthenticationService authenticationService() {  AuthenticationService authSvc = **new** AuthenticationService();  authSvc.setFailLogger(authFailLogger());  authSvc.setUserRepository(userRepository());  **return** authSvc;  }  } |
|  |
| List : java.util.ArrayList  Map : java.util.LinkedHashMap  Set : java.util.LinkedHashSet  Props : java.util.Properties |

메소드 호출해서 의존을 설정할 때 새로운 객체가 생성되지 않는 이유

실제 스프링은 하위 클래스를 생성할 때 CGLQIB라는 도구를 이용해서 런타임에 하위클래스를 생성한다 하위 클래스를 만들고 메서드를 재정의 하기 때문에 @configration클래스와 @bean메서드는 final이면 아니된다

또한 @bean메서도는 하위 클래스에서 재정의 할수 있도록 pricate이면안된다

어노테이션

|  |
| --- |
| <context:annotation-config /> |
| @Autowired |
| @Resoure |
| @Inject |

|  |
| --- |
| @Autowired |
| @Component("orderSvc")  **public** **class** OrderService {  **private** ErpClientFactory erpClientFactory;  **private** SearchClientFactory searchClientFactory;  @Inject  **public** **void** setErpClientFactory(ErpClientFactory erpClientFactory) {  **this**.erpClientFactory = erpClientFactory;  }  @Autowired  **public** **void** setSearchClientFactory(@Qualifier("order") SearchClientFactory searchClientFactory) {  **this**.searchClientFactory = searchClientFactory;  }  **public** **void** order(OrderInfo oi) {  sendOrderInfoToErp(oi);  addOrderInfoToSearch(oi);  }  **private** **void** sendOrderInfoToErp(OrderInfo oi) {  ErpClient erpClient = erpClientFactory.create();  erpClient.connect();  erpClient.sendPurchaseInfo(toErpOrderData(oi));  erpClient.close();  }  **private** ErpOrderData toErpOrderData(OrderInfo oi) {  **return** **new** ErpOrderData();  }  **private** **void** addOrderInfoToSearch(OrderInfo oi) {  searchClientFactory.create().addDocument(toSearchDocument(oi));  }  **private** SearchDocument toSearchDocument(OrderInfo oi) {  **return** **new** SearchDocument();  }  } |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  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"*>  <context:annotation-config />  <bean id=*"productService"* class=*"net.madvirus.spring4.chap02.shop.ProductService"*>  </bean>  **<bean id=*"orderService"* class=*"net.madvirus.spring4.chap02.shop.OrderService"*>**  **</bean>**  <bean id=*"orderSearchClientFactory"*  class=*"net.madvirus.spring4.chap02.search.SearchClientFactoryBean"*>  <qualifier value=*"order"* />  <property name=*"server"* value=*"10.20.30.40"* />  <property name=*"port"* value=*"8888"* />  <property name=*"contentType"* value=*"json"* />  </bean>  <bean id=*"productSearchClientFactory"*  class=*"net.madvirus.spring4.chap02.search.SearchClientFactoryBean"*>  <property name=*"server"* value=*"10.20.30.41"* />  <property name=*"port"* value=*"9999"* />  <property name=*"contentType"* value=*"json"* />  </bean>  <bean id=*"searchServiceHealthChecker"*  class=*"net.madvirus.spring4.chap02.search.SearchServiceHealthChecker"*>  <property name=*"factories"*>  <list>  <ref bean=*"orderSearchClientFactory"* />  <ref bean=*"productSearchClientFactory"* />  </list>  </property>  </bean>  <bean id=*"ecFactory"* class=*"net.madvirus.spring4.chap02.erp.ErpClientFactory"*  factory-method=*"instance"*>  <constructor-arg>  <props>  <prop key=*"server"*>10.50.0.101</prop>  </props>  </constructor-arg>  </bean>  </beans> |
| 자동으로 @Autowired 어노테이션된것은 config에서 적절한 클래스를 자동으로 인젝션시켜준다 필드에도 @Autowired처리도가능하다 |

|  |  |
| --- | --- |
| @Qualifer | @Autowired에 사용하는 코드는 @Qualifer어노테이션을 이용해서 사용할 의존 객체를 한정지을수 있다.  타입기준 빈객체 설정 |
| @Resource | 의존 자동설정 이름을 기준으로 빈객체를 선택한다 |
| @Inject | 인젝션 사용할 빈이 반드시 있어야 한다 |
| @Configuration | 여러 클래스에 빈정보를 나눠서 설정할수 있다 |

스캔을 통한 자동등록

|  |  |
| --- | --- |
| @Component | **@Component //** @Component("orderSvc") 🡨처럼 이름정할수도있음  **public** **class** ProductService {  **private** SearchClientFactory searchClientFactory;  **@Resource(name = "productSearchClientFactory")**  **public** **void** setSearchClientFactory(SearchClientFactory searchClientFactory) {  **this**.searchClientFactory = searchClientFactory;  }  **public** **void** createProduct(ProductInfo pi) {  searchClientFactory.create().addDocument(toSearchDocument(pi));  }  **private** SearchDocument toSearchDocument(ProductInfo pi) {  **return** **new** SearchDocument();  }  } |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  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"*>  **<context:component-scan base-package=*"net.madvirus.spring4.chap02.shop"/*>**  <bean id=*"orderSearchClientFactory"*  class=*"net.madvirus.spring4.chap02.search.SearchClientFactoryBean"*>  <qualifier value=*"order"* />  <property name=*"server"* value=*"10.20.30.40"* />  <property name=*"port"* value=*"8888"* />  <property name=*"contentType"* value=*"json"* />  </bean>  <bean id=*"productSearchClientFactory"*  class=*"net.madvirus.spring4.chap02.search.SearchClientFactoryBean"*>  <property name=*"server"* value=*"10.20.30.41"* />  <property name=*"port"* value=*"9999"* />  <property name=*"contentType"* value=*"json"* />  </bean>  <bean id=*"searchServiceHealthChecker"*  class=*"net.madvirus.spring4.chap02.search.SearchServiceHealthChecker"*>  <property name=*"factories"*>  <list>  <ref bean=*"orderSearchClientFactory"* />  <ref bean=*"productSearchClientFactory"* />  </list>  </property>  </bean>  <bean id=*"ecFactory"* class=*"net.madvirus.spring4.chap02.erp.ErpClientFactory"*  factory-method=*"instance"*>  <constructor-arg>  <props>  <prop key=*"server"*>10.50.0.101</prop>  </props>  </constructor-arg>  </bean>  </beans> | |
| 지정 패키지 및 그 하위 패키지에 위치한 클래스중 @component어노테이션 적용된 클래스를 검색해서 빈에 등록한다.  BeanPostProcessor를 함께 등록해준다 따라서 @Component, @Required, @Autowired,@Inject 같이 처리  **<context:component-scan base-package=*"net.madvirus.spring4.chap02.shop"*>**  **<context:include-filer type=’regex’ expression=”.\*Ser”/>**  **<context:exclude-filer type=’aspectj’ expression=”.\*Ser”/>**  **</context:component-scan>**  처럼 포함 미포함 처리할수있다.  Type :annotation, assignable, regex, aspectj | |

자바통한 스캔

|  |
| --- |
| @Configuration  **@ComponentScan(basePackages = "net.madvirus.spring4.chap02.shop",**  **includeFilters = {@Filter(type = FilterType.*REGEX*, pattern = ".\*Service")},**  **excludeFilters = @Filter(type = FilterType.*ASPECTJ*, pattern = "net..\*Dao")**  )  **public** **class** ConfigScan {  @Bean  @Qualifier("order")  **public** SearchClientFactoryBean orderSearchClientFactory() {  SearchClientFactoryBean searchClientFactoryBean = **new** SearchClientFactoryBean();  searchClientFactoryBean.setServer("10.20.30.40");  searchClientFactoryBean.setPort(8888);  searchClientFactoryBean.setContentType("json");  **return** searchClientFactoryBean;  }  @Bean  **public** SearchClientFactoryBean productSearchClientFactory() {  SearchClientFactoryBean searchClientFactoryBean = **new** SearchClientFactoryBean();  searchClientFactoryBean.setServer("10.20.30.41");  searchClientFactoryBean.setPort(9999);  searchClientFactoryBean.setContentType("json");  **return** searchClientFactoryBean;  }  @Bean  **public** ErpClientFactory erpClientFactory() {  Properties props = **new** Properties();  props.setProperty("server", "10.50.0.101");  **return** ErpClientFactory.*instance*(props);  }  @Bean  **public** SearchServiceHealthChecker searchServiceHealthChecker(  SearchClientFactory orderSearchClientFactory,  SearchClientFactory productSearchClientFactory  ) **throws** Exception {  SearchServiceHealthChecker healthChecker = **new** SearchServiceHealthChecker();  healthChecker.setFactories(Arrays.*asList*(  orderSearchClientFactory, productSearchClientFactory  ));  **return** healthChecker;  }  } |

스프링 컨테이너의 생성과 종료

1. 컨테이너 생성
2. 빈 메타 정보(xml이나 자바 기반 설정)를 이용해서 빈객체 생성
3. 컨테이너 사용
4. 컨테이너 종료 빈객체 제거

|  |
| --- |
| String configLocation = "classpath:applicationContext.xml";  AbstractApplicationContext ctx =  **new** GenericXmlApplicationContext(configLocation);  Project project = ctx.getBean("sampleProject", Project.**class**);  project.build();  ctx.close(); |
| 부모 컨테이너등을 지정할수 있다  AnnotationConfigApplicationContext ctx =  **new** AnnotationConfigApplicationContext("net.madvirus.spring4.chap02.conf");  **ctx.setParent(...);**  많이 사용할일은 없지만 ..  부모컨테이너를 지정하기 전에 자식컨테이너를 초기화해야한다. |

Chapter3 127p

빈 객체의 라이프 사이클

1. 스프링이 제공하는 특정 인터페이스를 상속받아 빈을 구현한다
2. 스프링 설정에서 특정 메서드를 호출하라고 지정한다.

|  |  |
| --- | --- |
| 사용자 삽입 이미지 | IMG_20160225_233141.jpg |
| 빈의 초기화와 소멸방법은 각각 세가지가 존재하며  각방식이 쌍을 이루어 함께 사용되곤한다 즉 @PostConstruct어노테이션을 사용해소 초기화 메서드를 지정했다면 @PreDestroy 어노테이션 사용해서 소멸 메서드를 지정하고 커스텀 init메서드를 사용했ㄷ면 커스텀 destory메서드를 사용하는 식이다. | |

o.s.beans.factory.InitializiongBean

o.s.beansfactory.DisposableBean

|  |
| --- |
| **public** **class** ConnPool1 **implements** InitializingBean, DisposableBean {  @Override  **public** **void** afterPropertiesSet() **throws** Exception {  System.***out***.println("ConnPool1.afterPropertiesSet()");  }  @Override  **public** **void** destroy() **throws** Exception {  System.***out***.println("ConnPool1.destroy()");  }  }  **public** **class** ConnPool2 {  @PostConstruct  **public** **void** initPool() {  System.***out***.println("ConnPool2.initPool()");  }  @PreDestroy  **public** **void** destroyPool() {  System.***out***.println("ConnPool2.destroyPool()");  }  } |

커스텀 init 메서드와 destory메서드

|  |
| --- |
| **public** **class** ConnPool3 {  **public** **void** init() {  System.***out***.println("ConnPool3.init()");  }  **public** **void** destroy() {  System.***out***.println("ConnPool3.destroy()");  }  } |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  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"*>  **<context:annotation-config />**  <bean class=*"net.madvirus.spring4.chap03.MyExtension"* />    <bean class=*"net.madvirus.spring4.chap03.ConnPool1"* />  <bean class=*"net.madvirus.spring4.chap03.ConnPool2"* />  <bean class=*"net.madvirus.spring4.chap03.ConnPool3"*  init-method=*"init"* destroy-method=*"destroy"* />  <bean id=*"myBean"* class=*"net.madvirus.spring4.chap03.MyBean"*  **init-method=*"customInit"* destroy-method=*"customDestroy"***>  <property name=*"property1"* value=*"값1"* />  </bean>  </beans> |

ApplicationContextAware인터페이스와 BeanNameAware인터페이스

[org](eclipse-javadoc:%E2%98%82=spring4-chap03/lib%5C/spring-context-4.0.4.RELEASE.jar%3Corg).[springframework](eclipse-javadoc:%E2%98%82=spring4-chap03/lib%5C/spring-context-4.0.4.RELEASE.jar%3Corg.springframework).[context](eclipse-javadoc:%E2%98%82=spring4-chap03/lib%5C/spring-context-4.0.4.RELEASE.jar%3Corg.springframework.context).ApplicationContextAware

|  |
| --- |
| **public** **class** WorkScheduler **implements** ApplicationContextAware {  **private** WorkRunner workRunner;  **private** ApplicationContext ctx;  @Override  **public** **void** setApplicationContext(ApplicationContext applicationContext) **throws** BeansException {  **this**.ctx = applicationContext;  }  **public** **void** makeAndRunWork() {  **for** (**long** order = 1; order <= 10; order++) {  Work work = ctx.getBean("workProto", Work.**class**);  work.setOrder(order);  workRunner.execute(work);  }  }  **public** **void** setWorkRunner(WorkRunner workRunner) {  **this**.workRunner = workRunner;  }  } |
| **ApplicationContext를 전달받는다 따라서 전달받은 컨테이너를 필드에 보관후 이를 이용해서 다른 빈 객체를 구하거나 컨테이너가 제공하는 이벤트 밸생, 메시지 구하기등을 사용할수 있다.** |

[org](eclipse-javadoc:%E2%98%82=spring4-chap03/lib%5C/spring-beans-4.0.4.RELEASE.jar%3Corg).[springframework](eclipse-javadoc:%E2%98%82=spring4-chap03/lib%5C/spring-beans-4.0.4.RELEASE.jar%3Corg.springframework).[beans](eclipse-javadoc:%E2%98%82=spring4-chap03/lib%5C/spring-beans-4.0.4.RELEASE.jar%3Corg.springframework.beans).[factory](eclipse-javadoc:%E2%98%82=spring4-chap03/lib%5C/spring-beans-4.0.4.RELEASE.jar%3Corg.springframework.beans.factory).BeanNameAware

|  |
| --- |
| **public** **class** WorkRunner **implements** BeanNameAware {  **private** String beanId;  @Override  **public** **void** setBeanName(String name) {  **this**.beanId = name;  }  **public** **void** execute(Work work) {  System.***out***.printf("WorkRunner[%s] execute Work[%d]\n", beanId, work.getOrder());  work.run();  }  } |
| 빈이름을 전달받는다 따라서 로그 메시지에 빈 이름을 함께 기록해야할 때처럼 빈의 이름이 필요한 경우 BeannameAware인터페이스를 사용하면된다. |

빈객체 범위 (scope) p133

1. 싱글톤 범위
   1. <bean class=*"net.madvirus.spring4.chap03.ConnPool1"* /> ctx.getBean으로 매번 호출을 하여도 매번 같은 Instance를 반환한다 즉 처음한번 생성된 빈객체가 호출된다
   2. 명시적으로 적어줄수도있다 scope=”singleton” @Scope(“singleton”)
2. 프로토타입 범위 (매번 새로이 생성) @Scope(“prototype”)

<bean id=*"workProto"* class=*"net.madvirus.spring4.chap03.Work"*

scope=*"prototype"*>

<property name=*"timeout"* value=*"2000"* />

<property name=*"type"* value=*"#{T(net.madvirus.spring4.chap03.Work$WorkType).SINGLE}"* />

</bean>

Chapter04 p138

Environment, 프로퍼티, 프로필, 메시지

Environment는 두가지 기능을 제공한다

1. 프로퍼티 통합관리
2. 프로필을 이용해서 선택적으로 설정 정보를 사용할 수 있는 방법 제공

시스템 환경 변수 JVM 시스템 프로퍼티 파일등의 프로퍼티를 PropertySource라는 것으로 통합관리한다

라서 설정 파일이나 클래스 수정없이 시스템 프로퍼티나 프로퍼티 파일등을 이용해서 설정 정보의 일부를 변경할수 있다.

서로 다른 환경을 위한 설정 정보를 활성 비활성 또는 설정 정보를 편하게 관리가능하다

|  |
| --- |
| **public** **class** MainByPropSource {  **public** **static** **void** main(String[] args) **throws** IOException {  AnnotationConfigApplicationContext ctx = **new** AnnotationConfigApplicationContext(ConfigByPropSource.**class**);  ConfigurableEnvironment env = ctx.getEnvironment();  String javaVersion = env.getProperty("java.version");  String dbUser = env.getProperty("db.user");  System.***out***.printf("java version is %s\n", javaVersion);  System.***out***.printf("dbUser is %s\n", dbUser);  ctx.close();  }  } |
| **IMG_20160226_010849.jpg** |

Chapter06 AOP p200

|  |  |
| --- | --- |
| Joinpoint | Advice를 적용 가능한 지점을 의미한다 메서드 호출 필드 값 변경등이 Joinpoint에 해당된다 |
| Pointcut | Joinpoint의 부분 집합으로서 실제로 Advice가 적용되는 Joinpoint를 나타낸다 스프링에서는 정규 표현식이나 AspectJ의 문법을 이용하여 Pointcut을 정의 할수 있다 |
| Advice | 언제 공통 관심 기능을 핵심 로직에 적용할 지를 정의하고 있다 예를 들어 메서드를 호출하기전 언제 에 트랙잭션 시작 공통 기능을 적용한다는 것을 정의하고 있다 |
| Weaving | Advice를 핵심로직 코드에 적용하는 것을 weaving이라고 한다 즉 공통 코드를 핵심 로직 코드에 삽입하는 것이 weaving이다 |
| Aspect | 여러 객체에 공통으로 적용되느느 기능을 Aspect라고 한다 트랜잭션이나 보안등이 Aspect의 좋은 예이다 |

구현 가능한 Advice종류

|  |  |
| --- | --- |
| Before | 대상 객체의 메서드 호출전에 공통기능 실행 |
| After Returning | 대상 객체의 메서드가 익센션 없이 실행된후 공통기능을 실행한다 |
| After Throwing | 대상객체의 메서드를 실행하는 도중 익셉션이 발생한 경유 공통 기능 실행 |
| After | 대상 객체의 메서드를 실행하는 도중 익셉션 발생유무와 상관잆이 실행 |
| **Around** | **대상 객체의 메서드 실행전 후 또는 익셉션 발생 시점에 공통 기능을 실행하는데 사용한다** |

|  |
| --- |
| **package** net.madvirus.spring4.chap06.aop;  **import** org.aspectj.lang.ProceedingJoinPoint;  **public** **class** Profiler {  **public** Object trace(ProceedingJoinPoint joinPoint) **throws** Throwable {  String signatureString = joinPoint.getSignature().toShortString();  System.***out***.println(signatureString + " 시작");  **long** start = System.*currentTimeMillis*();  **try** {  Object result = joinPoint.proceed();  **return** result;  } **finally** {  **long** finish = System.*currentTimeMillis*();  System.***out***.println(signatureString + " 종료");  System.***out***.println(signatureString + " 실행 시간 : " +  (finish - start) + "ms");  }  }  } |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:aop=*"http://www.springframework.org/schema/aop"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans.xsd*  *http://www.springframework.org/schema/aop*  *http://www.springframework.org/schema/aop/spring-aop.xsd"*>  <!-- 공통 기능을 제공할 클래스를 빈으로 등록 -->  <bean id=*"profiler"* class=*"net.madvirus.spring4.chap06.aop.Profiler"* />  <bean id=*"memberUpdateTraceAdvice"*  class=*"net.madvirus.spring4.chap06.aop.UpdateMemberInfoTraceAdvice"* />  <!-- Aspect 설정: Advice를 어떤 Pointcut에 적용할 지 설정 -->  **<aop:config>**  **<aop:aspect id=*"traceAspect"* ref=*"profiler"*>**  **<aop:pointcut id=*"publicMethod"***  **expression=*"execution(public \* net.madvirus.spring4.chap06..\*(..))"* />**  **<aop:around pointcut-ref=*"publicMethod"* method=*"trace"* />**  **</aop:aspect>**  **<aop:aspect id=*"memberUpdateTraceAspect"* ref=*"memberUpdateTraceAdvice"*>**  **<aop:after-returning pointcut=*"args(memberId,info)"***  **method=*"traceReturn"* />**  **</aop:aspect>**  **</aop:config>**  <bean id=*"writeArticleService"*  class=*"net.madvirus.spring4.chap06.board.WriteArticleServiceImpl"*>  <constructor-arg>  <ref bean=*"articleDao"* />  </constructor-arg>  </bean>  <bean id=*"articleDao"* class=*"net.madvirus.spring4.chap06.board.MemoryArticleDao"* />  <bean id=*"memberService"* class=*"net.madvirus.spring4.chap06.member.MemberServiceImpl"* />  <bean id=*"cache"* class=*"net.madvirus.spring4.chap06.aop.ArticleCacheAdvice"* />  <aop:config>  <aop:aspect id=*"cacheAspect"* ref=*"cache"*>  <aop:around method=*"cache"*  pointcut=*"execution(public \* \*..ReadArticleService.\*(..))"* />  </aop:aspect>  </aop:config>  <bean id=*"readArticleService"*  class=*"net.madvirus.spring4.chap06.board.ReadArticleServiceImpl"*>  <property name=*"articleDao"* ref=*"articleDao"* />  </bean>  </beans> |

|  |
| --- |
| **public** **class** MainQuickStart {  **public** **static** **void** main(String[] args) {  GenericXmlApplicationContext ctx =  **new** GenericXmlApplicationContext("classpath:acQuickStart.xml");  WriteArticleService writeArticleService =  ctx.getBean("writeArticleService", WriteArticleService.**class**);  writeArticleService.write(**new** NewArticleRequest("writer", "title", "content"));  ReadArticleService readArticleService = ctx.getBean(ReadArticleService.**class**);  readArticleService.read(1);  readArticleService.read(1);  MemberService memberService = ctx.getBean(MemberService.**class**);  MemberRegRequest memberRegReq =  **new** MemberRegRequest("id", "name", "pw");  memberService.regist(memberRegReq);  UpdateInfo updateInfo = **new** UpdateInfo();  updateInfo.setNewName("새이름");  memberService.update("madvirus", updateInfo);  ctx.close();  }  } |

….더내용있음 아몰랑

Chapter07 스프링 웹개발 p258

서블릿 등록(org.springframework.web.servlet.DispatcherServlet)

|  |
| --- |
| **web.xml**  <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xmlns=*"http://java.sun.com/xml/ns/javaee"*  xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee*  *http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"*  id=*"spring4-chap07"* version=*"3.0"*>  <display-name>spring4-chap07</display-name>  **<servlet>**  **<servlet-name>dispatcher</servlet-name>**  **<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>**  **<init-param>**  **<param-name>contextConfigLocation</param-name>**  **<param-value>**  **/WEB-INF/mvc-quick-start.xml**  **</param-value>**  **</init-param>**  **<load-on-startup>1</load-on-startup>**  **</servlet>**  **<servlet-mapping>**  **<servlet-name>dispatcher</servlet-name>**  **<url-pattern>\*.do</url-pattern>**  **</servlet-mapping>**  http://javajigi.net/download/attachments/1160/RequestLifeCycleInSpringMVC.jpg  <filter>  <filter-name>encodingFilter</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>encodingFilter</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>  </web-app> |
| **mvc-quick-start.xml**  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:mvc=*"http://www.springframework.org/schema/mvc"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans.xsd*  *http://www.springframework.org/schema/mvc*  *http://www.springframework.org/schema/mvc/spring-mvc.xsd"*>  **<mvc:annotation-driven />**  <bean id=*"viewResolver"*  class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>  <property name=*"prefix"* value=*"/WEB-INF/view/"* />  <property name=*"suffix"* value=*".jsp"* />  </bean>  <bean class=*"net.madvirus.spring4.chap07.quickstart.HelloController"* />  </beans> |
| **HelloController.java**  **@Controller**  **public** **class** HelloController {  **@RequestMapping("/hello.do")**  **public** String hello(Model model) {  model.addAttribute("greeting", "안녕하세요");  **return** "hello";  }    **@RequestMapping("/hello-raw.do")**  **public** **void** hello(HttpServletResponse response) **throws** IOException {  response.setContentType("text/plain");  response.setCharacterEncoding("utf-8");  PrintWriter writer = response.getWriter();  writer.write("안녕하세요");  writer.flush();  }  } |

스프링 MVC의 주요 구성요소

|  |  |
| --- | --- |
| DispatcherServlet | 클라이언트의 요청을 전달받는다 컨트롤러에게 클라이언트의 요청을 전달하고 컨트롤러가 리턴한 결과값을 view에 전달하여 알맞은 응답을 생성하도록한다 |
| HandlerMapping | 클라이언트의 요청 URL을 어떤 컨트롤러가 처리할지 결정한다 |
| HandlerAdapter | DispatcherServlet의 처리 요청을 변환해서 컨트롤러에게 전달하고 컨트롤러의 응답 결과를 DispatcherServlet이 요구 하는 형식으로 변환한다 웹브라우저 캐시 등의 설정 담당한다 |
| 컨트롤러(Controller) | 클아이언트의 요청을 처리한뒤 결과를 리턴한다 응답 결과에서 보여줄 데이터를 모델에 담아 전달한다 |
| ModelAndView | 컨트롤러가 처리한 결과 정보 및 뷰 선택에 필요한 정보를 담는다 |
| ViewResolver | 컨트롤러의 처리 결과를 보여줄 뷰를 결정한다 |
| 뷰(View) | 컨트롤러의 처리 결과 화면을 생성한다 j네나 velocity템플릿 파일 등을 이용해서 클라이언트에 응답 결과를 전송한다 |
| ... | ... |

@Configuration클래스를 이용해서 설정 정보를 작성했다면

|  |
| --- |
| <servlet>  <servlet-name>dispatcher2</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextClass</param-name>  <param-value>  **org.springframework.web.context.support.AnnotationConfigWebApplicationContext**  </param-value>  </init-param>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>  **net.madvirus.spring4.chap07.quickstart.MvcQuickStartConfig**  </param-value>  </init-param>  <load-on-startup>1</load-on-startup>  </servlet> |
| **MvcQuickStartConfig**  **@Configuration**  **@EnableWebMvc**  **public** **class** MvcQuickStartConfig {  @Bean  **public** InternalResourceViewResolver viewResolver() {  InternalResourceViewResolver viewResolver = **new** InternalResourceViewResolver();  viewResolver.setPrefix("/WEB-INF/view/");  viewResolver.setSuffix(".jsp");  **return** viewResolver;  }  @Bean  **public** HelloController2 helloController() {  **return** **new** HelloController2();  }  } |

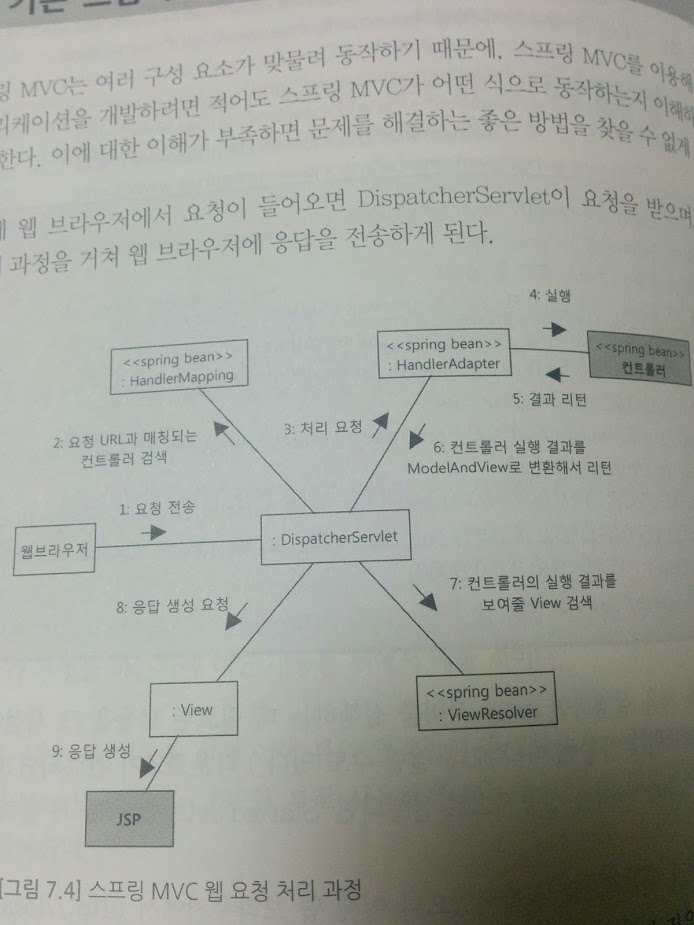
캐릭터 인코딩 필터 설정

|  |
| --- |
| <filter>  <filter-name>encodingFilter</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>encodingFilter</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping> |

MVC설정 기초p272

1. HandlerMapping 구현객체
2. HandlerAdapter구현객체
3. ViewResolver구현객체

HandlerMapping, HandlerAdapter는 다음과같이 <mvc:annotation-driven /> 태그를 이용하면 설정이 끝난다 따라서 ViewResolver만 추가로 설정해주면 끝난다. 오호~



<mvc:annotation-driven /> 등록하면 자동으로 아래2개 등록해준다

|  |  |
| --- | --- |
| RequestMappingHandlerMapping | RequestMappingHandlerAdapter |
|  |  |
| 위두 클래스는@Controller어노테이션이 적용된 클래스를 컨트롤러로 사용할수 있도록 한다 JSON이나 XML등 요청 응답 처리를 위해 필요한 변환 모듈이나 데이터 바인등 처리를 위한 ConversionService등을 빈으로 등록해준다  @Configuration 자바 설정을 사용한다면 @EnableWebMvc어노테이션을 사용하면 동일한 효과를 얻을수 있다. | |

컨트롤로 경로 매핑과 디폴트 서블릿 설정 p275

디폴트 서블릿

|  |
| --- |
| <servlet>  <servlet-name>dispatcher3</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>  /WEB-INF/sample.xml  </param-value>  </init-param>  <load-on-startup>1</load-on-startup>  </servlet>  <servlet-mapping>  <servlet-name>dispatcher3</servlet-name>  <url-pattern>/</url-pattern>  </servlet-mapping> |
| <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:mvc=*"http://www.springframework.org/schema/mvc"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans.xsd*  *http://www.springframework.org/schema/mvc*  *http://www.springframework.org/schema/mvc/spring-mvc.xsd"*>  <mvc:annotation-driven />  <mvc:default-servlet-handler />... |
| urlpattern를 /로 설정하게되면 jsp요청을 제외한 나머지요청은 DispatchServlet이 받아들이게된다.  디폴트 서블릿은 예를들어 /bootstrap/bootstrap.min.css경로의 경우매핑된 컨트롤러가 없을경우 404응답코드를 웹브라우저에 전송하게된다 이때 404 대신 컨테이너의 디폴트 서블릿을 이용해서 요청을 처리하도록 만들어주는것이 바로 <mvc:annotation-driven /> 설정이다   1. 요청 URL에 매핑되는 컨트롤러를 검색한다    1. 존재할경우, 컨트롤러를 이용해서 클라이언트 요청처리 2. 디폴트 서블릿 핸들러가 등록되어 있지 않으면    1. 404응답 에러 3. 디폴트 서블릿 핸들러가 등록되어 있으면 디폴트 서블릿 핸들러에 요청을 전달한다    1. 디폴트 서블릿 핸들러는 WAS의 디폴트 서블릿에 요청을 전달한다   jsp요청에 처리하는것이 바로 디폴트 서블릿이다. |

컨트롤구현

1. @Controller어노테이션 클래스 적용
2. @RequestMapping 어노테이션 이용 처리 요청 경로 지정
3. 웹브라우저 요청 처리할 메서드 구현하고 뷰 이름 리턴.

|  |
| --- |
| @Controller  **public** **class** EventController {  @RequestMapping("/list")  **public** String list(SearchOption option, Model model) {  List<Event> eventList = eventService.getOpenedEventList(option);  model.addAttribute("eventList", eventList);  model.addAttribute("eventTypes", EventType.*values*());  **return** "event/list";  } |
| Model 을 추가하여 거기에 데이터를 넣을수 있다.  addAttribute, addAllAttributes <Map..>  containsAttribute(name) 포함하는지 |
| <!DOCTYPE html>  <html>  <head>  <title>이벤트 목록</title>  </head>  <body>  <form>  <label><input type=*"checkbox"* name=*"allType"* value=*"true"*>전체</label>  <c:forEach var=*"eventType"* items=*"*${eventTypes}*"*>  <label><input type=*"checkbox"* name=*"types"* value=*"*${eventType}*"*>${eventType}</label>  </c:forEach>  시작일:<input type=*"text"* name=*"from"* />~<input type=*"text"* name=*"to"* />  <input type=*"submit"* value=*"검색"* />  </form>  현재 오픈된 이벤트:  <ul>  <c:forEach var=*"event"* items=*"*${eventList}*"*>  <li><a href=*"/spring4-chap07/event/detail?id=*${event.id}*"*>${event.name} [${event.type}]</a></li>  </c:forEach>  </ul>  추천 이벤트:  <ul>  <c:forEach var=*"event"* items=*"*${recEventList}*"*>  <li><a href=*"/spring4-chap07/event/detail?id=*${event.id}*"*>${event.name}</a></li>  </c:forEach>  </ul>  </body>  </html> |

ModelAndView로 처리

|  |
| --- |
| @RequestMapping("/list2")  **public** ModelAndView list2(SearchOption option) {  List<Event> eventList = eventService.getOpenedEventList(option);  ModelAndView modelView = **new** ModelAndView();  **modelView.setViewName("event/list");**  modelView.addObject("eventList", eventList);  modelView.addObject("eventTypes", EventType.*values*());  **return** modelView;  } |

@RequestMapping을 이용한 매핑

|  |
| --- |
| **@RequestMapping(value = "/newevent/step2", method = RequestMethod.*POST***)  **public** String step2(@ModelAttribute("eventForm") EventForm formData, BindingResult result) {  **new** EventFormStep1Validator().validate(formData, result);  **if** (result.hasErrors())  **return** ***EVENT\_CREATION\_STEP1***;  **return** ***EVENT\_CREATION\_STEP2***;  }  또는  @RequestMapping(value = {"/newevent/done","/new/v"}, method = RequestMethod.***POST***) |
| POST , GET을 지정하여 HTTP전송방식에 따라 처리가능  여러 URL맵핑을 지정가능 |

클래스 와 메서드에 @RequestMapping적용하기

|  |
| --- |
| @Controller  @RequestMapping("/event")  **public** **class** EventController {  **private** **static** **final** String ***REDIRECT\_EVENT\_LIST*** = "redirect:/event/list";  **private** EventService eventService;  @RequestMapping("/list")  **public** String list(SearchOption option, Model model) {  List<Event> eventList = eventService.getOpenedEventList(option);  model.addAttribute("eventList", eventList);  model.addAttribute("eventTypes", EventType.*values*());  **return** "event/list";  } |
| 클래스에 /event를 적용시켰기때문에 안쪽 메서드에 적용된 /list 는  /event/list에 맵핑이된다 |

@PathVariable를 이용한 경로변수

|  |
| --- |
| @Controller  **public** **class** MemberController {  **private** MemberService memberService;  @RequestMapping("/members")  **public** String members(Model model) {  List<MemberInfo> members = memberService.getMembers();  model.addAttribute("members", members);  **return** "member/members";  }  **@RequestMapping("/members/{memberId}")**  **public String memberDetail(@PathVariable String memberId, Model model) {**  **MemberInfo mi = memberService.getMemberInfo(memberId);**  **if (mi == null) {**  **return "member/memberNotFound";**  **}**  **model.addAttribute("member", mi);**  **return "member/memberDetail";**  **}**  @RequestMapping("/members/{memberId}/orders")  **public** String memberOrders(@PathVariable("memberId") String memberId, Model model) {  MemberInfo mi = memberService.getMemberInfo(memberId);  **if** (mi == **null**) {  **return** "member/memberNotFound";  }  model.addAttribute("member", mi);  model.addAttribute("orders", getOrdersOfMember(memberId));  **return** "member/memberOrders";  }  **private** List<OrderInfo> getOrdersOfMember(String memberId) {  **return** Arrays.*asList*(  **new** OrderInfo(1L, 10000, memberId),  **new** OrderInfo(2L, 15000, memberId)  );  }  **@RequestMapping("/members/{memberId}/orders/{orderId}")**  **public String memberOrderDetail(@PathVariable("memberId") String memberId,**  **@PathVariable("orderId") Long orderId,**  **Model model) {**  **model.addAttribute("member", memberService.getMemberInfo(memberId));**  **model.addAttribute("order", new OrderInfo(orderId, orderId.intValue() \* 5000 + 5000, memberId));**  **return "member/memberOrderDetail";**  **}**  **public** **void** setMemberService(MemberService memberService) {  **this**.memberService = memberService;  }  } |
| @Pathvariable로 지정한곳에는 RequestMapping에서 범위를 잡은 값을 파라미터값으로 넘어온다.  정규표현식 또는 Ant패턴가능 |

Content-Type요청 해더가 application/json 인경우만 처리하고 싶다면 다음같은

@RequestMapping(value=”/members”, method=RequestMethod.POST, consumes=”application/json”

HTTP 요청 파라미터와 폼 데이터 처리

HttpServletRequest request 처리

|  |
| --- |
| @RequestMapping("/detail")  **public** String detail(HttpServletRequest request, Model model) **throws** IOException {  String id = request.getParameter("id");  **if** (id == **null**)  **return** ***REDIRECT\_EVENT\_LIST***;  Long eventId = **null**;  **try** {  eventId = Long.*parseLong*(id);  } **catch** (NumberFormatException e) {  **return** ***REDIRECT\_EVENT\_LIST***;  }  Event event = getEvent(eventId);  **if** (event == **null**)  **return** ***REDIRECT\_EVENT\_LIST***;  model.addAttribute("event", event);  **return** "event/detail";  } |

@RequestParam 어노테이션을 이용한 요청 파라미터 구학

|  |
| --- |
| @RequestMapping("/detail2")  **public** String detail2(@RequestParam("id") **long** eventId, Model model) {  Event event = getEvent(eventId);  **if** (event == **null**)  **return** ***REDIRECT\_EVENT\_LIST***;  model.addAttribute("event", event);  **return** "event/detail";  }  } |
| 파라미터 값을 이미 가져와서 파리미터값으로 넣음 만약 id값이 없으면 null이면 위 메서드는 타지 않는다.  하지만 값이 없어도 호출되도록 하고싶다면  @RequestMapping("/search")  **public** String search(@RequestParam(value = "q", required=**false**) String query,  Model model) {  System.***out***.println("검색어: " + query);  **return** "search/result";  }  기본값을 처리하고싶다면  @RequestMapping("/search")  **public** String search(@RequestParam(value = "q", defaultValue="") String query,  Model model) {  System.***out***.println("검색어: " + query);  **return** "search/result";  } |

커맨드 객체를 이용한 폼 전송 처리하기

**public** **class** MemberRegistRequest {

**private** String email;

**private** String name;

**private** String password;

**private** String confirmPassword;

**private** **boolean** allowNoti;

**private** Address address;

**private** Date birthday;

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getPassword() {

...

}

}

|  |
| --- |
| @RequestMapping(method = RequestMethod.***POST***)  **public** String regist(  @ModelAttribute("memberInfo") **MemberRegistRequest memRegReq**,  BindingResult bindingResult) {  **new** MemberRegistValidator().validate(memRegReq, bindingResult);  **if** (bindingResult.hasErrors()) {  **return** ***MEMBER\_REGISTRATION\_FORM***;  }  memberService.registNewMember(memRegReq);  **return** "member/registered";  } |
| <form method=*"post"*>  <label for=*"email"*>이메일</label>:  <input type=*"text"* name=*"email"* id=*"email"* value=*"*${memberInfo.email}*"*/>  <label for=*"name"*>이름</label>:  <input type=*"text"* name=*"name"* id=*"name"* value=*"*${memberInfo.name}*"* />  <label for=*"password"*>암호</label>:  <input type=*"password"* name=*"password"* id=*"password"* value=*"*${memberInfo.password}*"*/>  <label for=*"password"*>확인</label>:  <input type=*"password"* name=*"confirmPassword"* id=*"confirmPassword"* value=*"*${memberInfo.confirmPassword}*"*/>  <label>주소</label>:  주소1  <input type=*"text"* name=*"address.address1"* value=*"*${memberInfo.address.address1}*"* />  주소2  <input type=*"text"* name=*"address.address2"* value=*"*${memberInfo.address.address2}*"* />  우편번호  <input type=*"text"* name=*"address.zipcode"* value=*"*${memberInfo.address.zipcode}*"* />  <label>  <input type=*"checkbox"* name=*"allowNoti"* value=*"true"* ${memberInfo.allowNoti ? 'checked' : '' }/>  이메일을 수신합니다.  </label>  <br/>  <label for=*"birthday"*>생일</label>: 형식: YYYYMMDD, 예: 20140101  <input type=*"text"* name=*"birthday"* id=*"birthday"* value=*'*<fmt:formatDate value="${memberInfo.birthday}" pattern=*"yyyyMMdd"* />*'*/>  <input type=*"submit"* value=*"가입"* />  </form> |
| ModelAttribute("memberInfo") 통해 view에서 사용할 이름을 지정 지정하지 않으면 클래스 명그대로 사용한다 맨앞글자는 소문자로..  name의 배열로도 받을수도있다. 배열대신 List로 처리할수도있다. |

중첩 프로퍼티 지원

|  |  |
| --- | --- |
| **public** **class** MemberInfo {  **...**  ...  **public** **void** setAddress(Address address) {  **this**.address = address;  }  } | **public** **class** Address {  @NotEmpty  **private** String address1;  @NotEmpty  **private** String address2;  **private** String zipcode;  **public** String getAddress1() {  **return** address1;  }  **public** **void** setAddress1(String address1) {  **this**.address1 = address1;  }  **public** String getAddress2() {  **return** address2;  }  **public** **void** setAddress2(String address2) {  **this**.address2 = address2;  }  **public** String getZipcode() {  **return** zipcode;  }  **public** **void** setZipcode(String zipcode) {  **this**.zipcode = zipcode;  }  } |
| <input type=*"password"* name=*"confirmPassword"* id=*"confirmPassword"* value=*"*${memberInfo.confirmPassword}*"*/>  주소1  <input type=*"text"* name=*"address.address1"* value=*"*${memberInfo.**address**.address1}*"* />  주소2  <input type=*"text"* name=*"address.address2"* value=*"*${memberInfo.**address**.address2}*"* />  우편번호  <input type=*"text"* name=*"address.zipcode"* value=*"*${memberInfo.**address**.zipcode}*"* /> | |

GET/POST에서 동일 타입 커맨드 객체 사용하기,

|  |
| --- |
| @Controller  @RequestMapping("/member/modify")  **public** **class** MemberModificationController {  **private** **static** **final** String ***MEMBER\_MODIFICATION\_FORM*** = "member/modificationForm";  **private** **static** **final** String ***MEMBER\_NOT\_FOUND\_VIEW*** = "member/memberNotFound";  **private** MemberService memberService;  @RequestMapping(method = RequestMethod.***GET***)  **public** String form(@ModelAttribute("modReq") MemberModRequest modReq,  @RequestParam("mid") String memberId) {  MemberInfo mi = memberService.getMemberInfo(memberId);  **if** (mi == **null**)  **return** ***MEMBER\_NOT\_FOUND\_VIEW***;  modReq.setId(mi.getId());  modReq.setName(mi.getName());  modReq.setEmail(mi.getEmail());  modReq.setAllowNoti(mi.isAllowNoti());  **return** ***MEMBER\_MODIFICATION\_FORM***;  }  @RequestMapping(method = RequestMethod.***POST***)  **public** String modify(@Valid @ModelAttribute("modReq") MemberModRequest modReq, Errors errors) {  **if** (errors.hasErrors()) {  **return** ***MEMBER\_MODIFICATION\_FORM***;  }  **try** {  memberService.modifyMemberInfo(modReq);  **return** "member/modified";  } **catch** (NotMatchPasswordException ex) {  errors.rejectValue("currentPassword", "invalidPassword");  **return** ***MEMBER\_MODIFICATION\_FORM***;  } **catch** (MemberNotFoundException ex) {  **return** ***MEMBER\_NOT\_FOUND\_VIEW***;  }  }  **public** **void** setMemberService(MemberService memberService) {  **this**.memberService = memberService;  }  } |

@ModelAttribute 어노테이션 이용한 공통 모델 처리

|  |
| --- |
| @Controller  @RequestMapping("/event")  **public** **class** EventController {  **private** **static** **final** String ***REDIRECT\_EVENT\_LIST*** = "redirect:/event/list";  **private** EventService eventService;  **public** EventController() {  eventService = **new** EventService();  }  @ModelAttribute("recEventList")  **public** List<Event> recommend() {  **return** eventService.getRecommendedEventService();  }  @RequestMapping("/list3")  **public** String list3(@ModelAttribute("recEventList") List<Event> recEventList, Model model) {  **return** "event/list";  } |

@CookieValue와 @RequestHeader를 이용한 쿠키 및 요청 헤더

|  |
| --- |
| @Controller  **public** **class** SimpleHeaderController {  @RequestMapping("/header/simple")  **public** String simple(  @RequestHeader(value = "Accept", defaultValue = "text/html") String acceptType,  @CookieValue(value = "auth", required = **false**) Integer authValue,  Model model) {  model.addAttribute("acceptType", acceptType);  **if** (authValue != **null**)  model.addAttribute("auth", authValue);  **return** "header/simpleValue";  }  @RequestMapping("/header/createauth")  **public** String createAuth(HttpServletResponse response,  Model model) {  Random random = **new** Random();  String authValue = Integer.*toString*(random.nextInt());  response.addCookie(**new** Cookie("auth", authValue));  **return** "redirect:simple";  // return "redirect:http://localhost:8080/spring4-chap07/index.jsp";  }  }  <!DOCTYPE html>  <html>  <head>  <title>헤더 정보</title>  </head>  <body>  Accept: ${acceptType}  <br/>  Auth Cookie: ${auth}  </body>  </html> |
| Accept인 요청 헤더의 값과 이름 auth인 쿠키를 각각 acceptType파라미터와 cookie파라미터에 전달한다 |

리다이렉트처리

|  |
| --- |
| **return** "redirect:simple"; //접두어 redirect처리 웹어플리케이션 내에서의 경로  /로시작하지 않으면 위쪽 @reqeustMapping 값에서부터 상대경로가된다  @RequestMapping(value=”/file/{fileId}”)  return “redirect:/file/{fileId} 처럼 변수로도 사용가능하다 |

커맨드 객체 값 검증과 에러 메시지p310

|  |
| --- |
| **public** **class** MemberRegistValidator **implements** Validator {    @Override  **public** **boolean** supports(Class<?> clazz) {  **return** MemberRegistRequest.**class**.isAssignableFrom(clazz);  }  @Override  **public** **void** validate(Object target, Errors errors) {  MemberRegistRequest regReq = (MemberRegistRequest) target;  **if** (regReq.getEmail() == **null** || regReq.getEmail().trim().isEmpty())  **errors.rejectValue("email", "required");**  ValidationUtils.*rejectIfEmptyOrWhitespace*(errors, "name", "required");  ValidationUtils.*rejectIfEmptyOrWhitespace*(errors, "password", "required");  ValidationUtils.*rejectIfEmptyOrWhitespace*(errors, "confirmPassword", "required");  **if** (regReq.hasPassword()) {  **if** (regReq.getPassword().length() < 5)  **errors.rejectValue("password", "shortPassword");**  **else** **if** (!regReq.isSamePasswordConfirmPassword())  **errors.rejectValue("confirmPassword", "notSame");**  }  Address address = regReq.getAddress();  **if** (address == **null**) {  **errors.rejectValue("address", "required");**  } **else** {  **errors.pushNestedPath("address");**  **try** {  **ValidationUtils.*rejectIfEmptyOrWhitespace*(errors, "address1", "required");**  ValidationUtils.*rejectIfEmptyOrWhitespace*(errors, "address2", "required");  } **finally** {  **errors.popNestedPath();**  }  }  ValidationUtils.*rejectIfEmptyOrWhitespace*(errors, "birthday", "required");  }  } |
| **ValidationUtils.*rejectIfEmptyOrWhitespace*(errors, "address1", "required"); *null이거나 0이면 required로 처리한다*** |
| @Controller  @RequestMapping("/member/regist")  **public** **class** RegistrationController {  **private** **static** **final** String ***MEMBER\_REGISTRATION\_FORM*** = "member/registrationForm";    **private** MemberService memberService;  @RequestMapping(method = RequestMethod.***GET***)  **public** String form(@ModelAttribute("memberInfo") MemberRegistRequest memRegReq) {  **return** ***MEMBER\_REGISTRATION\_FORM***;  }  **@RequestMapping(method = RequestMethod.*POST*)**  **public String regist(**  **@ModelAttribute("memberInfo") MemberRegistRequest memRegReq,**  **BindingResult bindingResult) {**  **new MemberRegistValidator().validate(memRegReq, bindingResult);**  **if (bindingResult.hasErrors()) {**  **return *MEMBER\_REGISTRATION\_FORM*;**  **}**  **memberService.registNewMember(memRegReq);**  **return "member/registered";**  **}**  **public** **void** setMemberService(MemberService memberService) {  **this**.memberService = memberService;  }  } }  **public** **void** setMemberService(MemberService memberService) {  **this**.memberService = memberService;  }  } |
| 커맨드 객체의 에러 정보를 담기 위해 사용되는 Error파라미터나 BindingResult 파라미터는 반드시 커맨드 객체 파리미터 바로 뒤에 위치해야 한다. |

에러 코드와 메시지

Validator를 이용해서 오류를 확인하면 오류 내용을 화면에 보여주고 싶을것이다.

1. 메시지를 읽어올 떄 사용할 MessageSource를 스프링 설정에 등록한다
2. MessageSource에 메시지를 가져올 떄 사용할 프로퍼티 파일을 작성한다
3. jsp와 가은 뷰 코드에 스프링이 제공하는 태그를 이용해서 에러메시지를 출력한다.

|  |
| --- |
| <bean id=*"messageSource"*  class=*"org.springframework.context.support.ResourceBundleMessageSource"*>  <property name=*"basenames"*>  <list>  <value>**message.error**</value>  </list>  </property>  <property name=*"defaultEncoding"* value=*"UTF-8"* />  </bean> |

스프링 MVC는 에러 코드로부터 생성된 메시지 코드를 사용해서 에러 메시지를 출력하게 된다

글로벌 에러 코드인 경우 다음의 우선순위에 따라 메시지 코드를 생성한다

1. 에러코드.커맨드객체이름
2. 에러코드

예를들어 login{ error.reject(“invalidIdOrPassword”);} //글로벌 에러코드

에러코드는 invalidIdOrPassword이다.

처음으로 invalidIdOrPassword.loginCommand인 메시지를 찾는다 존재하지 않으면 invalidIdOrPassword찾아 보여준다

Errors.rejectValue()를 이용해서 생성한 에러코드는 이렇다

1. 에러코드.커맨드객체이름.필드명
2. 에러코드.필드명
3. 에러코드.필드타입
4. 에러코드

필드가 List나 목록인 경우 다음 순서를 사용해서 코드 생성한다

1. 에러코드.커맨드객체이름.필드명[인덱스].중첩필드명
2. 에러코드.커맨드객체이름.필드명.중첩필드명
3. 에러코드.필드명[인덱스]중첩필드명
4. 에러코드.필드명.중첩필드명
5. 에러코드.중첩필드명
6. 에러코드.필드타입
7. 에러코드

에러메시지 출력

|  |
| --- |
| <%@ page contentType=*"text/html; charset=utf-8"* %>  **<%@ taglib prefix=*"form"* uri=*"http://www.springframework.org/tags/form"* %>**  <!DOCTYPE html>  <html>  <head><title>로그인</title></head>  <body>  **<form:form commandName=*"loginCommand"*>**  **<form:errors element=*"div"* />**  <label for=*"email"*>이메일</label>:  <input type=*"text"* name=*"email"* id=*"email"* value=*"*${loginCommand.email}*"*>  **<form:errors path=*"email"*/> <br>**  <label for=*"password"*>암호</label>:  <input type=*"password"* name=*"password"* id=*"password"*>  **<form:errors path=*"password"*/> <br>**  <br/>  <input type=*"submit"* value=*"로그인"*>  **</form:form>**  <ul>  <li>이메일/암호로 yuna@yuna.com/yuna 입력 또는 sanghwa@sanghwa.com/sanghwa 로 테스트</li>  </ul>  </body>  </html> |
| @Controller  @RequestMapping("/auth/login")  **public** **class** LoginController {  **private** **static** **final** String ***LOGIN\_FORM*** = "auth/loginForm";  **private** Authenticator authenticator;  @RequestMapping(method = RequestMethod.***GET***)  **public** String loginForm(LoginCommand loginCommand) {  **return** ***LOGIN\_FORM***;  }  @RequestMapping(method = RequestMethod.***POST***)  **public String login(LoginCommand loginCommand, Errors errors,**  **HttpServletRequest request) {**  **...**  **if (errors.hasErrors()) {**  **return *LOGIN\_FORM*;**  **}**  **try {**  **Auth auth = authenticator.authenticate(loginCommand.getEmail(), loginCommand.getPassword());**  **HttpSession session = request.getSession();**  **session.setAttribute("auth", auth);**  **return "redirect:/index.jsp";**  **} catch (AuthenticationException ex) {**  **errors.reject("invalidIdOrPassword");**  **return *LOGIN\_FORM*;**  **}**  **}**  ... |
| spring:hasBindErrors를 통해서 할수도있다  <%@ page contentType=*"text/html; charset=utf-8"* %>  <%@ taglib prefix=*"c"* uri=*"http://java.sun.com/jsp/jstl/core"* %>  <%@ taglib prefix=*"fmt"* uri=*"http://java.sun.com/jsp/jstl/fmt"* %>  <%@ taglib prefix=*"spring"* uri=*"http://www.springframework.org/tags"* %>  <%@ taglib prefix=*"form"* uri=*"http://www.springframework.org/tags/form"* %>  <!DOCTYPE html>  <html>  <head>  <title>회원 가입</title>  </head>  <body>  <spring:hasBindErrors name=*"memberInfo"* />  <form method=*"post"*>  <label for=*"email"*>이메일</label>:  <input type=*"text"* name=*"email"* id=*"email"* value=*"*${memberInfo.email}*"*/>  <form:errors path=*"memberInfo.email"*/> <br/>  <label for=*"name"*>이름</label>:  <input type=*"text"* name=*"name"* id=*"name"* value=*"*${memberInfo.name}*"* />  <form:errors path=*"memberInfo.name"*/> <br/>  <label for=*"password"*>암호</label>:  <input type=*"password"* name=*"password"* id=*"password"* value=*"*${memberInfo.password}*"*/>  <form:errors path=*"memberInfo.password"*/> <br/>  <label for=*"password"*>확인</label>:  <input type=*"password"* name=*"confirmPassword"* id=*"confirmPassword"* value=*"*${memberInfo.confirmPassword}*"*/>  <form:errors path=*"memberInfo.confirmPassword"*/> <br/>  <label>주소</label>:  주소1  <input type=*"text"* name=*"address.address1"* value=*"*${memberInfo.address.address1}*"* />  <form:errors path=*"memberInfo.address.address1"*/> <br/>  주소2  <input type=*"text"* name=*"address.address2"* value=*"*${memberInfo.address.address2}*"* />  <form:errors path=*"memberInfo.address.address2"*/> <br/>  우편번호  <input type=*"text"* name=*"address.zipcode"* value=*"*${memberInfo.address.zipcode}*"* />  <form:errors path=*"memberInfo.address.zipcode"*/> <br/>  <label>  <input type=*"checkbox"* name=*"allowNoti"* value=*"true"* ${memberInfo.allowNoti ? 'checked' : '' }/>  이메일을 수신합니다.  </label>  <br/>  <label for=*"birthday"*>생일</label>: 형식: YYYYMMDD, 예: 20140101  <input type=*"text"* name=*"birthday"* id=*"birthday"* value=*'*<fmt:formatDate value="${memberInfo.birthday}" pattern=*"yyyyMMdd"* />*'*/>  <form:errors path=*"memberInfo.birthday"*/> <br/>  <input type=*"submit"* value=*"가입"* />  </form>  </body>  </html> |

@Valid어노테이션과 @InitBinder어노테이션 이용한 검증

|  |
| --- |
| @RequestMapping(method = RequestMethod.***POST***)  **public** String login(**@Valid** LoginCommand loginCommand, Errors errors,  HttpServletRequest request) {  **if** (errors.hasErrors()) {  **return** ***LOGIN\_FORM***;  }  **try** {  Auth auth = authenticator.authenticate(loginCommand.getEmail(), loginCommand.getPassword());  HttpSession session = request.getSession();  session.setAttribute("auth", auth);  **return** "redirect:/index.jsp";  } **catch** (AuthenticationException ex) {  errors.reject("invalidIdOrPassword");  **return** ***LOGIN\_FORM***;  }  }  **@InitBinder**  **protected** **void** initBinder(**WebDataBinder** binder) {  binder.setValidator(**new** LoginCommandValidator());  } |
| 스프링 프레임워크가 직접 호출하도록 해주는것  @Valid어노테이션을 적용한 login메서드 내부에서 Validator객체의 validate 메서드를 명시적으로 호출하고 있지 않는다.  login메서드는 단지 두번쨰 파라미터로 전달받은 Errors를 이용해서 에러에대한 처리만 할뿐이다.  객체를 검증할지의 여부는 initBinder()메서드를 통해 결정된다 스프링은 @InitBinder 어노테이션이 적용된 메서드를 이용해서 폼과 커맨드 객체사이의 매핑을 처리해주는 WebDataBinder를 초기화할수 있도록 하고있다 WebDataBinder.setValidate()를 통해 커맨드 객체의 유효성 여부를 검사할때 사용할 Validator를 설정하게된다. @valid 어노테이션이 붙은 커맨드 객체를 검증한다. |

글로벌 Validator와 컨트롤 Validator

|  |
| --- |
| <mvc:annotation-driven validator=”validator”>  <bean id=”validator” class=”custom.commomVal”/> |
| 위코드는 글로벌 Validator에 등록을 이용해서 @validdjshxpdltus 붙은 커맨드 객체를 검사하게된다  supports메서드가 false를 리턴하면 그 validator는 검사히자 않는다. 오호~ 지금이해됨  글로벌을 사용하지 않고 그냥 내것을 사용하고 싶다면 @InitBinder 쪽에 setValidator를 사용하면되고  글로벌을 사용하면서도 내것을 추가하고 싶으면 addValidator를 하면된다 |

@Valid어노테이션 이용한 값 검증처리

1. 커맨드 객체에 @NotNull, @Digits등 어노테이션을 이용해서 검증 규칙을 설정한다
2. LocalValidatorFactoryBean클래스를 이용해서 JSR 303프로바이더를 스프링 Validator로 등록한다
3. 컨트롤러가 두번째에 생성한 Validator로 사용하도록 설정한다

|  |  |
| --- | --- |
| **public** **class** MemberModRequest {  @NotEmpty  **private** String id;  @NotEmpty  **private** String name;  @NotEmpty  @Email  **private** String email;  **private** **boolean** allowNoti;  @NotEmpty  **private** String currentPassword;  @Valid  **private** Address address;  **public** String getId() {  **return** id;  }  ...  } |  |
| <mvc:annotation-driven/> 태그를 시용하면 기본으로 LocalValidatorFactoryBean을 등록해준다 | |

JSR303

**Table 9-2 Built-In Bean Validation Constraints**

|  |  |  |
| --- | --- | --- |
| **Constraint** | **Description** | **Example** |
| @AssertFalse | The value of the field or property must be false. | @AssertFalse  boolean isUnsupported; |
| @AssertTrue | The value of the field or property must be true. | @AssertTrue  boolean isActive; |
| @DecimalMax | The value of the field or property must be a decimal value lower than or equal to the number in the value element. | @DecimalMax("30.00")  BigDecimal discount; |
| @DecimalMin | The value of the field or property must be a decimal value greater than or equal to the number in the value element. | @DecimalMin("5.00")  BigDecimal discount; |
| @Digits | The value of the field or property must be a number within a specified range. Theinteger element specifies the maximum integral digits for the number, and thefraction element specifies the maximum fractional digits for the number. | @Digits(integer=6, fraction=2)  BigDecimal price; |
| @Future | The value of the field or property must be a date in the future. | @Future  Date eventDate; |
| @Max | The value of the field or property must be an integer value lower than or equal to the number in the value element. | @Max(10)  int quantity; |
| @Min | The value of the field or property must be an integer value greater than or equal to the number in the value element. | @Min(5)  int quantity; |
| @NotNull | The value of the field or property must not be null. | @NotNull  String username; |
| @Null | The value of the field or property must be null. | @Null  String unusedString; |
| @Past | The value of the field or property must be a date in the past. | @Past  Date birthday; |
| @Pattern | The value of the field or property must match the regular expression defined in theregexp element. | @Pattern(regexp="\\(\\d{3}\\)\\d{3}-\\d{4}")  String phoneNumber; |
| @Size | The size of the field or property is evaluated and must match the specified boundaries. If the field or property is a String, the size of the string is evaluated. If the field or property is a Collection, the size of theCollection is evaluated. If the field or property is a Map, the size of the Map is evaluated. If the field or property is an array, the size of the array is evaluated. Use one of the optional max or min elements to specify the boundaries. | @Size(min=2, max=240)  String briefMessage; |

요청파라미터의 값 변환 처리p334

WebDataBinder/@initBinder와 PropertyEditor를 이용한 타입변환

Webdatabiner는 웹요청 파라미터로부터 커맨드 객체를 생성할 때에도 사용된다

|  |
| --- |
| @InitBinder  **protected** **void** initBinder(WebDataBinder binder) {  CustomDateEditor dateEditor = **new** CustomDateEditor(**new** SimpleDateFormat("yyyyMMdd"), **true**);  binder.registerCustomEditor(Date.**class**, dateEditor);  } |

WebDataBiner와 ConversionService

<mvc:annotation-driven>태그를 정말 많은 설정을 대신해준다.

|  |
| --- |
| <bean id=*"formattingConversionService"*  class=*"org.springframework.format.support.FormattingConversionServiceFactoryBean"*>  </bean>  <bean id=*"configurableWebBindingInitializer"*  class=*"org.springframework.web.bind.support.ConfigurableWebBindingInitializer"*>  <property name=*"conversionService"* ref=*"formattingConversionService"* />  <property name=*"validator"* ref=*"optionalValidatorFactoryBean"* />  <property name=*"messageCodesResolver"*><null/></property>  </bean>  <bean id=*"requestMappingHandlerAdapter"*  class=*"org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter"*>  **<property name=*"contentNegotiationManager"* ref=*"mvcContentNegotiationManager"* />**  <property name=*"webBindingInitializer"* ref=*"configurableWebBindingInitializer"* />  <property name=*"messageConverters"* ref=*"messageConverters"* />  </bean> |
| *ConfigurableWebBindingInitializer 는 WebDataBiner를 초기화 해주는 기능* |

@DataTimeFormat 어노테이션을 이용한 시간 변환337

DATETIMEFORMATdjshxpdltusdms 특정 형식을 갖는 요청 파라미터를 java.util.Date타입이나 LocalDate타입과같은 날짜/시간 타입으로 변환할때 사용된다

|  |
| --- |
| <label for=*"birthday"*>생일</label>: 형식: YYYYMMDD, 예: 20140101  <input type=*"text"* name=*"birthday"* id=*"birthday"* value=*'*<fmt:formatDate value="${memberInfo.birthday}" pattern=*"yyyyMMdd"* />*'*/> |
| @RequestMapping(method = RequestMethod.***POST***)  **public** String regist(  @ModelAttribute("memberInfo") MemberRegistRequest memRegReq,  BindingResult bindingResult) {  **new** MemberRegistValidator().validate(memRegReq, bindingResult);  **if** (bindingResult.hasErrors()) {  **return** ***MEMBER\_REGISTRATION\_FORM***;  }  memberService.registNewMember(memRegReq);  **return** "member/registered";  } |
| **public** **class** MemberRegistRequest {  **private** String email;  **private** String name;  **private** String password;  **private** String confirmPassword;  **private** **boolean** allowNoti;  **private** Address address;  **private** Date birthday;  **public** String getEmail() {  **return** email;  }  **...**  **public** Address getAddress() {  **return** address;  }  @DateTimeFormat(pattern="yyyyMMdd")  **public** **void** setBirthday(Date birthday) {  **this**.birthday = birthday;  }  } |
| **형식이 다르게 들어온다면**  **#메시지파일**  **typeMismatch.birthday=날짜형식이 올바르지 않습니다.** |
| **글로벌 변환기등록**  <mvc:annotation-driven conversion-service=*"formattingConversionService"* />  <bean id=*"formattingConversionService"*  class=*"org.springframework.format.support.FormattingConversionServiceFactoryBean"*>  <property name=*"formatters"*>  <set>  <bean class=*"net.madvirus.spring4.chap07.common.MoneyFormatter"* />  </set>  </property>  </bean>  클래스로 지정할때  @Configuration  @EnableWebMvc  **public** **class** SampleConfig **extends** WebMvcConfigurerAdapter {  @Override  **public** **void** configureDefaultServletHandling(DefaultServletHandlerConfigurer configurer) {  configurer.enable();  }  @Override  **public** **void** addFormatters(FormatterRegistry registry) {  registry.addFormatter(**new** MoneyFormatter());  } |

HandlerAdapter 파라미터 맵핑시킬때 Adapter처리

한다 뒤쪽에 HandlerMethodArgumentResolver 를 등록해서 사용한다.

|  |
| --- |
| <bean class="com.nhis.ggij.cms.nhis.vo.CustomRequestMappingHandlerAdapter">  <property name="webBindingInitializer">  <bean class="org.springframework.web.bind.support.ConfigurableWebBindingInitializer">  <property name="validator" ref="validator" />  <property name="conversionService" ref="conversion-service" />  </bean>  </property>  <property name="messageConverters" ref="messageConverters"/>   <!-- <mvc:argument-resolvers> 태그 존재시 -->  <property name="customArgumentResolvers">  <util:list>  <bean class="com.nhis.ggij.cms.nhis.CustomArgumentResolver" />  </util:list>  </property> </bean>  주의할점이 mvc:annotation-driven 와함게 사용할수 없다 |
| public class CustomRequestMappingHandlerAdapter extends RequestMappingHandlerAdapter {  ... @Override public void afterPropertiesSet() {  super.afterPropertiesSet();  }...  } |
| @Slf4j public class CustomArgumentResolver implements HandlerMethodArgumentResolver{  @Override  public boolean supportsParameter(MethodParameter parameter) {  return NoticeVO.class.isAssignableFrom(parameter.getParameterType());  }   @Override  public Object resolveArgument(MethodParameter parameter, ModelAndViewContainer mavContainer, NativeWebRequest webRequest, WebDataBinderFactory binderFactory) throws Exception {   *log*.debug(parameter.toString());  *log*.debug(mavContainer.toString());  *log*.debug(webRequest.toString());  *log*.debug(binderFactory.toString());  return null;  } } |

HTTP 세션 사용하기 p341

|  |
| --- |
| @Controller  **public** **class** LogoutController {  @RequestMapping("/auth/logout")  **public** String logout(HttpSession session) {  session.invalidate();  **return** "redirect:/index.jsp";  }  }  @RequestMapping("/auth/logout")  **public** String logout(HttpServletRequest request,**HttpSession session**) {  session.invalidate();  **return** "redirect:/index.jsp";  } |

@SessionAttributes어노테이션을 이용한 모델과 세션연동 p343

이벤트 생성 단계의 경우 3단계로 나눠져있다고 치면 3단계에서 2단계로 뒤로가기 눌러 간다던거 하게되면

다시 폼양식을 작성해야되는경우가 있다 이럴경우 세션을 공유하여 처리할수있다.

1. 클래스에 @SessionAttributes를 적용하고 세션으로 공유할 객체의 모델 이름을 지정한다
2. 컨트롤러 메서드에서 객체를 모델에 추가한다
3. 공유한 모델의 사용이 끝나면 SessionStatus를 사용해서 세션에서 객체를 제거한다.

|  |
| --- |
| @Controller  @SessionAttributes("eventForm")  **public** **class** EventCreationController {  **private** **static** **final** String ***EVENT\_CREATION\_STEP1*** = "event/creationStep1";  **private** **static** **final** String ***EVENT\_CREATION\_STEP2*** = "event/creationStep2";  **private** **static** **final** String ***EVENT\_CREATION\_STEP3*** = "event/creationStep3";  **private** **static** **final** String ***EVENT\_CREATION\_DONE*** = "event/creationDone";  // @RequestMapping("/newevent/step1")  // public String step1(Model model) {  // model.addAttribute("eventForm", new EventForm());  // return EVENT\_CREATION\_STEP1;  // }  @ModelAttribute("eventForm")  **public** EventForm formData() {  **return** **new** EventForm();  }  @RequestMapping("/newevent/step1")  **public** String step1(HttpSession session) {  **return** ***EVENT\_CREATION\_STEP1***;  }    @RequestMapping(value = "/newevent/step2", method = RequestMethod.***POST***)  **public** String step2(@ModelAttribute("eventForm") EventForm formData, BindingResult result,HttpSession session) {  **new** EventFormStep1Validator().validate(formData, result);  **if** (result.hasErrors())  **return** ***EVENT\_CREATION\_STEP1***;  **return** ***EVENT\_CREATION\_STEP2***;  }  @RequestMapping(value = "/newevent/step2", method = RequestMethod.***GET***)  **public** String step2FromStep3(@ModelAttribute("eventForm") EventForm formData) {  **return** ***EVENT\_CREATION\_STEP2***;  }  @RequestMapping(value = "/newevent/step3", method = RequestMethod.***POST***)  **public** String step3(@ModelAttribute("eventForm") EventForm formData, BindingResult result) {  ValidationUtils.*rejectIfEmpty*(result, "target", "required");  **if** (result.hasErrors())  **return** ***EVENT\_CREATION\_STEP2***;  **return** ***EVENT\_CREATION\_STEP3***;  }  **@RequestMapping(value = "/newevent/done", method = RequestMethod.*POST*)**  **public String done(@ModelAttribute("eventForm") EventForm formData, SessionStatus sessionStatus) {**  **sessionStatus.setComplete();**  **return *EVENT\_CREATION\_DONE*;**  **}**  } |
| <%@ page contentType=*"text/html; charset=utf-8"* %>  <%@ page import=*"net.madvirus.spring4.chap07.event.EventType"* %>  <%@ taglib prefix=*"c"* uri=*"http://java.sun.com/jsp/jstl/core"* %>  <%@ taglib prefix=*"fmt"* uri=*"http://java.sun.com/jsp/jstl/fmt"* %>  <%@ taglib prefix=*"form"* uri=*"http://www.springframework.org/tags/form"* %>  <!DOCTYPE html>  <html>  <head>  <title>이벤트 생성: ${project}</title>  </head>  <body>  <form:form commandName=*"eventForm"* action=*"/spring4-chap07/newevent/step2"*>  <label for=*"name"*>이벤트 명</label>:  <input type=*"text"* name=*"name"* id=*"name"* value=*"*${eventForm.name}*"*/>  <form:errors path=*"name"*/><br/>  <label for=*"type"*>타입</label>:  <select name=*"type"* id=*"type"*>  <option value=*""*>선택하세요</option>  <c:forEach var=*"type"* items=*"*<%= EventType.values() %>*"*>  <option value=*"*${type}*"* ${eventForm.type == type ? 'selected' : ''}>${type}</option>  </c:forEach>  </select>  <form:errors path=*"type"*/><br/>  <label>이벤트 기간</label>:  <input type=*"text"* name=*"beginDate"* value=*'*<fmt:formatDate value="${eventForm.beginDate}" pattern=*"yyyyMMdd"*/>*'*/>부터  <input type=*"text"* name=*"endDate"* value=*'*<fmt:formatDate value="${eventForm.endDate}" pattern=*"yyyyMMdd"*/>*'*/>까지  <form:errors path=*"beginDate"*/><br/>  <form:errors path=*"endDate"*/><br/>  <input type=*"submit"* value=*"다음 단계로"* />  </form:form>  세션 존재 여부: <%= session.getAttribute("eventForm") != **null** ? "존재" : "없음" %>  </body>  </html> |
| 2가지 방법이있다 위에 주석처리되어있는것처럼 step1에서 처리하는방법 또는  @ModelAttribute("eventForm")  **public** EventForm formData() {  **return** **new** EventForm();  }  처리하는방법이있다.  세션해제는 done메서드에보면 **SessionStatus sessionStatus --> sessionStatus.setComplete();** |

익셉션처리 p349

컨트롤러 메서드를 실행하는 도중에 익셉션을 발생하면 예쁘지 안항요~ ㅋㅋ

1. ExceptionHandler어노테이션이용
2. ControllerAdvice어노테이션이용
3. ResponseStatus어노테이션이용

|  |
| --- |
| @Controller  **public** **class** CalculationController {  @RequestMapping("/cal/add")  **public** String add(  @RequestParam("op1") **int** op1,  @RequestParam("op2") **int** op2,  Model model) {  model.addAttribute("result", op1 + op2);  **return** "cal/result";  }  @RequestMapping("/cal/divide")  **public** String divide(Model model,  @RequestParam("op1") **int** op1, @RequestParam("op2") **int** op2) {  model.addAttribute("result", op1 / op2);  **return** "cal/result";  }  @ExceptionHandler(RuntimeException.**class**)  **public** String handleException(HttpServletResponse response) {  response.setStatus(HttpServletResponse.***SC\_INTERNAL\_SERVER\_ERROR***);  **return** "error/exception";  }  /\*  Exception객체 직접 접근  @ExceptionHandler  **public** String handleException(ArithmeticException exception) {  exception.printStackTrace();  **return** "error/exception";  }  \*/  }  위내용은 해당 컨트롤러만 해당되는것이다. |
| exception.jsp  <%@ page contentType=*"text/html; charset=utf-8"* %>  **<%@ page isErrorPage=*"true"* %>**  <%@ taglib prefix=*"c"* uri=*"http://java.sun.com/jsp/jstl/core"* %>  <!DOCTYPE html>  <html>  <head>  <title>에러 발생</title>  </head>  <body>  작업 처리 도중 문제가 발생했습니다.  <%= exception %>  </body>  </html> |

@ControllerAdvice 공통 익셉션처리

|  |
| --- |
| <bean class="net.madvirus.spring4.chap07.exhandler.CommonExceptionHandler"/> |
| @ControllerAdvice("net.madvirus.spring4.chap07")  **public** **class** CommonExceptionHandler {  @ExceptionHandler(RuntimeException.**class**)  **public** String handleException() {  **return** "error/commonException";  }  } |

@ResponseStatus이용 익셉션의 응답 코드 설정

자체 응답코드 설정

|  |
| --- |
| @Controller  **public** **class** FileController {  @RequestMapping(value = "/files/{fileId:[a-zA-Z]\\d{3}}", method = RequestMethod.***GET***)  **public** String fileInfo(@PathVariable String fileId) **throws** NoFileInfoException {  FileInfo fileInfo = getFileInfo(fileId);  **if** (fileInfo == **null**) {  **throw** **new** NoFileInfoException();  }  **return** "files/fileInfo";  } |
| @ResponseStatus(HttpStatus.***NOT\_FOUND***)  **public** **class** NoFileInfoException **extends** Exception {  **private** **static** **final** **long** ***serialVersionUID*** = 1L;  } |
| 서비스/도메인/영속성 영역의 익셉션에 @ResponseStatus어노테이션을 적용하지 않도록 하자. UI처리의 의미를 내포하고 있기때문에. |

스프링MVC설정p357

디폴트 서블릿 설정과 동작방식p361

|  |
| --- |
| <servlet>  <servlet-name>dispatcher3</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>  /WEB-INF/sample.xml  </param-value>  </init-param>  <load-on-startup>1</load-on-startup>  </servlet>  <servlet-mapping>  <servlet-name>dispatcher3</servlet-name>  <url-pattern>/</url-pattern>  </servlet-mapping> |
| sample.xml  <mvc:default-servlet-handler />   1. 요청 경로와 일치하는 컨트롤러를 찾는다 2. 컨트롤러가 존재하지 않으면 디폴트 서블릿 핸들러에 전달한다 3. 디폴트 서블릿 핸들러는 WAS의 디폴트 서블릿에 처리위임한다 4. 디폴트 서블릿의 처리결과를 응답으로 전송한다 |

디폴트 서블릿의 이름은 WAs마다 다른데 디폴트 서블릿 핸들러 설정에서 디폴트 서블릿의 이름을 지정하고 싶다면 다음과같이 사용하면된다

<mvc:default-servlet-handler default-servlet-name=”default”/>

**public** **class** SampleConfig **extends** WebMvcConfigurerAdapter {

@Override

**public** **void** configureDefaultServletHandling(DefaultServletHandlerConfigurer configurer) {

configurer.enable("default");

}

|  |
| --- |
| 클래스로 설정  @Configuration  @EnableWebMvc  **public** **class** SampleConfig **extends** WebMvcConfigurerAdapter {  @Override  **public** **void** configureDefaultServletHandling(DefaultServletHandlerConfigurer configurer) {  configurer.enable();  }  @Override  **public** **void** addFormatters(FormatterRegistry registry) {  registry.addFormatter(**new** MoneyFormatter());  }  @Override  **public** **void** addViewControllers(ViewControllerRegistry registry) {  registry.addViewController("/index").setViewName("index");  }  @Override  **public** **void** addResourceHandlers(ResourceHandlerRegistry registry) {  registry.addResourceHandler("/images/\*\*")  .addResourceLocations("/images/", "/WEB-INF/resources/")  .setCachePeriod(60);  }  @Override  **public** **void** addInterceptors(InterceptorRegistry registry) {  registry.addInterceptor(**new** AuthInterceptor()).addPathPatterns("/acl/\*\*");  registry.addInterceptor(measuringInterceptor()).addPathPatterns("/\*\*");  registry.addInterceptor(commonModelInterceptor())  .addPathPatterns("/acl/\*\*", "/header/\*\*", "/newevent/\*\*")  .excludePathPatterns("/acl/modify");  }  @Bean  **public** MeasuringInterceptor measuringInterceptor() {  **return** **new** MeasuringInterceptor();  }  @Bean  **public** CommonModelInterceptor commonModelInterceptor() {  **return** **new** CommonModelInterceptor();  }  @Bean  **public** ViewResolver viewResolver() {  InternalResourceViewResolver result = **new** InternalResourceViewResolver();  result.setPrefix("/WEB-INF/view/");  result.setSuffix(".jsp");  **return** result;  }  @Bean  **public** MemberService memberService() {  **return** **new** MemberService();  }  @Bean  **public** EventController eventController() {  **return** **new** EventController();  }  @Bean  **public** EventCreationController eventCreationController() {  **return** **new** EventCreationController();  }  @Bean  **public** RegistrationController registrationController() {  RegistrationController result = **new** RegistrationController();  result.setMemberService(memberService());  **return** result;  }  @Bean  **public** MemberController memberController() {  MemberController result = **new** MemberController();  result.setMemberService(memberService());  **return** result;  }  @Bean  **public** MemberModificationController memberModController() {  MemberModificationController result = **new** MemberModificationController();  result.setMemberService(memberService());  **return** result;  }  @Bean  **public** FileController fileController() {  **return** **new** FileController();  }  @Bean  **public** SearchController searchController() {  **return** **new** SearchController();  }  @Bean  **public** SimpleHeaderController simpleHeaderController() {  **return** **new** SimpleHeaderController();  }  @Bean  **public** AclService aclService() {  **return** **new** AclService();  }  @Bean  **public** ACLController aclController() {  ACLController aclController = **new** ACLController();  aclController.setAclService(aclService());  **return** aclController;  }  @Bean  **public** Authenticator authenticator() {  **return** **new** Authenticator(memberService());  }  @Bean  **public** LoginController loginController() {  LoginController result = **new** LoginController();  result.setAuthenticator(authenticator());  **return** result;  }  @Bean  **public** LogoutController logoutController() {  **return** **new** LogoutController();  }  @Bean  **public** CalculationController calculationController() {  **return** **new** CalculationController();  }  // @Bean  // public CommonExceptionHandler commonExceptionHandler() {  // return new CommonExceptionHandler();  // }  @Bean  **public** MessageSource messageSource() {  ResourceBundleMessageSource ms = **new** ResourceBundleMessageSource();  ms.setBasenames("message.error");  ms.setDefaultEncoding("UTF-8");  **return** ms;  }  } |
| sample.xml설정  <mvc:resources mapping=*"/images/\*\*"* location=*"/images/, /WEB-INF/resources/"*  cache-period=*"60"* /> |

정적 자원 설정하기

|  |
| --- |
| <mvc:resources mapping=*"/images/\*\*"* location=*"/images/, /WEB-INF/resources/"*  cache-period=*"60"* /> |
| @Override  **public** **void** addResourceHandlers(ResourceHandlerRegistry registry) {  registry.addResourceHandler("/images/\*\*")  .addResourceLocations("/images/", "/WEB-INF/resources/")  .setCachePeriod(60);  } |

HandlerInterceptor이용한 인터셉터 구현 p363

요청 경로마다 접근 제어를 다르게 해야 한다거나 사용자가 특정 URL을 요청할 떄마다 접근 내역을 기록하고 싶다면 어떻게 해야할까

이런 기능은 특정 컨트롤러에 종속되기보다는 여러 컨토로로에 공통으로 적용되는 기능이다. 따라서 중복코드 없이 컨트롤 적용하는 방법이 필요하당.

1. 컨트롤러(핸들로)실행전
2. 컨트롤러(핸들러)실행후, 아직 뷰를 실행하기전
3. 뷰를 실행한후

|  |  |
| --- | --- |
|  | perHandle 객체실행하기전  postHandle 실행된후  after뷰전송후 |

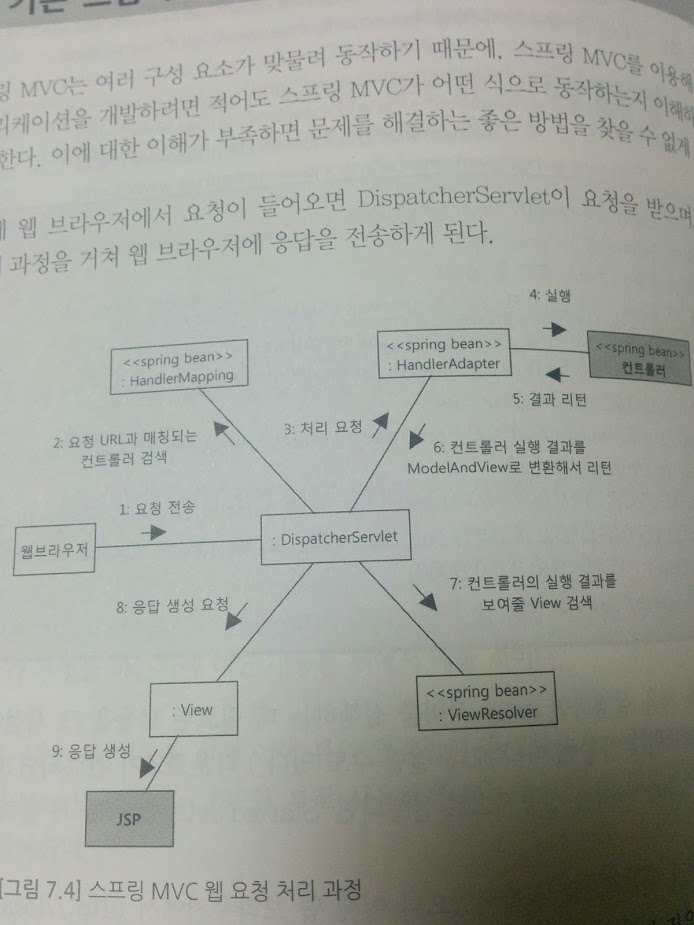
|  |
| --- |
| <mvc:interceptors>  <mvc:interceptor>  <mvc:mapping path=*"/acl/\*\*"* />  <bean class=*"net.madvirus.spring4.chap07.common.AuthInterceptor"* />  </mvc:interceptor>  <bean class=*"net.madvirus.spring4.chap07.common.MeasuringInterceptor"* />  <mvc:interceptor>  <mvc:mapping path=*"/acl/\*\*"* />  <mvc:mapping path=*"/header/\*\*"* />  <mvc:mapping path=*"/newevent/\*\*"* />  <mvc:exclude-mapping path=*"/acl/modify"* />  <ref bean=*"commonModelInterceptor"* />  </mvc:interceptor>  </mvc:interceptors> |
| **public** **class** MeasuringInterceptor **extends** HandlerInterceptorAdapter {  @Override  **public** **boolean** preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) **throws** Exception {  System.***out***.println("MI: preHandle()");  request.setAttribute("mi.beginTime", System.*currentTimeMillis*());  **return** **true**;  }  @Override  **public** **void** afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handler, Exception ex) **throws** Exception {  System.***out***.println("MI: afterCompletion()");  Long beginTime = (Long) request.getAttribute("mi.beginTime");  **long** endTime = System.*currentTimeMillis*();  System.***out***.println(request.getRequestURI() + " 실행 시간: " + (endTime - beginTime));  }  @Override  **public** **void** postHandle(HttpServletRequest request, HttpServletResponse response, Object handler, ModelAndView modelAndView) **throws** Exception {  System.***out***.println("MI: postHandle()");  }    } |
| @Override  **public** **void** addInterceptors(InterceptorRegistry registry) {  registry.addInterceptor(**new** AuthInterceptor()).addPathPatterns("/acl/\*\*");  registry.addInterceptor(measuringInterceptor()).addPathPatterns("/\*\*");  registry.addInterceptor(commonModelInterceptor())  .addPathPatterns("/acl/\*\*", "/header/\*\*", "/newevent/\*\*")  .excludePathPatterns("/acl/modify");  } |

DelegatingFilterProxy를 이용한 서블릿 필터 등록p375

...

Chapter08

스프링 MVC 뷰 구현



ViewResolver설정

|  |  |
| --- | --- |
|  |  |

상속받아서 처리할수 있다

HTML 특수문자 처리 방식설정

|  |
| --- |
| <web-app>  <context-param>  <param-name>defaultHtmlEscape</param-name>  <param-value>true</param-value>  </context-param>  </web-app> |

메시지 출력을 위한 <spring:message>커스텀 태그

|  |  |
| --- | --- |
| <bean id=*"messageSource"*  class=*"org.springframework.context.support.ResourceBundleMessageSource"*>  <property name=*"basenames"*>  <list>  <value>message.error</value>  <value>message.label</value>  </list>  </property>  <property name=*"defaultEncoding"* value=*"UTF-8"* />  </bean> | |
| email=이메일  password=암호  login.form.title=로그인 폼  login.form.login=로그인  login.form.help=이메일/암호로 yuna@yuna.com/yuna 입력 테스트  login.form.type=로그인 유형  greeting=**{0}** 회원님, **{1}** |  |
| {0}등은 palce holder로서 값설정가능하다  <spring:message code=*"greeting"* arguments=*"*${me}*,* ${greeting}*"* /> |  |
| <%@ page contentType=*"text/html; charset=utf-8"* %>  <%@ taglib prefix=*"spring"* uri=*"http://www.springframework.org/tags"* %>  <%@ taglib prefix=*"form"* uri=*"http://www.springframework.org/tags/form"* %>  <!DOCTYPE html>  <html>  <head><title><spring:message code=*"login.form.title"*/></title></head>  <body>  <form:form commandName=*"loginCommand"*>  <form:hidden path=*"securityLevel"*/>  <form:errors element=*"div"* />  <p>  <label for=*"email"*><spring:message code=*"email"* /></label>:  <input type=*"text"* name=*"email"* id=*"email"* value=*"*${loginCommand.email}*"*>  <form:errors path=*"email"*/>  </p>  <p>  <label for=*"password"*><spring:message code=*"password"* /></label>:  <input type=*"password"* name=*"password"* id=*"password"*>  <form:errors path=*"password"*/>  </p>  <p>  <label for=*"loginType"*><spring:message code=*"login.form.type"* /></label>  <form:select path=*"loginType"* items=*"*${loginTypes}*"* />  </p>  <input type=*"submit"* value=*"*<spring:message code=*"login.form.login"* />*"*>  </form:form>  <ul>  <li><spring:message code=*"login.form.help"* /></li>  </ul>  </body>  </html> | |

자바스크립트에서 처리하고 싶다면

|  |
| --- |
| <script>  var a = ‘<spring:message code=”title” javaScriptEscape=”true”/>’  </script> |

스프링에서 제공하는 폼관련 커스텀 태그 p391 ~402

값포맷팅 <form:input> p403

<spring:htmlEscape> 커스텀태그 p407

파일다운로드 구현을 위한 커스텀 view p401

액셀다운로드구현 413

pdf구현 417

Local처리 p420

Chapter09 스프링 MVC: XML/JSON, 파일업로드 웹소켓 p 426

...

Chapter11 데이터 베이스 연동과 JDBC지원 p474

1. 커넥션 풀을 이용한 DataSource설정
2. JNDI를통한 DataSource설정
3. DriverManager이용 DataSource설정

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans.xsd"*>  <bean id=*"dataSource"* class=*"com.mchange.v2.c3p0.ComboPooledDataSource"*  destroy-method=*"close"*>  <property name=*"driverClass"* value=*"com.mysql.jdbc.Driver"* />  <property name=*"jdbcUrl"*  value=*"jdbc:mysql://localhost/logersstest?characterEncoding=utf8"* />  <property name=*"user"* value=*"root"* />  <property name=*"password"* value=*"javadev"* />  </bean>  <bean id=*"jdbcMessageDao"* class=*"net.madvirus.spring4.chap11.guest.jdbc.JdbcMessageDao"*>  <property name=*"dataSource"* ref=*"dataSource"* />  </bean> |
| **private** **static** **void** runMessageDao(MessageDao messageDao) {  Message message = **new** Message();  message.setMessage("메시지");  message.setCreationTime(**new** Date());  message.setName("최범균");  **int** id = messageDao.insert(message);  System.***out***.printf("Message[%d]가 추가되었습니다.\n", id);  **int** count = messageDao.counts();  System.***out***.printf("전체 개수: %d\n", count);  List<Message> messages = messageDao.select(0, 10);  System.***out***.printf("읽어온 메시지 개수: %d\n", messages.size());  } |
| **public** **class** JdbcMessageDao **implements** MessageDao {  **private** DataSource dataSource;  **private** SQLExceptionTranslator exceptionTranslator;  **public** **void** setDataSource(DataSource dataSource) {  **this**.dataSource = dataSource;  **this**.exceptionTranslator = **new** SQLErrorCodeSQLExceptionTranslator(  dataSource);  }  @Override  **public** List<Message> select(**int** start, **int** size) {  Connection conn = **null**;  PreparedStatement pstmt = **null**;  ResultSet rs = **null**;  String sql = "select \* from guestmessage order by id desc limit ?, ?";  **try** {  conn = dataSource.getConnection();  pstmt = conn.prepareStatement(sql);  pstmt.setInt(1, start);  pstmt.setInt(2, size);  rs = pstmt.executeQuery();  **if** (rs.next()) {  List<Message> messages = **new** ArrayList<>();  **do** {  Message m = **new** Message();  m.setId(rs.getInt("id"));  m.setName(rs.getString("name"));  m.setMessage(rs.getString("message"));  m.setCreationTime(rs.getTimestamp("creationTime"));  messages.add(m);  } **while** (rs.next());  **return** messages;  } **else** {  **return** Collections.*emptyList*();  }  } **catch** (SQLException ex) {  **throw** exceptionTranslator.translate("select", sql, ex);  } **finally** {  JdbcUtils.*closeResultSet*(rs);  JdbcUtils.*closeStatement*(pstmt);  JdbcUtils.*closeConnection*(conn);  }  }  @Override  **public** **int** counts() {  Connection conn = **null**;  Statement stmt = **null**;  ResultSet rs = **null**;  String sql = "select count(\*) from guestmessage";  **try** {  conn = dataSource.getConnection();  stmt = conn.createStatement();  rs = stmt.executeQuery(sql);  **if** (rs.next()) {  **return** rs.getInt(1);  } **else** {  **return** 0;  }  } **catch** (SQLException ex) {  **throw** exceptionTranslator.translate("counts", sql, ex);  } **finally** {  JdbcUtils.*closeResultSet*(rs);  JdbcUtils.*closeStatement*(stmt);  JdbcUtils.*closeConnection*(conn);  }  }  @Override  **public** **int** insert(Message message) {  Connection conn = **null**;  PreparedStatement pstmt = **null**;  Statement stmt = **null**;  ResultSet rs = **null**;  String sql = "insert into guestmessage (name, message, creationTime) values (?,?,?)";  **try** {  conn = dataSource.getConnection();  pstmt = conn.prepareStatement(sql);  pstmt.setString(1, message.getName());  pstmt.setString(2, message.getMessage());  pstmt.setTimestamp(3, **new** Timestamp(message.getCreationTime()  .getTime()));  **int** insertedCount = pstmt.executeUpdate();  **if** (insertedCount > 0) {  stmt = conn.createStatement();  rs = stmt  .executeQuery("select last\_insert\_id() from guestmessage");  **if** (rs.next()) {  **return** rs.getInt(1);  }  }  **return** -1;  } **catch** (SQLException ex) {  **throw** exceptionTranslator.translate("insert", sql, ex);  } **finally** {  JdbcUtils.*closeResultSet*(rs);  JdbcUtils.*closeStatement*(stmt);  JdbcUtils.*closeStatement*(pstmt);  JdbcUtils.*closeConnection*(conn);  }  }  @Override  **public** **int** delete(**int** id) {  Connection conn = **null**;  PreparedStatement pstmt = **null**;  String sql = "delete from guestmessage where id = ?";  **try** {  conn = dataSource.getConnection();  pstmt = conn.prepareStatement(sql);  pstmt.setInt(1, id);  **return** pstmt.executeUpdate();  } **catch** (SQLException ex) {  **throw** exceptionTranslator.translate("delete", sql, ex);  } **finally** {  JdbcUtils.*closeStatement*(pstmt);  JdbcUtils.*closeConnection*(conn);  }  } |

JdbcTemplate

|  |
| --- |
| <bean id=*"jdbcTemplateMessageDao"*  class=*"net.madvirus.spring4.chap11.guest.jdbc.JdbcTemplateMessageDao"*>  <constructor-arg ref=*"dataSource"* />  </bean> |
| **blic** **class** JdbcTemplateMessageDao **implements** MessageDao {  **private** JdbcTemplate jdbcTemplate;  **public** JdbcTemplateMessageDao(DataSource dataSource) {  jdbcTemplate = **new** JdbcTemplate(dataSource);  }  **private** RowMapper<Message> messageRowMapper = **new** MessageRowMapper();  @Override  **public** List<Message> select(**int** start, **int** size) {  List<Message> messages = jdbcTemplate.query(  "select \* from guestmessage order by id desc limit ?, ?",  **new** Object[] { start, size },  messageRowMapper  );  **return** messages;  }  @Override  **public** **int** counts() {  **return** jdbcTemplate.queryForObject(  "select count(\*) from guestmessage",  Integer.**class**);  }  @Override  **public** **int** insert(**final** Message message) {  KeyHolder keyHolder = **new** GeneratedKeyHolder();  jdbcTemplate.update(**new** PreparedStatementCreator() {  @Override  **public** PreparedStatement createPreparedStatement(Connection conn)  **throws** SQLException {  PreparedStatement pstmt = conn  .prepareStatement(  "insert into guestmessage (name, message, creationTime) values (?,?,?)",  **new** String[] { "id" });  pstmt.setString(1, message.getName());  pstmt.setString(2, message.getMessage());  pstmt.setTimestamp(3, **new** Timestamp(message.getCreationTime()  .getTime()));  **return** pstmt;  }  }, keyHolder);  Number idNum = keyHolder.getKey();  **return** idNum.intValue();  }  @Override  **public** **int** delete(**int** id) {  **return** jdbcTemplate.update("delete from guestmessage where id = ?", id);  }  } |
| <bean id=*"dataSource"* class=*"org.springframework.jdbc.datasource.DriverManagerDataSourc"*  destroy-method=*"close"*>  <property name=*"driverClass"* value=*"com.mysql.jdbc.Driver"* />  <property name=*"jdbcUrl"*  value=*"jdbc:mysql://localhost/logersstest?characterEncoding=utf8"* />  <property name=*"user"* value=*"root"* />  <property name=*"password"* value=*"javadev"* />  </bean> |

NamedJdbcTemplate

|  |
| --- |
| **public** **class** NamedJdbcTemplateMessageDao **implements** MessageDao {  **private** NamedParameterJdbcTemplate template;  **public** NamedJdbcTemplateMessageDao(DataSource dataSource) {  **this**.template = **new** NamedParameterJdbcTemplate(dataSource);  }  @Override  **public** List<Message> select(**int** start, **int** size) {  Map<String, Object> params = **new** HashMap<>();  params.put("start", start);  params.put("size", size);  List<Message> messages = template.query(  "select \* from guestmessage order by id desc limit :start, :size",  params,  **new** MessageRowMapper()  );  **return** messages;  }  @Override  **public** **int** counts() {  **return** template.queryForObject(  "select count(\*)from guestmessage",  Collections.<String, Object>*emptyMap*(),  Integer.**class**);  //return template.getJdbcOperations().queryForObject("select count(\*) from guestmessage", Integer.class);  }  @Override  **public** **int** insert(Message message) {  SqlParameterSource paramSource = **new** BeanPropertySqlParameterSource(message);  KeyHolder keyHolder = **new** GeneratedKeyHolder();  template.update("insert into guestmessage (name, message, creationTime) values (:name, :message,:creationTime)", paramSource, keyHolder);  Number idNum = keyHolder.getKey();  **return** idNum.intValue();  }  @Override  **public** **int** delete(**int** id) {  MapSqlParameterSource paramSource = **new** MapSqlParameterSource();  paramSource.addValue("id", id);  **return** template.update("delete from guestmessage where id = :id", paramSource);  }  } |

SimpleJdbcInsert

|  |
| --- |
| **public** **class** SimpleInsertMessageDao **implements** MessageDao {  **private** MessageDao delegate;  **private** SimpleJdbcInsert simpleInsert;  **public** SimpleInsertMessageDao(DataSource dataSource) {  simpleInsert = **new** SimpleJdbcInsert(dataSource);  simpleInsert.withTableName("guestmessage")  .usingColumns("name", "message", "creationTime")  .setGeneratedKeyName("id");    delegate = **new** JdbcTemplateMessageDao(dataSource);  }  @Override  **public** **int** insert(Message message) {  /\*  Map<String, Object> values = new HashMap<>();  values.put("NAME", message.getName());  values.put("message", message.getMessage());  values.put("creationTime", new Timestamp(message.getCreationTime().getTime()));  Number genKey = simpleInsert.executeAndReturnKey(values);  \*/  BeanPropertySqlParameterSource paramSource =  **new** BeanPropertySqlParameterSource(message);  Number genKey = simpleInsert.executeAndReturnKey(paramSource);  **return** genKey.intValue();  }  **public** List<Message> select(**int** start, **int** size) {  **return** delegate.select(start, size);  }  **public** **int** counts() {  **return** delegate.counts();  }  **public** **int** delete(**int** id) {  **return** delegate.delete(id);  }  } |

JDBCTemplate

|  |
| --- |
| **public** **class** JdbcTemplateMessageDao **implements** MessageDao {  **private** JdbcTemplate jdbcTemplate;  **public** JdbcTemplateMessageDao(DataSource dataSource) {  jdbcTemplate = **new** JdbcTemplate(dataSource);  }  @Override  **public** List<Message> select(**int** start, **int** size) {  List<Message> messages = jdbcTemplate.query(  "select \* from guestmessage order by id desc limit ?, ?",  **new** Object[]{start, size},  (rs, rowNum) -> {  Message m = **new** Message();  m.setId(rs.getInt("id"));  m.setName(rs.getString("name"));  m.setMessage(rs.getString("message"));  m.setCreationTime(rs.getTimestamp("creationTime"));  **return** m;  }  );  **return** messages;  }  @Override  **public** **int** counts() {  **return** jdbcTemplate.queryForObject(  "select count(\*) from guestmessage",  Integer.**class**);  }  @Override  **public** **int** insert(Message message) {  KeyHolder keyHolder = **new** GeneratedKeyHolder();  jdbcTemplate.update((Connection conn) -> {  PreparedStatement pstmt = conn  .prepareStatement(  "insert into guestmessage (name, message, creationTime) values (?,?,?)",  **new** String[]{"id"});  pstmt.setString(1, message.getName());  pstmt.setString(2, message.getMessage());  pstmt.setTimestamp(3, **new** Timestamp(message.getCreationTime()  .getTime()));  **return** pstmt;  }  , keyHolder);  Number idNum = keyHolder.getKey();  **return** idNum.intValue();  }  } |

트랜젝션관리 p 507

..

ORM연동지원 537

스프링 데이터 JPA소개 577

스프링 웹개발2

Chapter15 웹 어플리케이션 구조 628

|  |  |
| --- | --- |
| 모델 | 데이터 베이스 테이블과 관련된 클래스가 위치한다 CRUD의 기본 단위로 사용되며 서비스 –DAO간 데이터를 주고 받기 위한 객체로도 사용된다 |
| DAO | DAta Access Object 의 약자로 데이터 베이스 테이블에 대한 CRUD기능을 정의한다 데이터 입력이나 수정 조회시 데이터 타입으로 모델사용 |
| 서비스 | 컨트롤러 통해서 전달받은 사용자의 요청을 구현한다 DB연산이 필요한 경우 DAO를 이용한다 |
| 컨트롤러 | 사용자의 웹 요청을 받아 서비스나 DAO에 전달하고 결과를 뷰에 전달한다 |

...

Chapter16 웹보안 스프링 시큐리티 652

1. 인증처리: 현재 사용자가 누구인지 확인하는 과정
2. 인가처리:현재 사용자가 특정 대상URL기능등을 사용권한 있는지
3. UI처리 권한이 없는 사용자가 접근했을때 알맞은 화면 처리

spring-security-acl-3.2.4.RELEASE.jar

spring-security-config-3.2.4.RELEASE.jar

spring-security-core-3.2.4.RELEASE.jar

spring-security-taglibs-3.2.4.RELEASE.jar

spring-security-web-3.2.4.RELEASE.jar

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  **xmlns:sec=*"http://www.springframework.org/schema/security"***  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans.xsd*  *http://www.springframework.org/schema/security*  *http://www.springframework.org/schema/security/spring-security.xsd"*>  <sec:http use-expressions=*"true"*>  <sec:intercept-url pattern=*"/admin/\*\*"*  **access=*"hasAuthority('ROLE\_ADMIN')"* />**  <sec:intercept-url pattern=*"/manager/\*\*"*  **access=*"hasRole('ROLE\_MANAGER')"* />**  <sec:intercept-url pattern=*"/member/\*\*"* access=*"isAuthenticated()"* />  <sec:intercept-url pattern=*"/\*\*"* access=*"permitAll"*/>  <sec:form-login />  <sec:logout />  </sec:http>  <sec:authentication-manager>  <sec:authentication-provider>  <sec:user-service>  <sec:user name=*"bkchoi"* password=*"1234"*  authorities=*"ROLE\_USER"* />  <sec:user name=*"manager"* password=*"qwer"*  authorities=*"ROLE\_MANAGER"* />  <sec:user name=*"admin"* password=*"asdf"*  authorities=*"ROLE\_ADMIN,ROLE\_USER"* />  </sec:user-service>  </sec:authentication-provider>  </sec:authentication-manager>  </beans> |
| web.xml  <filter>  <filter-name>springSecurityFilterChain</filter-name>  <filter-class>  org.springframework.web.filter.DelegatingFilterProxy  </filter-class>  </filter>  <filter-mapping>  <filter-name>springSecurityFilterChain</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>  <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:/spring-security.xml</param-value>  </context-param> |
| <%@ page contentType=*"text/html; charset=utf-8"* %>  <%@ taglib prefix=*"c"* uri=*"http://java.sun.com/jsp/jstl/core"* %>  <%@ taglib prefix=*"sec"* uri=*"http://www.springframework.org/security/tags"* %>  <html>  <head>  <title>어드민 메인</title>  </head>  <body>  어드민(연결 계정: <sec:authentication property=*"name"*/>) 메인 화면입니다.  <br/>  <a href=*"*<c:url value=*'/index'*/>*"*>[/index로 가기]</a>  </body>  </html> |

스프링 시큐리티 구조 개요 p668

...

JSR https://ko.wikipedia.org/wiki/%EC%9E%90%EB%B0%94\_%EC%BB%A4%EB%AE%A4%EB%8B%88%ED%8B%B0\_%ED%94%84%EB%A1%9C%EC%84%B8%EC%8A%A4

# 자바 커뮤니티 프로세스

위키백과, 우리 모두의 백과사전.

**자바 커뮤니티 프로세스**(Java Community Process, 약자 **JCP**)는 1998년에 설립되어, 이해관계자간에 [자바 플랫폼](https://ko.wikipedia.org/wiki/%EC%9E%90%EB%B0%94_%ED%94%8C%EB%9E%AB%ED%8F%BC)의 향후 버전 및 기능에 대한 정의에 관여하는 표준화 과정이다.

JCP 과정에는 자바 스펙 요구서(Java Specification Request, 약자 JSR)을 이용하는데 JSR은 [자바 플랫폼](https://ko.wikipedia.org/wiki/%EC%9E%90%EB%B0%94_%ED%94%8C%EB%9E%AB%ED%8F%BC)에 추가된 사양 및 기술을 기술하는 공식 문서이다. JSR의 공식 공개 리뷰는 JSR이 확정되기 전에 실행하고 JCP의 최상위 커뮤니티에서 투표를 한다. 최종판은 소스코드 형식의 기술에 관련되어 자유롭게 구현이 가능하도록 참조구현과 API 규격을 확인하는 기술호환 장치를 제공한다.

## JSR[[편집](https://ko.wikipedia.org/w/index.php?title=%EC%9E%90%EB%B0%94_%EC%BB%A4%EB%AE%A4%EB%8B%88%ED%8B%B0_%ED%94%84%EB%A1%9C%EC%84%B8%EC%8A%A4&action=edit&section=1)]

**자바 스펙 요구서**(Java Specification Request,**JSR**)은 자바 플랫폼에 추가된 사양 및 기술을 기술하는 공식 문서이다. 현재 JSR은 다음과 같다.

|  |  |
| --- | --- |
| **JSR #** | **Specification or Technology** |
| [1](http://www.jcp.org/en/jsr/detail?id=1) | [Real-Time Specification for Java](https://ko.wikipedia.org/w/index.php?title=Real-Time_Specification_for_Java&action=edit&redlink=1) (RTSJ) 1.0 |
| [3](http://www.jcp.org/en/jsr/detail?id=3) | [Java Management Extensions](https://ko.wikipedia.org/wiki/Java_Management_Extensions) (JMX) 1.0, 1.1, & 1.2[[1]](https://ko.wikipedia.org/wiki/%EC%9E%90%EB%B0%94_%EC%BB%A4%EB%AE%A4%EB%8B%88%ED%8B%B0_%ED%94%84%EB%A1%9C%EC%84%B8%EC%8A%A4#cite_note-1) |
| [5](http://www.jcp.org/en/jsr/detail?id=5) | [Java API for XML Processing](https://ko.wikipedia.org/wiki/Java_API_for_XML_Processing) (JAXP) 1.0 |
| [9](http://www.jcp.org/en/jsr/detail?id=9) | [Jiro](https://ko.wikipedia.org/w/index.php?title=Jiro&action=edit&redlink=1) (Federated Management Architecture Specification) 1.0 |
| [12](http://www.jcp.org/en/jsr/detail?id=12) | [Java Data Objects](https://ko.wikipedia.org/w/index.php?title=Java_Data_Objects&action=edit&redlink=1) (JDO) 1.0 |
| [13](http://www.jcp.org/en/jsr/detail?id=13) | Improved BigDecimal ([Java Platform, Standard Edition#java.math](https://ko.wikipedia.org/wiki/Java_Platform,_Standard_Edition#java.math)) |
| [14](http://www.jcp.org/en/jsr/detail?id=14) | Add [Generic Types](https://ko.wikipedia.org/w/index.php?title=Generic_programming&action=edit&redlink=1) To The Java Programming Language (as of J2SE 5.0) |
| [16](http://www.jcp.org/en/jsr/detail?id=16) | [Java EE Connector Architecture](https://ko.wikipedia.org/wiki/Java_EE_Connector_Architecture) (JCA) 1.0 |
| [19](http://www.jcp.org/en/jsr/detail?id=19) | [엔터프라이즈 자바빈즈](https://ko.wikipedia.org/wiki/%EC%97%94%ED%84%B0%ED%94%84%EB%9D%BC%EC%9D%B4%EC%A6%88_%EC%9E%90%EB%B0%94%EB%B9%88%EC%A6%88) (Enterprise JavaBean; EJB) 2.0 |
| [30](http://www.jcp.org/en/jsr/detail?id=30) | [Connected Limited Device Configuration](https://ko.wikipedia.org/w/index.php?title=Connected_Limited_Device_Configuration&action=edit&redlink=1) (CLDC) 1.0 for [Java ME](https://ko.wikipedia.org/wiki/Java_ME) |
| [31](http://www.jcp.org/en/jsr/detail?id=31) | [Java Architecture for XML Binding](https://ko.wikipedia.org/wiki/Java_Architecture_for_XML_Binding) (JAXB) 1.0 |
| [32](http://www.jcp.org/en/jsr/detail?id=32) | [JAIN SIP API Specification](https://ko.wikipedia.org/w/index.php?title=Session_Initiation_Protocol_(Java)&action=edit&redlink=1) (JSIP) 1.0, 1.1 and 1.2 for Java ME |
| [36](http://www.jcp.org/en/jsr/detail?id=36) | [Connected Device Configuration](https://ko.wikipedia.org/w/index.php?title=Connected_Device_Configuration&action=edit&redlink=1) (CDC) 1.0 for Java ME |
| [37](http://www.jcp.org/en/jsr/detail?id=37) | [Mobile Information Device Profile](https://ko.wikipedia.org/w/index.php?title=Mobile_Information_Device_Profile&action=edit&redlink=1) (MIDP) 1.0 for Java ME |
| [40](http://www.jcp.org/en/jsr/detail?id=40) | [Java Metadata Interface](https://ko.wikipedia.org/w/index.php?title=Java_Metadata_Interface&action=edit&redlink=1) (JMI) 1.0 |
| [41](http://www.jcp.org/en/jsr/detail?id=41) | A Simple [Assertion Facility](https://ko.wikipedia.org/w/index.php?title=Assertion_(computing)&action=edit&redlink=1) (as of J2SE 1.4) |
| [47](http://www.jcp.org/en/jsr/detail?id=47) | [Logging](https://ko.wikipedia.org/w/index.php?title=Data_logging&action=edit&redlink=1) API Specification (as of J2SE 1.4) |
| [48](http://www.jcp.org/en/jsr/detail?id=48) | [WBEM Services Specification](https://ko.wikipedia.org/w/index.php?title=WBEM_Services_Specification&action=edit&redlink=1) (as of J2SE 1.4) |
| [51](http://www.jcp.org/en/jsr/detail?id=51) | [New I/O APIs for the Java Platform](https://ko.wikipedia.org/w/index.php?title=New_I/O&action=edit&redlink=1) (NIO) (as of J2SE 1.4) |
| [52](http://www.jcp.org/en/jsr/detail?id=52) | [JavaServer Pages Standard Tag Library](https://ko.wikipedia.org/wiki/JavaServer_Pages_Standard_Tag_Library) (JSTL) 1.0 and 1.1[[2]](https://ko.wikipedia.org/wiki/%EC%9E%90%EB%B0%94_%EC%BB%A4%EB%AE%A4%EB%8B%88%ED%8B%B0_%ED%94%84%EB%A1%9C%EC%84%B8%EC%8A%A4#cite_note-2) |
| [53](http://www.jcp.org/en/jsr/detail?id=53) | [Java Servlet](https://ko.wikipedia.org/wiki/Java_Servlet) 2.3 and [JavaServer Pages](https://ko.wikipedia.org/wiki/JavaServer_Pages) (JSP) 1.2 Specifications |
| [54](http://www.jcp.org/en/jsr/detail?id=54) | [Java Database Connectivity](https://ko.wikipedia.org/wiki/Java_Database_Connectivity) (JDBC) 3.0 |
| [56](http://www.jcp.org/en/jsr/detail?id=56) | [Java Network Launching Protocol](https://ko.wikipedia.org/w/index.php?title=Java_Network_Launching_Protocol&action=edit&redlink=1) and API (JNLP) 1.0, 1.5 and 6.0[[3]](https://ko.wikipedia.org/wiki/%EC%9E%90%EB%B0%94_%EC%BB%A4%EB%AE%A4%EB%8B%88%ED%8B%B0_%ED%94%84%EB%A1%9C%EC%84%B8%EC%8A%A4#cite_note-3) ([Java Web Start](https://ko.wikipedia.org/w/index.php?title=Java_Web_Start&action=edit&redlink=1)) |
| [58](http://www.jcp.org/en/jsr/detail?id=58) | [Java 2 Platform, Enterprise Edition](https://ko.wikipedia.org/wiki/Java_2_Platform,_Enterprise_Edition) (J2EE) 1.3 |
| [59](http://www.jcp.org/en/jsr/detail?id=59) | [Java 2 Platform, Standard Edition](https://ko.wikipedia.org/wiki/Java_2_Platform,_Standard_Edition) (J2SE) 1.4 (Merlin) |
| [63](http://www.jcp.org/en/jsr/detail?id=63) | [Java API for XML Processing](https://ko.wikipedia.org/wiki/Java_API_for_XML_Processing) (JAXP) 1.1 and 1.2[[4]](https://ko.wikipedia.org/wiki/%EC%9E%90%EB%B0%94_%EC%BB%A4%EB%AE%A4%EB%8B%88%ED%8B%B0_%ED%94%84%EB%A1%9C%EC%84%B8%EC%8A%A4#cite_note-4) |
| [68](http://www.jcp.org/en/jsr/detail?id=68) | [Java Platform, Micro Edition](https://ko.wikipedia.org/wiki/Java_Platform,_Micro_Edition) (Java ME) 1.0 |
| [73](http://www.jcp.org/en/jsr/detail?id=73) | [Java Data Mining](https://ko.wikipedia.org/w/index.php?title=Java_Data_Mining&action=edit&redlink=1) API (JDM) 1.0 |
| [75](http://www.jcp.org/en/jsr/detail?id=75) | [PDA Optional Packages for the J2ME Platform](https://ko.wikipedia.org/w/index.php?title=PDA_Optional_Packages_for_the_J2ME_Platform&action=edit&redlink=1) |
| [80](http://www.jcp.org/en/jsr/detail?id=80) | Java [USB](https://ko.wikipedia.org/w/index.php?title=Universal_Serial_Bus&action=edit&redlink=1) API |
| [82](http://www.jcp.org/en/jsr/detail?id=82) | [Java APIs for Bluetooth](https://ko.wikipedia.org/w/index.php?title=Java_APIs_for_Bluetooth&action=edit&redlink=1) |
| [93](http://www.jcp.org/en/jsr/detail?id=93) | [Java API for XML Registries](https://ko.wikipedia.org/wiki/Java_API_for_XML_Registries) (JAXR) 1.0 |
| [94](http://www.jcp.org/en/jsr/detail?id=94) | Java [Rule Engine](https://ko.wikipedia.org/w/index.php?title=Rule_engine&action=edit&redlink=1) API |
| [102](http://www.jcp.org/en/jsr/detail?id=102) | [Java Document Object Model](https://ko.wikipedia.org/w/index.php?title=Java_Document_Object_Model&action=edit&redlink=1) (JDOM) 1.0 |
| [110](http://www.jcp.org/en/jsr/detail?id=110) | Java APIs for [WSDL](https://ko.wikipedia.org/wiki/Web_Services_Description_Language) (WSDL4J) 1.0 |
| [112](http://www.jcp.org/en/jsr/detail?id=112) | [Java EE Connector Architecture](https://ko.wikipedia.org/wiki/Java_EE_Connector_Architecture) (JCA) 1.5 |
| [114](http://www.jcp.org/en/jsr/detail?id=114) | [Java Database Connectivity](https://ko.wikipedia.org/wiki/Java_Database_Connectivity) (JDBC) Rowset Implementations |
| [116](http://www.jcp.org/en/jsr/detail?id=116) | [SIP Servlet API](https://ko.wikipedia.org/w/index.php?title=Session_Initiation_Protocol_(Java)&action=edit&redlink=1) 1.0 |
| [118](http://www.jcp.org/en/jsr/detail?id=118) | [Mobile Information Device Profile](https://ko.wikipedia.org/w/index.php?title=Mobile_Information_Device_Profile&action=edit&redlink=1) (MIDP) 2.0 for Java ME |
| [120](http://www.jcp.org/en/jsr/detail?id=120) | [Wireless Messaging API](https://ko.wikipedia.org/w/index.php?title=Wireless_Messaging_API&action=edit&redlink=1) (WMA) |
| [121](http://www.jcp.org/en/jsr/detail?id=121) | [Application Isolation API](https://ko.wikipedia.org/w/index.php?title=Application_Isolation_API&action=edit&redlink=1) |
| [127](http://www.jcp.org/en/jsr/detail?id=127) | [JavaServer Faces](https://ko.wikipedia.org/wiki/JavaServer_Faces) (JSF) 1.0 and 1.1[[5]](https://ko.wikipedia.org/wiki/%EC%9E%90%EB%B0%94_%EC%BB%A4%EB%AE%A4%EB%8B%88%ED%8B%B0_%ED%94%84%EB%A1%9C%EC%84%B8%EC%8A%A4#cite_note-5) |
| [133](http://www.jcp.org/en/jsr/detail?id=133) | [Java Memory Model](https://ko.wikipedia.org/w/index.php?title=Java_Memory_Model&action=edit&redlink=1) and Thread Specification Revision |
| [135](http://www.jcp.org/en/jsr/detail?id=135) | [Java Mobile Media API](https://ko.wikipedia.org/w/index.php?title=Java_Mobile_Media_API&action=edit&redlink=1) (MMAPI) for Java ME |
| [139](http://www.jcp.org/en/jsr/detail?id=139) | [Connected Limited Device Configuration](https://ko.wikipedia.org/w/index.php?title=Connected_Limited_Device_Configuration&action=edit&redlink=1) (CLDC) 1.1 for Java ME |
| [140](http://www.jcp.org/en/jsr/detail?id=140) | [Service Location Protocol](https://ko.wikipedia.org/w/index.php?title=Service_Location_Protocol&action=edit&redlink=1) (SLP) API for Java |
| [141](http://www.jcp.org/en/jsr/detail?id=141) | [Session Description Protocol](https://ko.wikipedia.org/wiki/Session_Description_Protocol) (SDP) API for Java |
| [151](http://www.jcp.org/en/jsr/detail?id=151) | [Java 2 Platform, Enterprise Edition](https://ko.wikipedia.org/wiki/Java_2_Platform,_Enterprise_Edition) (J2EE) 1.4 |
| [152](http://www.jcp.org/en/jsr/detail?id=152) | [JavaServer Pages](https://ko.wikipedia.org/wiki/JavaServer_Pages) (JSP) 2.0 |
| [153](http://www.jcp.org/en/jsr/detail?id=153) | [Enterprise JavaBeans](https://ko.wikipedia.org/w/index.php?title=Enterprise_JavaBean&action=edit&redlink=1) (EJB) 2.1 |
| [154](http://www.jcp.org/en/jsr/detail?id=154) | [Java Servlet](https://ko.wikipedia.org/wiki/Java_Servlet) 2.4 and 2.5 Specifications[[6]](https://ko.wikipedia.org/wiki/%EC%9E%90%EB%B0%94_%EC%BB%A4%EB%AE%A4%EB%8B%88%ED%8B%B0_%ED%94%84%EB%A1%9C%EC%84%B8%EC%8A%A4#cite_note-6) |
| [160](http://www.jcp.org/en/jsr/detail?id=160) | [Java Management Extensions](https://ko.wikipedia.org/wiki/Java_Management_Extensions) (JMX) Remote API 1.0 |
| [166](http://www.jcp.org/en/jsr/detail?id=166) | [Concurrency](https://ko.wikipedia.org/w/index.php?title=Java_concurrency&action=edit&redlink=1) Utilities (as of J2SE 5.0 [틀:Javadoc:SE](https://ko.wikipedia.org/w/index.php?title=%ED%8B%80:Javadoc:SE&action=edit&redlink=1), [틀:Javadoc:SE](https://ko.wikipedia.org/w/index.php?title=%ED%8B%80:Javadoc:SE&action=edit&redlink=1) and [틀:Javadoc:SE](https://ko.wikipedia.org/w/index.php?title=%ED%8B%80:Javadoc:SE&action=edit&redlink=1)) |
| [168](http://www.jcp.org/en/jsr/detail?id=168) | [Java Portlet specification](https://ko.wikipedia.org/w/index.php?title=Java_Portlet_specification&action=edit&redlink=1) 1.0 |
| [170](http://www.jcp.org/en/jsr/detail?id=170) | [Content repository API for Java](https://ko.wikipedia.org/w/index.php?title=Content_repository_API_for_Java&action=edit&redlink=1) (JCR) 1.0 |
| [172](http://jcp.org/en/jsr/detail?id=172) | Web Services Specification for Java ME |
| [173](http://www.jcp.org/en/jsr/detail?id=173) | [StAX](https://ko.wikipedia.org/wiki/StAX) (Streaming API for XML) |
| [175](http://www.jcp.org/en/jsr/detail?id=175) | [A Metadata Facility for the Java Programming Language](https://ko.wikipedia.org/w/index.php?title=A_Metadata_Facility_for_the_Java_Programming_Language&action=edit&redlink=1) |
| [176](http://www.jcp.org/en/jsr/detail?id=176) | [Java 2 Platform, Standard Edition](https://ko.wikipedia.org/wiki/Java_2_Platform,_Standard_Edition) (J2SE) 5.0 (Tiger) |
| [177](http://www.jcp.org/en/jsr/detail?id=177) | [Security and Trust Services API for J2ME](https://ko.wikipedia.org/w/index.php?title=Security_and_Trust_Services_API_for_J2ME&action=edit&redlink=1) (SATSA) |
| [179](http://www.jcp.org/en/jsr/detail?id=179) | [Location API](https://ko.wikipedia.org/w/index.php?title=Location_API_for_Java_ME&action=edit&redlink=1) 1.0 for Java ME |
| [180](http://www.jcp.org/en/jsr/detail?id=180) | [Session Initiation Protocol (SIP) API for Java ME](https://ko.wikipedia.org/w/index.php?title=Session_Initiation_Protocol_(Java)&action=edit&redlink=1) |
| [181](http://www.jcp.org/en/jsr/detail?id=181) | [Web Services](https://ko.wikipedia.org/w/index.php?title=Web_Services&action=edit&redlink=1) Metadata for the Java Platform |
| [184](http://www.jcp.org/en/jsr/detail?id=184) | [Mobile 3D Graphics API](https://ko.wikipedia.org/w/index.php?title=Mobile_3D_Graphics_API&action=edit&redlink=1) for Java ME 1.0 and 1.1 |
| [185](http://www.jcp.org/en/jsr/detail?id=185) | [Java Technology for the Wireless Industry](https://ko.wikipedia.org/w/index.php?title=Java_Technology_for_the_Wireless_Industry&action=edit&redlink=1) (JTWI) |
| [187](http://www.jcp.org/en/jsr/detail?id=187) | [Instant messaging](https://ko.wikipedia.org/w/index.php?title=Instant_messaging&action=edit&redlink=1) ([Java ME](https://ko.wikipedia.org/wiki/Java_ME) and [Java SE](https://ko.wikipedia.org/wiki/Java_SE)) |
| [198](http://www.jcp.org/en/jsr/detail?id=198) | A Standard Extension API for [Integrated Development Environments](https://ko.wikipedia.org/w/index.php?title=Integrated_Development_Environment&action=edit&redlink=1) |
| [199](http://www.jcp.org/en/jsr/detail?id=199) | [Java Compiler](https://ko.wikipedia.org/w/index.php?title=Java_compiler&action=edit&redlink=1) API |
| [201](http://www.jcp.org/en/jsr/detail?id=201) | Extending the [Java Programming Language](https://ko.wikipedia.org/w/index.php?title=Java_(programming_language)&action=edit&redlink=1) with Enumerations, Autoboxing, Enhanced for loops and Static Import (as of J2SE 5.0) |
| [202](http://www.jcp.org/en/jsr/detail?id=202) | [Java Class File](https://ko.wikipedia.org/w/index.php?title=Class_(file_format)&action=edit&redlink=1) Specification Update |
| [203](http://www.jcp.org/en/jsr/detail?id=203) | More [New I/O APIs for the Java Platform](https://ko.wikipedia.org/w/index.php?title=New_I/O&action=edit&redlink=1) (NIO2) |
| [204](http://www.jcp.org/en/jsr/detail?id=204) | Unicode Supplementary Character Support (as of J2SE 5.0 adds support for [Unicode](https://ko.wikipedia.org/wiki/Unicode) 3.1) |
| [205](http://www.jcp.org/en/jsr/detail?id=205) | [Wireless Messaging API 2.0](https://ko.wikipedia.org/w/index.php?title=Wireless_Messaging_API_2.0&action=edit&redlink=1) (WMA) 2.0 |
| [206](http://www.jcp.org/en/jsr/detail?id=206) | [Java API for XML Processing](https://ko.wikipedia.org/wiki/Java_API_for_XML_Processing) (JAXP) 1.3 |
| [208](http://www.jcp.org/en/jsr/detail?id=208) | [Java Business Integration](https://ko.wikipedia.org/w/index.php?title=Java_Business_Integration&action=edit&redlink=1) (JBI) 1.0 |
| [215](http://www.jcp.org/en/jsr/detail?id=215) | Java Community Process (JCP) 2.6 |
| [218](http://www.jcp.org/en/jsr/detail?id=218) | [Connected Device Configuration](https://ko.wikipedia.org/w/index.php?title=Connected_Device_Configuration&action=edit&redlink=1) (CDC) 1.1 for Java ME |
| [220](http://www.jcp.org/en/jsr/detail?id=220) | [Enterprise JavaBeans](https://ko.wikipedia.org/w/index.php?title=Enterprise_JavaBean&action=edit&redlink=1) (EJB) 3.0 |
| [221](http://www.jcp.org/en/jsr/detail?id=221) | [Java Database Connectivity](https://ko.wikipedia.org/wiki/Java_Database_Connectivity) (JDBC) 4.0 |
| [222](http://www.jcp.org/en/jsr/detail?id=222) | [Java Architecture for XML Binding](https://ko.wikipedia.org/wiki/Java_Architecture_for_XML_Binding) (JAXB) 2.0 |
| [223](http://www.jcp.org/en/jsr/detail?id=223) | [Scripting for the Java Platform](https://ko.wikipedia.org/w/index.php?title=Scripting_for_the_Java_Platform&action=edit&redlink=1) for Java SE 6 |
| [224](http://www.jcp.org/en/jsr/detail?id=224) | [Java API for XML Web Services](https://ko.wikipedia.org/wiki/Java_API_for_XML_Web_Services) (JAX-WS), successor of [JAX-RPC](https://ko.wikipedia.org/wiki/JAX-RPC) |
| [225](http://www.jcp.org/en/jsr/detail?id=225) | [XQuery](https://ko.wikipedia.org/w/index.php?title=XQuery&action=edit&redlink=1) API for Java (XQJ) |
| [226](http://www.jcp.org/en/jsr/detail?id=226) | [Scalable 2D Vector Graphics](https://ko.wikipedia.org/wiki/SVG) API for [J2ME](https://ko.wikipedia.org/wiki/J2ME) |
| [231](http://www.jcp.org/en/jsr/detail?id=231) | [Java Bindings for OpenGL](https://ko.wikipedia.org/w/index.php?title=Java_Bindings_for_OpenGL&action=edit&redlink=1) |
| [234](http://www.jcp.org/en/jsr/detail?id=234) | [Advanced Multimedia Supplements](https://ko.wikipedia.org/w/index.php?title=Advanced_Multimedia_Supplements&action=edit&redlink=1) API for Java ME |
| [235](http://www.jcp.org/en/jsr/detail?id=235) | [Service Data Objects](https://ko.wikipedia.org/w/index.php?title=Service_Data_Objects&action=edit&redlink=1) (SDO) |
| [239](http://www.jcp.org/en/jsr/detail?id=239) | Java Bindings for [OpenGL](https://ko.wikipedia.org/wiki/OpenGL) ES |
| [241](http://www.jcp.org/en/jsr/detail?id=241) | [The Groovy Programming Language](https://ko.wikipedia.org/w/index.php?title=Groovy_(programming_language)&action=edit&redlink=1) |
| [243](http://www.jcp.org/en/jsr/detail?id=243) | [Java Data Objects](https://ko.wikipedia.org/w/index.php?title=Java_Data_Objects&action=edit&redlink=1) (JDO) 2.0 |
| [244](http://www.jcp.org/en/jsr/detail?id=244) | [Java Platform, Enterprise Edition](https://ko.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition) (Java EE) 5 |
| [245](http://www.jcp.org/en/jsr/detail?id=245) | [JavaServer Pages](https://ko.wikipedia.org/wiki/JavaServer_Pages) (JSP) 2.1 |
| [247](http://www.jcp.org/en/jsr/detail?id=247) | [Java Data Mining](https://ko.wikipedia.org/w/index.php?title=Java_Data_Mining&action=edit&redlink=1) API (JDM) 2.0 |
| [248](http://www.jcp.org/en/jsr/detail?id=248) | Mobile Service Architecture |
| [249](http://www.jcp.org/en/jsr/detail?id=249) | Mobile Service Architecture Advanced |
| [250](http://www.jcp.org/en/jsr/detail?id=250) | Common [Annotations](https://ko.wikipedia.org/w/index.php?title=Java_annotation&action=edit&redlink=1) for the Java Platform (for the [Metadata facility for Java](https://ko.wikipedia.org/w/index.php?title=Metadata_facility_for_Java&action=edit&redlink=1)) |
| [252](http://www.jcp.org/en/jsr/detail?id=252) | [JavaServer Faces](https://ko.wikipedia.org/wiki/JavaServer_Faces) (JSF) 1.2 |
| [253](http://www.jcp.org/en/jsr/detail?id=253) | [Mobile Telephony API](https://ko.wikipedia.org/w/index.php?title=Mobile_Telephony_API&action=edit&redlink=1) (MTA) |
| [255](http://www.jcp.org/en/jsr/detail?id=255) | [Java Management Extensions](https://ko.wikipedia.org/wiki/Java_Management_Extensions) (JMX) 2.0 |
| [260](http://www.jcp.org/en/jsr/detail?id=260) | [Javadoc](https://ko.wikipedia.org/w/index.php?title=Javadoc&action=edit&redlink=1) Tag Technology Update |
| [269](http://www.jcp.org/en/jsr/detail?id=269) | Pluggable [Annotations](https://ko.wikipedia.org/w/index.php?title=Java_annotation&action=edit&redlink=1) Processing API (for the [Metadata facility for Java](https://ko.wikipedia.org/w/index.php?title=Metadata_facility_for_Java&action=edit&redlink=1)) |
| [270](http://www.jcp.org/en/jsr/detail?id=270) | [Java Platform, Standard Edition](https://ko.wikipedia.org/wiki/Java_Platform,_Standard_Edition) (Java SE) 6 (Mustang) |
| [271](http://www.jcp.org/en/jsr/detail?id=271) | [Mobile Information Device Profile](https://ko.wikipedia.org/w/index.php?title=Mobile_Information_Device_Profile&action=edit&redlink=1) (MIDP) 3.0 for Java ME |
| [274](http://www.jcp.org/en/jsr/detail?id=274) | The [BeanShell](https://ko.wikipedia.org/w/index.php?title=BeanShell&action=edit&redlink=1) Scripting Language |
| [275](http://www.jcp.org/en/jsr/detail?id=275) | Units specification (see [JScience](https://ko.wikipedia.org/w/index.php?title=JScience&action=edit&redlink=1)) |
| [276](http://www.jcp.org/en/jsr/detail?id=276) | Design-time [Metadata](https://ko.wikipedia.org/w/index.php?title=Metadata&action=edit&redlink=1) for [JavaServer Faces](https://ko.wikipedia.org/wiki/JavaServer_Faces) Components |
| [277](http://www.jcp.org/en/jsr/detail?id=277) | [Java Module System](https://ko.wikipedia.org/w/index.php?title=Java_Module_System&action=edit&redlink=1) |
| [281](http://www.jcp.org/en/jsr/detail?id=281) | [IMS Services API](https://ko.wikipedia.org/w/index.php?title=IMS_Services_API&action=edit&redlink=1) (See [IMS](https://ko.wikipedia.org/wiki/IP_Multimedia_Subsystem)) |
| [282](http://www.jcp.org/en/jsr/detail?id=282) | [Real-Time Specification for Java](https://ko.wikipedia.org/w/index.php?title=Real-Time_Specification_for_Java&action=edit&redlink=1) (RTSJ) 1.1 |
| [289](http://www.jcp.org/en/jsr/detail?id=289) | [SIP Servlet API](https://ko.wikipedia.org/w/index.php?title=Session_Initiation_Protocol_(Java)&action=edit&redlink=1) 1.1 |
| [291](http://www.jcp.org/en/jsr/detail?id=291) | Dynamic Component Support for Java SE (see [OSGi](https://ko.wikipedia.org/wiki/OSGi)) |
| [292](http://www.jcp.org/en/jsr/detail?id=292) | Supporting Dynamically Typed Languages on the JavaTM Platform (see also [Da Vinci Machine](https://ko.wikipedia.org/w/index.php?title=Da_Vinci_Machine&action=edit&redlink=1)) |
| [293](http://www.jcp.org/en/jsr/detail?id=293) | [Location API](https://ko.wikipedia.org/w/index.php?title=Location_API_for_Java_ME&action=edit&redlink=1) 2.0 for Java ME |
| [294](http://www.jcp.org/en/jsr/detail?id=294) | Improved Modularity Support in the Java Programming Language |
| [296](http://www.jcp.org/en/jsr/detail?id=296) | [Swing Application Framework](https://ko.wikipedia.org/w/index.php?title=Swing_Application_Framework&action=edit&redlink=1) (Java SE 7) |
| [301](http://www.jcp.org/en/jsr/detail?id=301) | JSF Portlet Bridge |
| [307](http://www.jcp.org/en/jsr/detail?id=307) | [Network Mobility and Mobile Data API](https://ko.wikipedia.org/w/index.php?title=Network_Mobility_and_Mobile_Data_API&action=edit&redlink=1) (not official as of July, 20th, 2007 but official release is planned for 2. Q. 2008 |
| [308](http://www.jcp.org/en/jsr/detail?id=308) | [Annotations](https://ko.wikipedia.org/w/index.php?title=Java_annotation&action=edit&redlink=1) on Java Types (Java SE 7) |
| [314](http://www.jcp.org/en/jsr/detail?id=314) | [JavaServer Faces](https://ko.wikipedia.org/wiki/JavaServer_Faces) (JSF) 2.0 |
| [316](http://www.jcp.org/en/jsr/detail?id=316) | [Java Platform, Enterprise Edition](https://ko.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition) (Java EE) 6 |
|  | |
| [901](http://www.jcp.org/en/jsr/detail?id=901) | [The Java Language](https://ko.wikipedia.org/wiki/%EC%9E%90%EB%B0%94_(%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%B0%8D_%EC%96%B8%EC%96%B4)) Specification, Third Edition (JLS) (as of J2SE 5.0 incorporates changes from JSRs 14, 41, 133, 175, 201, and 204) |
| [907](http://www.jcp.org/en/jsr/detail?id=907) | [Java Transaction API](https://ko.wikipedia.org/wiki/Java_Transaction_API) (JTA) 1.0 and 1.1 |
| [912](http://www.jcp.org/en/jsr/detail?id=912) | [Java 3D](https://ko.wikipedia.org/w/index.php?title=Java_3D&action=edit&redlink=1) API 1.3 |
| [913](http://www.jcp.org/en/jsr/detail?id=913) | Java Community Process (JCP) 2.0, 2.1 & 2.5[[7]](https://ko.wikipedia.org/wiki/%EC%9E%90%EB%B0%94_%EC%BB%A4%EB%AE%A4%EB%8B%88%ED%8B%B0_%ED%94%84%EB%A1%9C%EC%84%B8%EC%8A%A4#cite_note-7) |
| [914](http://www.jcp.org/en/jsr/detail?id=914) | [Java Message Service](https://ko.wikipedia.org/wiki/Java_Message_Service) (JMS) API 1.0 and 1.1 |
| [924](http://www.jcp.org/en/jsr/detail?id=924) | [Java Virtual Machine](https://ko.wikipedia.org/wiki/Java_Virtual_Machine) Specification, Second Edition (JVM) (as of J2SE 5.0)[[8]](https://ko.wikipedia.org/wiki/%EC%9E%90%EB%B0%94_%EC%BB%A4%EB%AE%A4%EB%8B%88%ED%8B%B0_%ED%94%84%EB%A1%9C%EC%84%B8%EC%8A%A4#cite_note-8) |
| [926](http://www.jcp.org/en/jsr/detail?id=926) | [Java 3D](https://ko.wikipedia.org/w/index.php?title=Java_3D&action=edit&redlink=1) API 1.4 |

## 각주[[편집](https://ko.wikipedia.org/w/index.php?title=%EC%9E%90%EB%B0%94_%EC%BB%A4%EB%AE%A4%EB%8B%88%ED%8B%B0_%ED%94%84%EB%A1%9C%EC%84%B8%EC%8A%A4&action=edit&section=2)]

1. [이동↑](https://ko.wikipedia.org/wiki/%EC%9E%90%EB%B0%94_%EC%BB%A4%EB%AE%A4%EB%8B%88%ED%8B%B0_%ED%94%84%EB%A1%9C%EC%84%B8%EC%8A%A4#cite_ref-1)