Spring 3.0 -> 3.1 -> 3.2 따라잡기

2013-07-13 제5회 hello world 오픈 세미나

정상혁

발표자 소개

- NHN Business Platform 웹플랫폼개발랩
- Java 프레임워크 개발, 기술지원, 교육
- Tech@NHN 시리즈 공저자, Helloworld 기고
- Java, TDD, Spring, Android

얻을 수 있는 것

- Spring 쓰시는 분
  - 실무에서 이슈가 되었던 부분(보안/성능/하위 호환성)
  - 버전업 체크리스트
  - Spring의 일부 코드를 리뷰
- Spring 안 쓰시는 분
  - 공통 모듈/ 프레임워크 개선 사례
  - 인터페이스, 클래스 설계 원칙

기대하지 말아야할 것

- Spring 3.1, 3.2의 신규 기능의 자세한 소개
  - 참고자료로 대신 제공

버전을 올리면서 조심할 부분

XSD 스키마 버전

- 상위 버전의 코드를 복사해오거나 jar버전을 올렸다내렸다 할 때 주의

```
<beans
  xmlns="http://www.springframework.org/schema/beans"
  xsi:schemaLocation=
    "http://www.springframework.org/schema/beans
     http://www.springframework.org/schema/beans/spring-beans-3.1.xsd">
```

- jar에 없는 상위버전의 파일은 HTTP로 읽어옴 - 불필요한 성능 저하

XSD 스키마 버전

- 서버에서 http://www.springframework.org/schema/beans/에 접근을 못한다면?

```
nested exception is org.xml.sax.SAXParseException:
    cvc-elt.1: Cannot find the declaration of element 'beans'.
```

- spring-beans.xsd 는 최신 버전으로 연결
  - jar안의 폴더에는 spring-beans.xsd가 없지만 spring.schemas의 선언에 의해 연결
  - http://www.springframework.org/schema/beans/spring-beans.xsd는 최신 버전

Dependency 변화

- Optional dependency
  - spring-tx -> spring-core 등
  - 직접 선언하지 않으면 ClassNotFoundException
- spring-asm 모듈이 spring-core로 통합
  - [SPR-9669] Upgrade to ASM 4.0 and CGLIB 3.0
  - 별도의 dependency 선언이 필요 없어졌음.

@MVC의 기반 클래스 변경

- 3.1 부터 MVC기반 클래스 변경
  - `<mvc:annotation-driven/>` 에 의해 등록되는 클래스들
  - HandlerMapping : DefaultAnnotationHandlerMapping -> RequestMappingHandlerMapping
  - HandlerAdaptor : AnnotationMethodHandlerAdapter -> RequestMappingHandlerAdapter
  - HandlerExceptionResolver : AnnotationMethodHandlerExceptionResolver -> ExceptionHandlerExceptionResolver

@MVC의 기반 클래스 변경

- 이름을 보면
  - 'Annotation’은 이제 당연하므로 생략.
  - @RequestMapping이 연상되도록
- Handler를 클래스가 단위가 아닌 메소드 단위로

@MVC의 기반 클래스 변경

- `<mvc:annotation-driven/>` 를 쓸 때 옛날 클래스와 같이 쓰지 않도록 주의

```
<mvc:annotation-driven/>
<bean id="handlerAdapter"
    class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter">
    <property name="customArgumentResolvers">
       <array>
             <bean class="net.benelog.UserArgumentResolver"/>
       </array>
    </property><property name="order" value="-1"/>
</bean>
```

@MVC의 기반 클래스 변경

- 아래와 같이 바꿔야 함

```
<mvc:annotation-driven>
    <mvc:argument-resolvers>
            <bean class="net.benelog.UserArgumentResolver"/>
    </mvc:argument-resolvers>
</mvc:annotation-driven>
```

ArgumentResolver

- 3.1이후 WebArgumentResolver -> MethodArgumentResolver
- WebArgumentResolver

```
public interface WebArgumentResolver {
    Object UNRESOLVED = new Object();
    Object resolveArgument(MethodParameter methodParameter, NativeWebRequest webRequest) throws Exception;
}
```

ArgumentResolver

- MethodArgumentResolver

```
public interface HandlerMethodArgumentResolver {
    boolean supportsParameter(MethodParameter parameter);
    Object resolveArgument(MethodParameter parameter,
                           ModelAndViewContainer mavContainer,
                           NativeWebRequest webRequest,
                           WebDataBinderFactory binderFactory) throws Exception;
}
```

ArgumentResolver

- 파라미터 지원여부와 해석의 역할을 분리
  - 파라미터 지원여부는 MethodParameter 만으로 판단
- supportsParameter(…) 메서드의 결과는 캐쉬 (HandlerMethodArgumentResolverComposite)
- ServletWebArgumentResolverAdapter,AbstractWebArgumentResolverAdapter로 WebArgumentResolver -> MethodArgumentResolver 변환
  - 최초 호출시에 WebArgumentResolver.resolveArgument 가 2번 호출됨
  - WebArguementResolver에서의 Exception을 무시

ArgumentResolver

```
@Override
public boolean supportsParameter(MethodParameter parameter) {
    try {
        NativeWebRequest webRequest = getWebRequest();
        Object result = this.adaptee.resolveArgument(parameter, webRequest);
         if (result == WebArgumentResolver.UNRESOLVED) {
              return false;
         }
         else {
             return ClassUtils.isAssignableValue(parameter.getParameterType(), result);
        }
    }   catch (Exception ex) {
         // ignore (see class-level doc)
         logger.debug("Error in checking support for parameter [" + parameter + "], message: " + ex.getMessage());
        return false;
    }
}
```

ArgumentResolver

- HandlerMethodArgumentResolver로 반드시 재구현해야 하는 경우
  - resolveArgument(…) 메소드가 NativeWebRequest 안에 들어간 값에 따라서 UNRESOLVED가 반환될 수 있는 경우.
  - resolveArgument(…) 메소드가 Exception을 던질 수 있는 가능성이 있을 때
- 문제가 없는 경우라도 되도록 재구현 권장
  - cache되는 영역은 반복호출되지 않아서 성능 이득
  - 역할 분담으로 코드 가독성 향상

HandlerInterceptor

- interface는 그대로

```
public boolean preHandle(HttpServletRequest request, HttpServletResponse response,
    Object handler) throws Exception {

....

}
```

- 3.1부터 3번째 파라미터인 hander의 type이 HandlerMethod로 바뀜
  - 메서드에 대한 정보를 제공 : getReturnType(), getMethodParameters()

대표적 Deprecated

- 3.0 -> 3.1
  - SimpleJdbcTemplate -> JdbcTemplate, NamedParameterJdbcTemplate
  - SimpleJdbcTestUtils -> JdbcTestUtils
- 3.1 -> 3.2
  - Spring 3.0까지 사용되는 MVC 기반 클래스 : AnnotationMethodHandlerAdapter 등
  - iBatis 지원클래스 : SqlMapClientTemplate, SqlMapClientDaoSupport
- Spring LOC : http://www.ohloh.net/p/spring
  - 갑자기 코드가 내려간 시점이 3.0출시로 추정
  - 4.0에서도 많은 코드가 없어질 가능성에 대비해야함

개선 지점 심층 분석

ViewResolver의 Cache

- "redirect:form.html?entityId=3" 유형의 OOM 가능성
  - Controller에서 String으로 return 혹은 ModelAndView.setViewName(…)으로 지정하면 AbstractCachingViewResolver에서 View를 해석후 cache.
  - view name에 id등이 포함되면 계속 새로운 View가 생성
  - 2012년 12월 03일 : 이슈 올라옴 (SPR-10065)
  - 2012년 12월 11일 : Commit by Juergen Hoeller
    - 오래된 View를 지우는 구현이 추가.
    - LinkedHashMap.removeEldestEntry() override해서 활용
    - 2012년 12월 13일 : 3.2 GA 버전에 포함되어 Release
    - 2013년 01월 23일 : 3.1.4 버전에도 포함되어 Release

ViewResolver의 Cache

- 3.0 이전의 해결책
  - View를 return
- 3.1의 이후 해결책
  - RequestAttribute

```
@RequestMapping(method = RequestMethod.POST)
public String onPost(RedirectAttributes attrs) {
    ...
    attrs.addAttribute(entityId, 123);
    return "redirect:form.html;   // resulting URL has entityId=123
}
```

ViewResolver의 Cache

- 3.1의 이후 해결책
  - URI Template

```
return "redirect:form.html?entityId={entityId}";
```

ViewResolver의 Cache

- 3.2.GA, 3.1.4이후의 OOM 방어 코드
  - 그래도 Cache효율성을 고려해서 URI template등 활용이 바람직.
- 3.2.2 이후의 성능 최적화
  - SPR-3145 (2006/12/26) : Performance improvement on AbstractCachingViewResolver
    - Cache를 HashMap + synchronized block 대신 ConcurrentHashMap으로 바꾸자는 의견이 있었으나 JDK5지원으로 하지 못했었음.
  - Commit by Juergen Hoeller
    - LinkedHashMap과 ConcurrentHashMap을 같이 사용.
    - OOM방어에 활용한 LinkedHashMap.removeEldestEntry()이 ConcurrentHashMap에는 없기 때문.

EL injection 방어

- 취약점의 조건 (AND조건)
  - EL 2.2를 지원하는 서블릿 컨테이너를 쓰거나 EL 2.2 라이브러리를 직접 jar파일로 참조해서 쓰고 있다. (대표적으로 Tomcat 7.x혹은 Glassfish 2.2.x)
  - Spring 3.1.x 미만 버전을 쓰고 있다.
  - Spring의 JSP Tag(`<spring:message..` 등)을 쓰고 있다.
  - Spring의 JSP Tag에서 EL을 지원하는 속성에 사용자가 입력한 값이 들어갈 수 있다.

EL injection 방어

- 현상 : double evaluation.

```
<spring:message scope="${param.name}"/>
```

그 페이지를 호출할 때 name=${applicationScope} 로 쓰면 아래와 같이 인식

```
<spring:message scope="${applicationScope}"/>
```

- 원인
  - JSP 2.0이하에서도 EL을 지원하기 위한 Spring의 기능 때문.
    - `${param.name}` 대신 `<c:out value=${param.name}>` 을 써야했던 시절

EL injection 방어

- 해결방안
  - Spring 3.1.x 버전 이상 사용
  - Spring 3.0.6 혹은 2.5.6.SEC03버전 이상 사용 + web.xml에 추가선언

```
<context-param>
     <description>Spring Expression Language Support</description>
     <param-name>springJspExpressionSupport</param-name>
     <param-value>false</param-value>
</context-param>
```

EL injection 방어

- 조치 코드 분석
  - 3.0.x의 ExpressionEvaluationUtils.isSpringJspExpressionSupportActive().
    - web.xml의 내용을 읽어서 옵션 판단
  - 3.1.x의 ExpressionEvaluationUtils.isSpringJspExpressionSupportActive()
    - Servlet스펙 2.4이상 일때는 Spring에서 다시 평가하지 않음.
- 자세한 내용은 https://gist.github.com/benelog/4582041

Method ArgumentResolver로 내부구조 개선

- 3.0 이전
  - Controller의 메소드 파라미터의 HttpSession, Locale, OutputStream, @PathVarible처리는 AnnotationMethodHandlerAdapter의 resolveStandardArgument(…)에서 담당. resolvePathVariable(…)에 포함
  - WebArguemntResolver는 확징 지점만의 의미
- 3.1 이후
  - 기본 파라미터 처리도 별도의 MethodArgumentResolver로 분리

Method ArgumentResolver로 내부구조 개선

- 기본 MethodArgumentResolver
  - ServletRequestMethodArgumentResolver : HttpServletRequest, HttpSession, InputStream, Reader, Locale 형의 파라미터의 해석
  - ServletResponseMethodArgumentResolver : HttpServletResponse, OutputStream, Writer 형의 파라미터 해석
  - PathVariableMapMethodArgumentResolver : @PathVarible 이 붙은 파라미터의 해석
  - RequestHeaderMethodArgumentResolver : @RequestHeader 가 붙은 파라미터의 해석
  - RequestParamMethodArgumentResolver : @RequestParam, @RequestPart 가 붙은 파라미터의 해석

당장 써야 할 신규기능

MVC Test

- API 통합테스트에 유용
- JSP 내용까지 나오지는 않음
- 테스트코드 입문자에게 추천

```
@RunWith(SpringJUnit4ClassRunner.class)
@WebAppConfiguration
@ContextHierarchy({
    @ContextConfiguration({ "/spring/applicationContext.xml" }),
    @ContextConfiguration("/spring/mvc-config.xml")
})
public class HomeMvcTest {
    @Autowired
    private WebApplicationContext wac;

    private MockMvc mvc;
```

MVC Test

```
@Before
public void setup() {
    this.mvc = webAppContextSetup(this.wac).build();
}

@Test
public void home() throws Exception {
    mvc.perform(get("/"))
        .andExpect(status().isOk())
        .andExpect(view().name("home"));
}
```

MVC Test

```
    @Test
    public void imageJson() throws Exception {
        mvc.perform(get("/viewImage/cloud.json"))
           .andExpect(status().isOk())
            .andExpect(content().string("{\"src\":\"/img/cloud.png\",\"height\":64,\"width\":64}"));
    }

    @Test
    public void containsImagePath() throws Exception {
        mvc.perform(get("/viewImage/phone.json"))
           .andExpect(status().isOk())
            .andExpect(content().string(containsString("/img/phone.png")));
    }
}
```

통합 Resource 관리

- 다양한 환경변수를 같은 방식으로
  - OS 환경변수 : System.getEnv()로 읽어오는 OS 레벨에 선언된 변수들
  - 시스템 프로퍼티 : System.getProperties()로 참조하는 값들. -D옵션으로 지정된 값, JVM 관련정보 -java.home, 등
  - JNDI
  - SerlvetContext 파라미터
  - ServletConfig 파라미터
  - 직접 지정한 파일명

통합 Resource 관리

- 명확한 에러
  - 3.0.x에서 적극적인 에러발견을 위해 SPel(#{api.url})을 쓰는 경우가 많았음.
- 똑같이 선언. Locations가 없어도 됨. ${api.url} 과 같이 사용.

```
<context:property-placeholder />
```

- context:property-placeholder의 기본 등록 클래스 변경
  - PropertyPlaceholderConfigurer -> PropertySourcesPlaceholderConfigurer

대표적 추가기능

- Spring 3.1
  - Cache absraction : @Cacheable
  - Profile : `<beans profile="dev">` , @Profile("dev")
  - 환경변수 통합 관리 : -파일, OS환경변수, JNDI 등의 다양한 위치의 환경변수를 통합해서 ${} 표현식으로 표현.
  - javaConfig 확장 : @~Enable
  - Test프레임워크에서 JavaConfig와 Profile 지원
  - c: namespace - `<bean id="employe eService" c:groupName="webpl".. />`

대표적 추가기능

- Spring 3.2
  - Servlet 3.0 기반 비동기 요청 처리
  - Spring MVC Test Framework
  - @ControllerAdvice
  - Matrix variables

시사점

환경변화에 맞춤

- JavaEE
  - EL 2.2 때문에 생긴 보안 취약점 대처
  - JavaEE 신규 스펙 지원
- Java
  - JDK 1.4를 위한 배려는 그만
    - ViewResolver 캐쉬 개선
    - SimpleJdbcTemplate deprecated

완벽한 하위 호환성은 포기한 개선 지점

- 완벽한 하위 호환성을 일부 포기
  - @MVC artitecture : WebArgumentResolver 등
- 완충지대
  - Custom namespace : `<mvc:annotation-driven/>` ,
  - bean 등록 방식으로는 이전 방식도 지원
  - Adaptor :
- 표준 스펙보다 빠른 개선 싸이클

신중한 접근

- 별도의 프로젝트로 검증 후 Spring 주류에 포함
  - Spring MVC test, Java Config

정리

업그레이드의 이득은?

- 위험성 대처 : 보안, 버그 혹은 잘못 쓰기 쉬운 기능
- 성능 개선
- 신규 기능, 더 짧은 코드
- 향후 4.0 따라가기 준비 : JDK8 지원

참고자료

Spring 3.0 -> 3.1

- 레퍼런스 메뉴얼의 3.1 신규 기능 설명
  - http://static.springsource.org/spring-framework/docs/3.1.x/spring-framework-reference/html/new-in-3.1.html
- 스프링소스 블로그 자료
  - http://blog.springsource.org/2011/02/11/spring-framework-3-1-m1-released/
  - http://blog.springsource.org/2011/02/14/spring-3-1-m1-introducing-profile/
  - http://blog.springsource.org/2011/02/15/spring-3-1-m1-unified-property-management/
  - http://blog.springsource.org/2011/06/09/spring-framework-3-1-m2-released/

Spring 3.0 -> 3.1

- 스프링소스 블로그 자료
  - http://blog.springsource.org/2011/06/10/spring-3-1-m2-configuration-enhancements/
  - http://blog.springsource.org/2011/06/21/spring-3-1-m2-testing-with-configuration-classes-and-profiles/
  - http://blog.springsource.org/2011/10/12/spring-framework-3-1-rc1-released/
  - http://blog.springsource.org/2011/12/13/spring-framework-3-1-goes-ga/
  - http://blog.springsource.org/2012/04/06/migrating-to-spring-3-1-and-hibernate-4-1/

Toby님의 Spring 3.1 분석 자료

- 스프링 3.1 (1) JavaConfig의 위험성
- 스프링 3.1 (2) HandlerInterceptor의 적용순서
- 스프링 3.1 (3) @Enable~
- 스프링 3.1 (4) Static @Bean 메소드
- 스프링 3.1 (5) @Enable*을 이용한 설정 재활용 기법 세미나 동영상 & 자료
- 스프링 3.1 (6) web.xml의 활성 프로파일 설정

Toby님의 Spring 3.1 분석 자료

- 스프링 3.1 (7) 프로퍼티 소스 추상화와 PropertySourcePlaceholderConfigurer
- 스프링 3.1 (8) web.xml 없는 스프링 개발
- 스프링 3.1 (9) 애노테이션은 상속되지 않는다?
- 스프링 3.1 (10) 심심풀이 @RequestMapping 요청 조건 퀴즈
- 스프링 3.1 (11) 심심풀이 @RequestMapping 요청 조건 퀴즈 풀이
- 토비의 Spring 3.1 VOl2의 '4.9 스프링 3.1의 @MVC (p633-669)' : @MVC의 구조가 변한 이유를 자세히 설명

Spring 3.0 -> 3.1 상세변경 이력

- Deprecated : http://static.springsource.org/spring-framework/docs/3.1.x/javadoc-api/deprecated-list.html
- 변경 패키지, 클래스 목록 : http://static.springsource.org/spring-framework/docs/3.0.6.RELEASE_to_3.1.0.BUILD-SNAPSHOT/
- Change log : http://static.springsource.org/spring/docs/3.1.x/changelog.txt

Spring 3.1 -> 3.2

- 레퍼런스 메뉴얼의 3.2 신규 기능 설명
  - http://static.springsource.org/spring-framework/docs/3.2.x/spring-framework-reference/html/new-in-3.2.html
- 스프링소스 블로그의 자료
  - http://blog.springsource.org/2012/05/06/spring-mvc-3-2-preview-introducing-servlet-3-async-support/
  - http://blog.springsource.org/2012/05/08/spring-mvc-3-2-preview-techniques-for-real-time-updates/
  - http://blog.springsource.org/2012/05/10/spring-mvc-3-2-preview-making-a-controller-method-asynchronous/

Spring 3.1 -> 3.2

- 스프링소스 블로그의 자료
  - http://blog.springsource.org/2012/05/13/spring-mvc-3-2-preview-adding-long-polling-to-an-existing-web-application/
  - http://blog.springsource.org/2012/11/05/spring-framework-3-2-rc1-released/
  - https://spring.io/blog/2012/11/07/spring-framework-3-2-rc1-new-testing-features
  - http://blog.springsource.org/2012/11/12/spring-framework-3-2-rc1-spring-mvc-test-framework/
  - http://blog.springsource.org/2012/12/13/spring-framework-3-2-goes-ga/

Spring 3.1 -> 3.2 상세변경 이력

- Deprecated : http://static.springsource.org/spring-framework/docs/3.2.x/javadoc-api/deprecated-list.html
- 변경 패키지, 클래스 목록 : http://static.springsource.org/spring-framework/docs/3.1.3.RELEASE_to_3.2.0.RELEASE/
- Change log : http://static.springsource.org/spring/docs/3.2.x/changelog.txt