Struts2

(주)아첸소프트웨어
대표 현철주
www.artszen.com
nockarm@gmail.com

Spring

스트럿츠2 ?

- 자바 서블릿과 자바 서버 페이지 기술에 기반한 웹 애플리케이션 개발 오픈 소스 프레임워크.
- 스트럿츠 프레임워크는 **Ant, Log4J, Tomcat**과 함께 가장 유명하고 성공한 아파치 자카르타 프로젝트 중 하나.
- 스트럿츠2 = 스트럿츠 + 웹워크.
- 애플리케이션 구축에서 배포, 유지 보수 전 영역에 거쳐 풀 개발 사이클 을 가질 수 있도록 디자인된 프레임워크.

스트럿츠2의 기능 및 특징

- 코어 **(core)**
- 뷰 **(view)**
- 기타 **(etc)**

스트럿츠2의 기능 및 특징 (계속)

● 코어 (core)

✱ 각 액션에 대해 요청의 라이프 사이클이 커스터마이즈될 수 있는 프레임워크이다.

✱ 데이터의 유효성 검사 규칙을 액션의 코드와 분리할 수 있는 유연한 프레임워크이다.

✱ 쉽고 단순한 방법으로 지역화된 애플리케이션을 구성할 수 있는 국제화 기능을 제공한다.

✱ 웹 애플리케이션을 개발할 때 가장 지루한 데이터 변환 작업을 해결하기 위하여 **HTTP**로부터 순수 자바 데이터 객체의 값으로 매핑하여 자동으로 변환하는 기능을 제공한다.

✱ 컴포넌트의 라이프 사이클과 의존성을 관리하는 의존성 삽입**(Dependency Injection)** 기능을 프레임워크 내에 포함시켰다. 기본적으로 의존성 삽입 기능을 위하여 스프링 프레임워크를 사용한다.

✱ 수 백개 이상의 액션들을 사용하게 되는 대형 프로젝트를 쉽게 관리하기 위한 패키지와 네임스페이스를 사용하여 환경설정 파일들을 모듈화한다.

✱ 자바5 어노테이션**(annotation)** 기능을 사용하여 환경설정의 오버헤드를 줄일 수 있다.

스트럿츠2의 기능 및 특징 (계속)

- **뷰 (view)**

✱ 테마와 템플릿을 사용하여 컴포넌트 기반의 개발을 할 수 있도록 하는 재사용성이 높은 사용자 인터페이스 태그를 제공한다. 스트럿츠2의 번들 태그들은 텍스트 입력 상자 같은 단순한 형태부터 데이트 픽커 또는 트리 뷰 같은 발전된 형태의 컴포넌트들을 제공한다.

✱ **JSTL** 호환 표현식 언어인 **OGNL**을 사용하여 객체들이 마치 단일 자바빈인 것처럼 다중 객체의 프로퍼티들을 액세스 할 수 있다.

✱ **JSP, FreeMarker, Velocity, PDF, JasperReports** 등을 포함한 다중 뷰 기술을 지원하는 결과 타입 개념을 사용한다.

✱ 양방향 대화식 웹 애플리케이션을 쉽게 개발하기 위한 **AJAX** 테마를 지원한다.

✱ 백그라운드에서 오랜 시간을 실행해야 하는 작업, 다중 폼 서브밋 방지 또는 사용자의 보안 계획을 적용하기에 적합한 인터셉터의 기능을 제공한다.

스트럿츠2의 기능 및 특징 (계속)

- 기타 (etc)

✱ 하이버네이트(Hibernate), 스프링(Spring), 사이트메쉬(Sitemesh), JSTL 같은 다른 유명한 제품들을 손쉽게 통합할 수 있다.

✱ 스트럿츠2 프레임워크는 아파치 라이센스 2.0 하에서 배포된다.

3가지 영역에서 바라보는 스트럿츠2의 기능 및 특징

- 애플리케이션 구축 영역
- 애플리케이션 배포 영역
- 애플리케이션 유지보수 영역

3가지 영역에서 바라보는 스트럿츠2의 기능 및 특징 (계속)

- 애플리케이션 구축 영역

✱ 개선된 디자인
모든 프레임워크 클래스들은 인터페이스에 기반을 두고 있다. 핵심 인터페이스들은 **HTTP**와 독립적으로 구성되어있다

✱ 개선된 결과**(Result)**
이전 버전의 스트럿츠의 **ActionForward**와 달리 스트럿츠2의 **Result**는 응답을 준비하는 데 실제로 도움을 준다.

✱ 개선된 태그들
스트럿츠2의 태그들은 데이터를 출력하기 위한 것 만은 아니다. 페이지를 구성하기 위한 코드를 최소화하기 위한 스타일시트-드리븐-마크업**(stylesheet-driven markup)**을 제공한다.

✱ 상태유지**(Stateful)** 체크박스
스트럿츠2 체크박스들은 **false** 값을 위하여 특별히 다뤄야 할 작업을 요구하지 않는다.

✱ 손쉬운 작업취소
스트럿츠2 취소버튼은 다른 액션으로 바로 갈 수 있도록 해준다.

3가지 영역에서 바라보는 스트럿츠2의 기능 및 특징 (계속)

● 애플리케이션 구축 영역 (계속)

✱ **POJO 폼(Form)**
더 이상 **ActionForm**에 국한될 필요가 없다. 원하는 어떠한 자바 빈을 사용할 수 있으며 **Action** 클래스들 상에 직접 정의한 프로퍼티들을 놓을 수도 있다. 모두 **String** 프로퍼티를 사용할 필요도 없다.

✱ **POJO 액션(Action)**
어떤 클래스도 **Action** 클래스로 사용할 수 있다. 인터페이스 조차도 옵션이 되었다.

✱ 손쉬운 스프링(Spring) 구축
스트럿츠2 액션들은 **Spring**을 인식한다. 단지 **Spring** 빈들을 추가하면 된다.

✱ **First-class AJAX** 지원
AJAX 테마는 클라이언트와 애플리케이션의 상호작용을 보다 강력하게 해준다.

✱ 스트럿츠2의 새 기능
Zero Configuration - XML 환경설정을 대화식 또는 어노테이션**(annotation)** 또는 옵션으로 대치할 수 있다.

3가지 영역에서 바라보는 스트럿츠2의 기능 및 특징 (계속)

- 애플리케이션 배포 영역

✱ 빠른 작업
변경된 환경설정 파일들의 내용이 웹 컨테이너를 다시 시작하지 않고 리로드 될 수 있다.

✱ 손쉬운 플러그인
프레임워크는 **JAR** 파일에 포함시켜 추가함으로써 확장할 수 있다. 어떠한 수작업 환경설정도 필요 없다. 번들 플러그인들은 **JavaServer Faces, JasperReports, JFreeChart, Tiles** 등을 지원한다.

✱ 손쉬운 포틀릿**(portlet)**
코드 변경 없이 자동으로 포탈과 서블릿 배포를 지원한다.

✱ 구축되어있는 프로파일링
로직 수행 중 사이클 과정을 찾기 위하여 스트럿츠 내부를 들여다 볼 수 있다.

3가지 영역에서 바라보는 스트럿츠2의 기능 및 특징 (계속)

- 애플리케이션 유지보수 영역

✱ 손쉬운 액션들의 테스트
스트럿츠2 Action들은 HTTP에 독립적이고, mock 객체에 의존하지 않고 테스트 할 수 있다.

✱ 디폴트 값 지원
프레임워크의 환경설정 요소들의 대부분은 디폴트 값을 가진다.

✱ 손쉬운 컨트롤러의 커스터마이징
스트럿츠1에서는 모듈 당 RequestProcessor를 커스트마이징하였으나, 스트럿츠2에서는 원한다면 액션 당 request handling을 커스터마이징 할 수 있다.

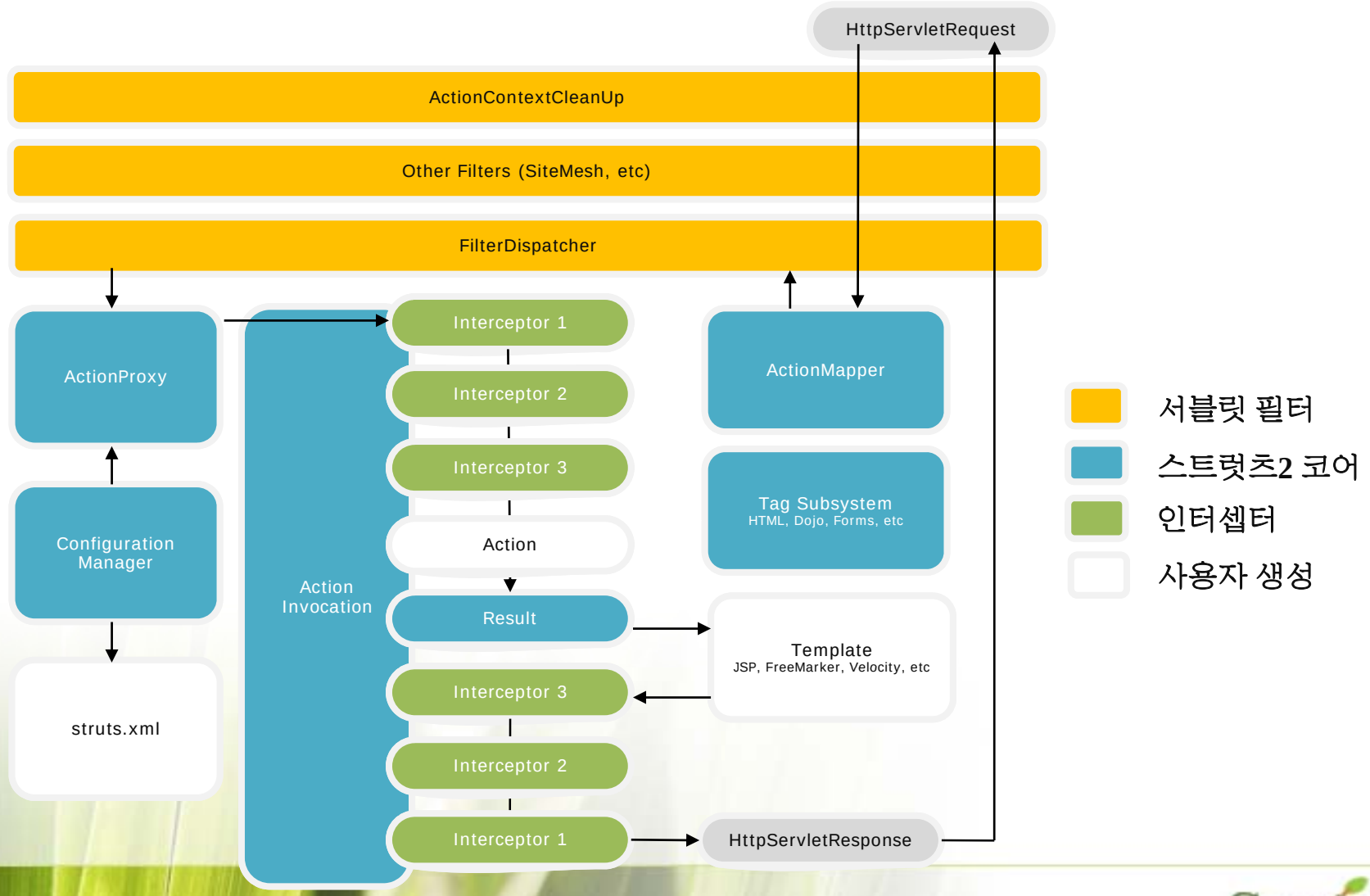
✱ 손쉬운 태그 변형
스트럿츠2 태그 마크업은 내재된 스타일시트를 변경하여 선택할 수 있다. 각각의 태그 마크업은 FreeMaker 템플릿을 편집하여 변경할 수 있다. 태그 라이브러리 API를 학습할 필요가 없다. JSP, FreeMarker, Velocity 태그들이 모두 지원된다.

✱ 디버깅 구축
프로파일링, 오류 보고 및 인터랙티브 객체 모델 쿼리를 지원하는 디버깅 도구가 내장되어있다.

스트럿츠2는 어떤 웹 페러다임에 적합한가?

- 액션 기반의 프레임워크
- 어노테이션 또는 **XML** 설정 옵션
- 테스트가 용이한 **POJO** 기반의 액션
- 스프링, 사이트메쉬, 타일즈 등의 통합
- **OGNL** 표현식 언어 통합
- 테마 기반의 태그 라이브러리와 **AJAX** 태그
- 다중 뷰 옵션 **(JSP, Freemarker, Velocity, XSLT)**

스트럿츠2의 아키텍처



Spring

스트럿츠2의 아키텍처 (계속)

- 액션 매퍼: **ActionMapper**
- 필터 디스패처 : **FilterDispatcher**
- 액션 프록시/액션 인보케이션 : **ActionProxy/ActionInvocation**
- 인터셉터 : **Interceptor**
- 액션 : **Action**
- 결과 : **Result**
- 밸류 스택 : **ValueStack**
- 환경설정 : **Configuration**
- 태그 라이브러리: **Tag Library**
- 의존성 주입 : **Dependency Injection**
- 손쉬운 **Ajax** 지원
- 플러그인: **Plugins**

스트럿츠2의 아키텍처 (계속)

● 액션 매퍼 : ActionMapper

✱ 필터 디스패처가 액션 매퍼를 통해서 액션 매핑(**ActionMapping**)을 생성한다.

✱ 액션 매핑은 **method, params, result** 등의 정보는 갖지 않는다.

✱ 액션 매퍼는 **HTTP** 요청과 액션 실행 사이에 매핑을 제공한다.

✱ **HttpServletRequest**가 주어졌을 때 일치하는 액션이 없다면 액션 매퍼는 널(**null**)을 반환하거나 프레임워크가 시도할 액션 호출에 관한 내용을 담고 있는 액션 매핑을 반환한다.

✱ 액션 매퍼가 반환하는 액션 매핑은 실제 액션이거나 유효한 요청일 것을 보장하도록 요구되지 않는다. 대부분의 액션 매퍼들은 요청을 매핑할 것인지를 결정하기 위해서 스트럿츠 환경설정을 참고할 필요가 없다.

✱ 액션 매퍼 종류
- ■ 디폴트 액션 매퍼
- ■ 커스텀 액션 매퍼
- ■ 레스트풀 액션 매퍼
- ■ 레스트풀2 액션 매퍼
- ■ 혼합 액션 매퍼

스트럿츠2의 아키텍처 (계속)

● 필터 디스패처: FilterDispatcher

✱ 필터디스패처는 스트럿츠2 프레임워크에서 요청에 대한 메인 엔트리 포인트이다. 이는 일반적으로 ***.action** 이라는 확장명과 매핑한다. 어떤 액션을 실행해야 되는지 결정하기 위해서 요청의 경로를 사용한다. 필터디스패처는 서블릿과 스트럿츠2의 **HTTP** 요청/응답과 **XWork**의 일반적인 커맨드 패턴**(Command Pattern)** 액션/결과**(Result)** 사이의 어댑터로서 작동한다.

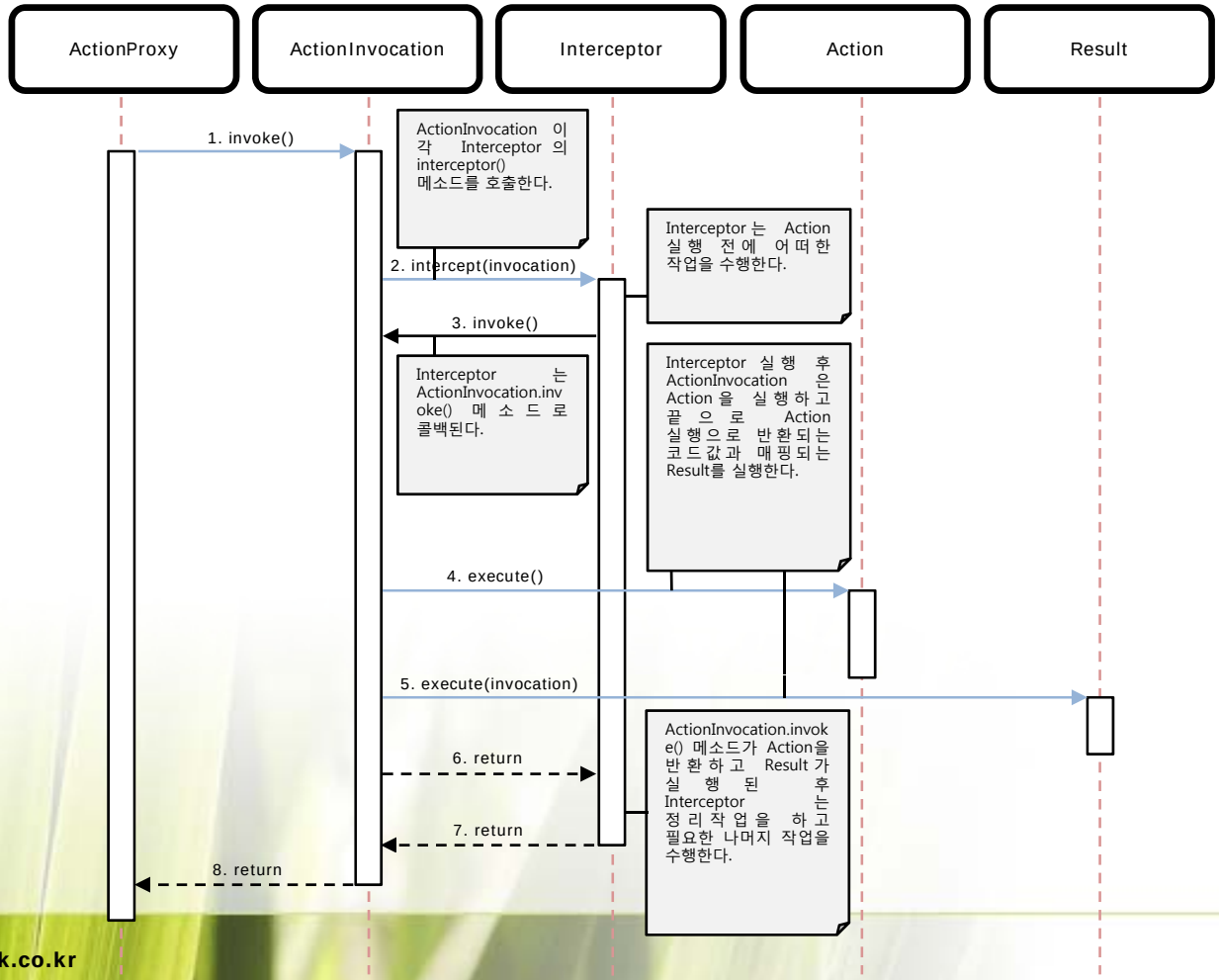
✱ 필터디스패처는 애플리케이션 스코프, 세션 스코프, 파라미터 스코프의 속성들을 래핑하는 **java.util.Map** 구현체를 설정하여 액션 실행 컨텍스트를 생성한다. 액션 프록시를 생성하기 위해서 액션 프록시 팩토리**(ActionProxyFactory)**를 사용한다. 만약 요청된 이름과 일치하는 액션이 존재하지 않는다면 사용자에게 오류를 반환한다.

✱ 필터디스패처는 인터셉터, 액션, 액션 실행 후 반환되는 코드 값과 매핑된 결과를 실행하는 액션 프록시를 실행한다. 예를 들면, 이 결과는 웹 페이지 또는 **PDF** 문서 등으로 렌더링된다. 또한 필터디스패처는 멀티 파트 파일 업로딩 요청에 대한 래핑을 다루고, 오류 코드를 다룬다.

✱ 스트럿츠2는 **MVC** 웹 애플리케이션 프레임워크로 래핑된 커맨드 패턴 구현체**(XWork)**이다. 프레임워크는 명령 실행, 코드 호출이 액션으로부터 분리되도록 하고, 액션 실행 주변으로 서비스를 추가하도록 한다. 이러한 서비스는 스트럿츠2에서 인터셉터 형식으로 제공되는 것이며 이들은 프레임워크의 많은 핵심 기능을 다룬다.

스트럿츠2의 아키텍처 (계속)

- 액션 프록시/액션 인보케이션: **ActionProxy/ActionInvocation**



Spring

스트럿츠2의 아키텍처 (계속)

- 액션 프록시/액션 인보케이션: **ActionProxy/ActionInvocation** (계속)

✱ 액션 프록시는 액션을 수행하기 위하여 제공되는 대행자이다. 액션은 프레임워크를 통해 실행되기 때문에 인터셉터, 결과 등의 나머지 기능들을 인캡슐레이션 하기 위해서는 액션 인스턴스 그 자체 보다 이 대행자를 사용한다.

✱ 액션 프록시는 액션 실행의 현재 상태를 나타내는 액션 인보케이션을 가진다. 액션 인보케이션은 액션 인스턴스, 순서대로 적용될 인터셉터들, 결과의 맵과 액션 컨텍스트를 가진다. 액션 프록시는 **static ActionProxyFactory** 인스턴스를 사용하여 필터디스패처에 의해 생성된다.

✱ 컨텍스트와 함께 액션 프록시를 생성한 후에 필터 디스패처는 **execute()** 메소드를 호출하여 액션 프록시를 실행시킨다.

✱ 액션 프록시는 액션 인보케이션의 실행 컨텍스트를 설정한 후 액션 인보케이션의 **invoke()** 메소드를 호출한다.

✱ 액션 인보케이션의 **invoke()** 메소드는 실행된 다음 인터셉터를 찾고 **interceptor()** 메소드를 호출한다. 인터셉터는 액션 인보케이션의 **invoke()** 메소드를 다시 호출하기 전에 액션 인보케이션을 사용하여 어떤 작업을 수행할 수 있다. 액션 인보케이션은 인터셉터가 실행되었는지 알 수 있는 상태 값을 유지하고 있으며, 하나 이상의 인터셉터가 존재한다면 다음 인터셉터의 **interceptor()** 메소드를 호출하게 된다. 더 이상 호출할 인터셉터가 없다면 다음은 액션 인스턴스가 실행된다. 액션이 반환하는 코드는 사용할 결과**(Result)**를 찾기 위하여 사용되고, 결과가 실행된다.

스트럿츠2의 아키텍처 (계속)

- 액션 프록시/액션 인보케이션: **ActionProxy/ActionInvocation** (계속)

✱ 스택 상의 마지막 인터셉터에게 컨트롤을 넘겨주며 **invoke()** 메소드가 반환된다. 이 인터셉터는 나머지 작업을 수행한 후 이 전 인터셉터가 나머지 작업을 수행하도록 **invoke()** 메소드가 반환된다. 모든 인터셉터가 반환될 때까지 바로 위의 작업을 수행한다. 마지막으로 액션 프록시가 상태 값들을 청소한 후 반환된다.

✱ 액션 인보케이션을 통해 인터셉터로 진행되고 또 다른 인터셉터로 처리를 계속된다. 인터셉터는 액션을 계속 수행해야 할지를 선택할 수 있다.

✱ **ActionProxyFactory/ActionProxy/ActionInvocation** 아키텍처는 액션 실행에 대한 또 다른 전략을 가질 수 있다. 예를 들면, 이 아키텍처를 사용하여 비 동기적인 액션 실행과 리치 클라이언트가 서버를 호출하도록 하는 클라이언트 디스패처가 가능하도록 **Java Message Service (JMS)** 디스패처를 구축할 수 있다.

스트럿츠2의 아키텍처 (계속)

● 인터셉터: Interceptor

✱ 인터셉터란 액션 실행 주위로 실행될 수 있는 코드를 인캡슐레이션 할 수 있게 한다.

✱ 인터셉터는 액션 실행에 투명성을 제공할 수 있는 커맨드 패턴의 추가 서비스이다.

✱ 인터페이스는 액션 외부에 정의되고, 런타임 시에 액션과 액션 실행 환경을 엑세스할 수 있어, 관심사 분리와 크로스 커팅 코드를 할 수 있게 해준다. 크로스 커팅 코드는 데이터베이스 연결, 트랜잭션이나, 액션 실행 후 그들을 해제 하는 것처럼 자원을 설정하는 실행 또는 그러한 실행에 대한 로깅, 실행 시간 등과 같은 어떤 것이 될 수 있다. 액션 인스턴스의 프로퍼티를 설정하는 것과 같은 기능들이 바로 인터셉터를 이용하여 수행된다.

✱ 인터셉터는 액션이 호출되기 전과 후에 명령코드를 실행할 수 있다. 대부분의 프레임워크 핵심 기능들은 인터셉터로 구현되어있다.

- ■ 이중 서브밋 방지
- ■ 타입 변환
- ■ 객체 파퓰레이션
- ■ 유효성 검사
- ■ 파일 업로드
- ■ 출력 페이지 준비

등과 같은 기능들은 모두 인터셉터의 도움으로 구현된다.

✱ 모든 인터셉터는 플러그인 방식이며, 인터셉터 스택으로 적용될 수 있다.

스트럿츠2의 아키텍처 (계속)

● 인터셉터: Interceptor (계속)

✱ 스트럿츠2에서 제공하는 인터셉터들

인터셉터	이름	설명
Alias Interceptor	alias	요청 사이에 서로 다른 이름을 가진 비슷한 파라미터들을 컨버팅한다.
Chaining Interceptor	chain	이전 액션의 프로퍼티들을 현재 액션에서 사용할 수 있게 한다. 이전 액션 정의에서 <result type="chain">과 함께 사용한다.
Checkbox Interceptor	checkbox	체크되지 않은 체크박스를 감지할 수 있는 코드를 자동으로 추가하고, 디폴트 값(보통 false)을 가진 파라미터들로 그들을 추가한다. 서브밋 되지 않은 체크박스를 감지하기 위해서 특별한 이름을 가진 히든(hidden) 필드를 사용한다.
Cookie Interceptor	cookie	액션에 설정가능한 '이름/값'을 가지는 쿠키 삽입(2.0.7부터 추가)
Conversion Error Interceptor	conversionError	ActionContext에서 액션의 필드 오류로 변환한다.

스트럿츠2의 아키텍처 (계속)

● 인터셉터: Interceptor (계속)

✱ 스트럿츠2에서 제공하는 인터셉터들 (계속)

인터셉터	이름	설명
Create Session Interceptor	createSession	자동으로 HttpSession을 생성한다. Token Interceptor와 같이 HttpSession이 적절하게 작동하도록 요구하는 인터셉터들에 유용하다.
Debugging Interceptor	debugging	페이지 뒷 단에 데이터를 출력할 수 있는 몇몇 디버깅 화면을 제공한다.
Execute and Wait Interceptor	execAndWait	백그라운드에서 액션을 실행하고 대기 상태 페이지를 사용자에게 즉각적으로 보낸다.
Exception Interceptor	exception	exception들을 결과(Result)에 매핑한다.
File Upload Interceptor	fileUpload	파일 업로드를 쉽게 할 수 있게 지원한다.
I18n Interceptor	i18n	사용자 세션에 대한 지역정보를 기억한다.
Logger Interceptor	logger	액션의 이름을 출력한다.

스트럿츠2의 아키텍처 (계속)

● 인터셉터: Interceptor (계속)

✱ 스트럿츠2에서 제공하는 인터셉터들 (계속)

인터셉터	이름	설명
Message Store Interceptor	store	ValidationAware 인터페이스를 구현한 액션에 대한 액션메시지/ 오류/ 필드오류을 저장하고 처리한다.
Model Driven Interceptor	model-driven	액션이 ModelDriven 인터페이스를 구현하였다면 getModel() 결과를 밸류 스택에 넣어 준다.
Scoped Model Driven Interceptor	scoped-model-driven	액션이 ScopedModelDriven 인터페이스를 구현하였다면 인터셉터는 스코프로부터 모델을 저장 처리하고, setModel()을 호출하여 액션에 저장한다.
Parameters Interceptor	params	요청 파라미터들을 액션에 저장한다.
Prepare Interceptor	prepare	액션이 Preparable 인터페이스를 구현하였다면 prepare() 메소드를 호출한다.
Scope Interceptor	scope	액션 상태 값을 세션 또는 애플리케이션 스코프에 저장하는 간단한 메커니즘

스트럿츠2의 아키텍처 (계속)

● 인터셉터: Interceptor (계속)

✱ 스트럿츠2에서 제공하는 인터셉터들 (계속)

인터셉터	이름	설명
Servlet Config Interceptor	servlet-config	HttpServletRequest와 HttpServletResponse를 다루는 맵을 엑세스할 수 있게 한다.
Static Parameters Interceptor	static-params	struts.xml에 정의된 파라미터 값들을 액션에 저장한다. <action> 태그 의 직속 자식으로 <param> 태그가 있다.
Roles Interceptor	roles	사용자가 올바른 JAAS 권한을 가진다면 액션은 실행된다.
Timer Interceptor	timer	인터셉터와 뷰 처리를 포함하여 액션의 작업 처리 시간을 출력한다.
Token Interceptor	token	액션 내에 유효한 토큰이 존재하는 지 검사하고 폼을 이중으로 서브밋하는 것을 방지한다.
Token Session Interceptor	token-session	Token Interceptor와 같다, 그러나 유효하지 않은 토큰을 가지고 있을 때 세션 내에 서브밋된 데이터를 저장한다.
Validation Interceptor	validation	action-validation.xml 내에 정의된 validator를 사용하여 유효성 검사를 한다.

스트럿츠2의 아키텍처 (계속)

● 인터셉터: Interceptor (계속)

✱ 스트럿츠2에서 제공하는 인터셉터들 (계속)

인터셉터	이름	설명
Workflow Interceptor	workflow	액션 클래스 내의 validate() 메소드를 호출한다. 액션에서 오류가 발생했다면 그것은 input 뷰로 리턴 한다.
Parameter Filter Interceptor	N/A	액션의 목록으로부터 파라미터를 제거한다.
Profiling Interceptor	profiling	파라미터를 통해서 프로파일링을 작동시킨다.

스트럿츠2의 아키텍처 (계속)

● 액션: Action

✱ 액션은 하나의 작업 단위이다.

✱ 액션은 하나의 **URL**이다.

✱ 액션은 하나의 클래스 또는 메소드이다.

✱ 액션은 한 종류의 비즈니스 로직을 수행하기 위한 통로이다.

✱ 액션은 **POJO** 클래스이다.

✱ 액션은 **struts.xml** 에서 정의된다.

✱ 액션 메소드는 하나의 토큰 스트링을 반환한다.

✱ 액션체인 : 액션을 수행한 결과로서 다른 액션을 호출할 수 있다.

✱ **<s:action />** 태그를 이용한 다중 액션 호출

✱ 액션의 프로퍼티는 태그와 매핑할 수 있다. **(params** 인터셉터의 역할**)**

스트럿츠2의 아키텍처 (계속)

- 액션: **Action** (계속)

✱ 액션 구성

```
...
public class MyAction {
    private User user;
    public String execute() throws Exception {
        ...
        return "success"
    }
    ...
}
```

MyAction.java

```
...
<action name="myAction">
    <result name="success">
        myAction.jsp
    </result>
</action>
...
```

struts.xml

```
<%@ taglib prefix="s" uri="/struts-tags" %>
...
<s:property value="user.name" />
...
```

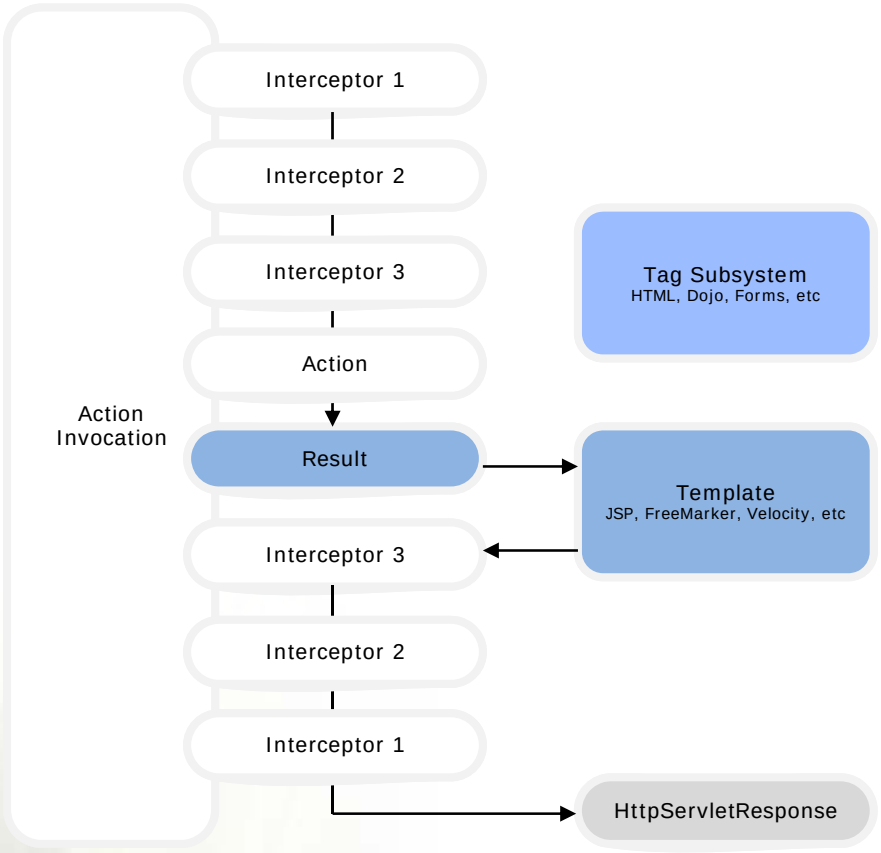
myAction.jsp

스트럿츠2의 아키텍처 (계속)

- 결과: **Result**

✱ 대부분의 유즈케이스들은 2개의 관점으로 분리된다. 첫 번째는 애플리케이션의 상태를 변경하거나 요구할 필요가 있고, 다음은 애플리케이션의 갱신된 뷰(View)를 출력할 필요가 있다. 액션 클래스는 애플리케이션의 상태를 관리하고 결과 타입은 뷰를 관리한다.

✱ 결과는 액션 인보케이션에 의해서 인터셉터 체인의 고리 제일 안쪽에서 액션이 수행된 후에 실행된다.



스트럿츠2의 아키텍처 (계속)

● 결과: Result (계속)

✱ 결과 타입 (Result Type)

결과 타입	설명
Chain Result	액션 체인을 위하여 사용된다.
Dispatcher Result	JSP를 포함한 웹 리소스 연동을 위하여 사용된다.
FreeMarker Result	FreeMarker 연동을 위하여 사용된다.
HttpHeader Result	특수한HTTP 작동을 조절하기 위해서 사용된다.
Redirect Result	다른 URL(웹 자원)로 리다이렉트하기 위하여 사용된다.
Redirect Action Result	다른 액션 매핑으로 리다이렉트하기 위하여 사용된다.
Stream Result	브라우저에게 inputStream을 스트리밍하기 위하여 사용된다. (예: 파일 다운로드 등)
Velocity Result	Velocity 연동을 위하여 사용된다.
XSLT Result	XML/XSLT 연동을 위하여 사용된다.
PlainText Result	특수한 페이지(예: jsp, html 등)의 원본 컨텐츠를 디스플레이하기 위하여 사용된다.
S2PLUGINS:Tiles Result	타일즈 구축을 위하여 사용된다.

스트럿츠2의 아키텍처 (계속)

- 결과: **Result** (계속)

✱ 체인 결과 타입 예:

```
...
<action name="myAction1">
    <result type="chain">myAction2</result>
</action>

<action name="myAction2">
    <result>myAction2.jsp</result>
</action>
...
```

struts.xml

스트럿츠2의 아키텍처 (계속)

● 밸류 스택: ValueStack

✱ 밸류 스택은 **XWork**와 스트럿츠2의 동적 컨텍스트 기반의 핵심이다.

✱ 밸류 스택은 객체의 스택이다. 객체 이름의 프로퍼티를 갖는 첫번째 객체를 검색함으로 써 동적으로 프로퍼티 값을 찾기 위하여 다뤄진다.

✱ 스트럿츠2는 액션이 실행되는 동안 액션을 스택상에 저장함으로서 밸류 스택을 구축한다.

✱ 스트럿츠2의 여러 **JSP** 태그들과 **Velocity** 매크로들은 밸류 스택을 엑세스 하고, 밸류 스택 로부터 객체를 읽어오거나 저장한다.

✱ 밸류 스택은 **OGNL(Object Graph Navigation Language)** 기반으로 구축되고, 다중 객체 스택을 지원하기 위하여 **OGNL**의 단일 객체 루트 개념을 확장하는 방식으로 작동한다.

스트럿츠2의 아키텍처 (계속)

● 밸류 스택: ValueStack (계속)

✱ OGNL (Object Graph Navigation Language)

■ **OGNL**은 자바 객체의 프로퍼티 값을 얻거나 저장하기 위하여 자바빈들 상의 프로퍼티들을 추적하는 표현식을 다룰 수 있게 한다. **(http://www.opensymphony.com/ognl** 참조**).**

■ **OGNL**은 정적 또는 인스턴스 메소드 실행**,** 컬렉션을 가로지르는 투영과 표현식 재사용을 위한 **Lambda** 표현식과 같은 진보된 표현식 기능을 제공한다**.** 또한 **OGNL**은 **XWork** 내에서 확장된 풍부한 타입**-**변환 모델을 제공한다**.**

■ **OGNL** 언어의 기본은 단순하며**,** 일반적인 사용법의 **90**퍼센트 정도를 다룬다**.** 기본적인 빈 프로퍼티들은 프로퍼티 이름으로 엑세스 된다**.**

(예) 표현식 **count**는 이름이 **count**인 프로퍼티의 **getter** 메소드 **getCount()**를 찾는다**.** 표현식 **address.street**는 프로퍼티를 얻으려 하는 경우에는 **getAddress().getStreet()**를 호출하고**,** 프로퍼티에 값을 저장하려 하는 경우에는 **getAddress().setStreet()**를 호출하게 된다**.**

■ **OGNL**의 또 다른 기능으로는 직진성을 들 수 있다**.**
(예) 표현식 **hashCode()**는 **OGNL** 컨텍스트 내의 현재 객체 상의 **hash code** 메소드를 호출한다**.**

스트럿츠2의 아키텍처 (계속)

● 환경설정: Configuration

✱ 스트럿츠 개발자의 입장에서 보면 프레임워크가 요구하는 단 하나의 환경설정 파일은 **web.xml**이다. 그러나 스트럿츠가 자신과 애플리케이션 양쪽을 컨트롤할 수 있게 해야 한다. 기본적으로 스트럿츠는 자신을 설정하는 내부 환경설정 파일을 로드한다. 그 다음 애플리케이션을 설정한다.

✱ 프레임워크와 애플리케이션을 설정할 때 필요한 파일 목록

파일	필수	위치(상대경로)	목적
web.xml	예	/WEB-INF/	모든 필수 프레임워크 컴포넌트를 포함하기 위한 웹 배치 디스크립터
struts.xml	아니오	/WEB-INF/classes/	result/view type, action mappings, interceptors 등을 포함한 메인 환경설정
struts.properties	아니오	/WEB-INF/classes/	프레임워크 속성
struts-default.xml	아니오	/WEB-INF/lib/strtus-core.jar	스트럿츠에 의해 제공되는 기본 환경설정
struts-default.vm	아니오	/WEB-INF/classes/	velocity.properties에 의해 참조되는 기본 매크로
struts-plugin.xml	아니오	플러그인 jar 파일의 루트경로	struts.xml과 같은 형식으로 되어있는 플러그인을 위한 선택적 환경설정 파일
velocity.properties	아니오	/WEB-INF/classes	기본 Velocity 환경설정 파일 오버라이드

스트럿츠2의 아키텍처 (계속)

● 환경설정: Configuration (계속)

✱ web.xml 에서 FilterDispatcher 설정

```
...
<filter>
    <filter-name>struts</filter-name>
    <filter-class>
        org.apache.struts2.dispatcher.FilterDispatcher
    </filter-class>
</filter>
...
<filter-mapping>
    <filter-name>struts</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
...
```

스트럿츠2의 아키텍처 (계속)

● 환경설정: Configuration (계속)

✱ struts.xml

- ■ 다른 환경설정 파일 포함
- ■ 인터셉터 설정
- ■ 인터셉터 스택 설정
- ■ 패키지 설정
- ■ 네임스페이스 설정
- ■ 액션 설정 : 와일드 카드 매핑 지원
- ■ 결과 설정
- ■ 상수 값 설정
- ■ 빈 설정
- ■ 예외 설정

스트럿츠2의 아키텍처 (계속)

● 환경설정: Configuration (계속)

✱ struts.xml (계속)

■ 다른 환경설정 파일 포함

```
<!DOCTYPE struts PUBLIC "-//Apache Software Foundation//DTD
Struts Configuration 2.0//EN"
"http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>
    <include file="Home.xml"/>
    <include file="Hello.xml"/>
    <include file="Simple.xml"/>
    <include file="/util/POJO.xml"/>
    ...
</struts>
```

스트럿츠2의 아키텍처 (계속)

● 환경설정: Configuration (계속)

✱ struts.xml (계속)

■ 인터셉터 / 인터셉터 스택 설정

```
...
<interceptors>
    <interceptor name="security"
        class="com.company.security.SecurityInterceptor"/>
    <interceptor-stack name="secureStack">
        <interceptor-ref name="security"/>
        <interceptor-ref name="defaultStack"/>
    </interceptor-stack>
</interceptors>
...
```

스트럿츠2의 아키텍처 (계속)

● 환경설정: Configuration (계속)

✱ struts.xml (계속)

■ 패키지 / 네임스페이스 / 액션 / 결과 설정

```
...
<package name="myPackage" namespace="/simple"
extends="defaultStack">
    <action name="myAction" class="mypackage.simpleAction"
    method="execute">
        <result name="success"
            type="dispatcher">myPage.jsp</result>
    </action>

    <!-- 위의 액션 정의와 동일
    <action name="myAction" class="mypackage.simpleAction">
        <result>myPage.jsp</result>
    </action>
    -->
</package>
...
```

스트럿츠2의 아키텍처 (계속)

● 환경설정: Configuration (계속)

✱ struts.xml (계속)

■ 액션의 와일드카드 매핑 설정

```
...
<action name="*Question" class="example.QuestionAction"
method="{1}Question">
    <result>/example/{1}Question.jsp</result>
</action>
...
```

[url]	[action method]	[jsp]
addQuestion.action	**add**Question()	**add**Question.jsp
viewQuestion.action	**view**Question()	**view**Question.jsp
editQuestion.action	**edit**Question()	**edit**Question.jsp
listQuestion.action	**list**Question()	**list**Question.jsp

스트럿츠2의 아키텍처 (계속)

● 환경설정: Configuration (계속)

✱ struts.xml (계속)

■ 액션의 와일드카드 매핑 설정 (계속)

```
...
<action name="list*" class="example.{1}Action" method="list{1}">
    <result>/{1}/list{1}.jsp</result>
</action>
...
```

[url]	[action]	[method]	[jsp]
list**Dept**.action	**Dept**Action	list**Dept**()	/**Dept**/list**Dept**.jsp
list**User**.action	**User**Action	list**User**()	/**User**/list**User**.jsp
list**Role**.action	**Role**Action	list**Role**()	/**Role**/list**Role**.jsp
list**Code**.action	**Code**Action	list**Code**()	/**Code**/list**Code**.jsp

스트럿츠2의 아키텍처 (계속)

● 환경설정: Configuration (계속)

✱ struts.properties

■ 스트럿츠2는 사용자가 필요한 프레임워크의 기능에 적합하게 변화시키기 위한 몇 가지 프로퍼티를 가진다.

■ 이러한 프로퍼티를 변경하기 위하여 **struts.properties** 파일에 프로퍼티 키 와 값으로 선언한다.

■ **properties** 파일은 클래스 패스 어떤 곳이라도 위치할 수 있다. 보통 **/WEB-INF/classes** 하위에 위치한다.

```
struts.i18n.reload=true
struts.devMode=true
struts.configuration.xml.reload=true
struts.continuations.package=org.apache.struts2.showcase
struts.objectFactory=spring
struts.custom.i18n.resources=globalMessages
struts.url.http.port=8080
struts.serve.static=true
struts.serve.static.browserCache=false
struts.multipart.maxSize=2097152
```

스트럿츠2의 아키텍처 (계속)

● 환경설정: Configuration (계속)

✱ 어노테이션: **Annotation**

스트럿츠2는 여러 곳에 **XML**에 자바 프로퍼티를 설정하는 처럼 자바5 어노테이션 기능을 사용할 수 있다.

- ■ 액션 어노테이션**(Action Annotation)**
- ■ 인터셉터 어노테이션**(Interceptor Annotation)**
- ■ 유효성 검사 어노테이션**(Validation Annotation)**
- ■ 타입 변환 어노테이션**(Type Conversion Annotation)**

스트럿츠2의 아키텍처 (계속)

● 환경설정: Configuration (계속)

✱ 액션 어노테이션의 **@Result Annotation** 사용 예

액션 어노테이션을 사용하려면 **web.xml**에 **FilterDispatcher** 필터를 등록할 때에 파라미터로 **actionPackages** 를 등록해야 한다.

```
...
<filter>
  <filter-name>struts2</filter-name>
  <filter-class>org.apache.struts2.dispatcher.FilterDispatcher</filter-
class>
  <init-param>
    <param-name>actionPackages</param-name>
    <param-value>example.annotation,example.test</param-value>
  </init-param>
</filter>
...
```

@Result 어노테이션을 사용할 액션 클래스가 존재하는 패키지를 콤마로 구분하여 **<init-param>** 태그 사이에 정의한다.

스트럿츠2의 아키텍처 (계속)

● 환경설정: Configuration (계속)

✱ 액션 어노테이션의 **@Result Annotation** 사용 예 (계속)

```
package example.annotation.test;

@Result(name="success", value="/annotation/main.jsp")
public class MainAction {
    ...
    public String execute throws Exception {
        result "success";
    }
}
```

web.xml에 **actionPackages** 파라미터에 '**example.annotation**' 패키지를 정의하였고, 위와 같이 액션 클래스 명이 '**MainAction**' 이라면 호출할 **url** 은 다음과 같이 결정된다.
만약 액션 클래스의 패키지가 **example.annotation.*subpackage*** 라면 네임스페이스 이름은 **'subpackage'** 가 되고, 액션 이름은 액션 클래스에서 '**Action**' 부분을 뺀 '**Main**' 의 소문자인 '**main**' 이 된다. 따라서 다음과 같이 된다.
http://host:port/context/namspace/actionname.action
예를 들어 웹 애플리케이션의 **context** 이름이 '**struts2**' 라면

```
http://localhost:8080/struts2/test/main.action
```

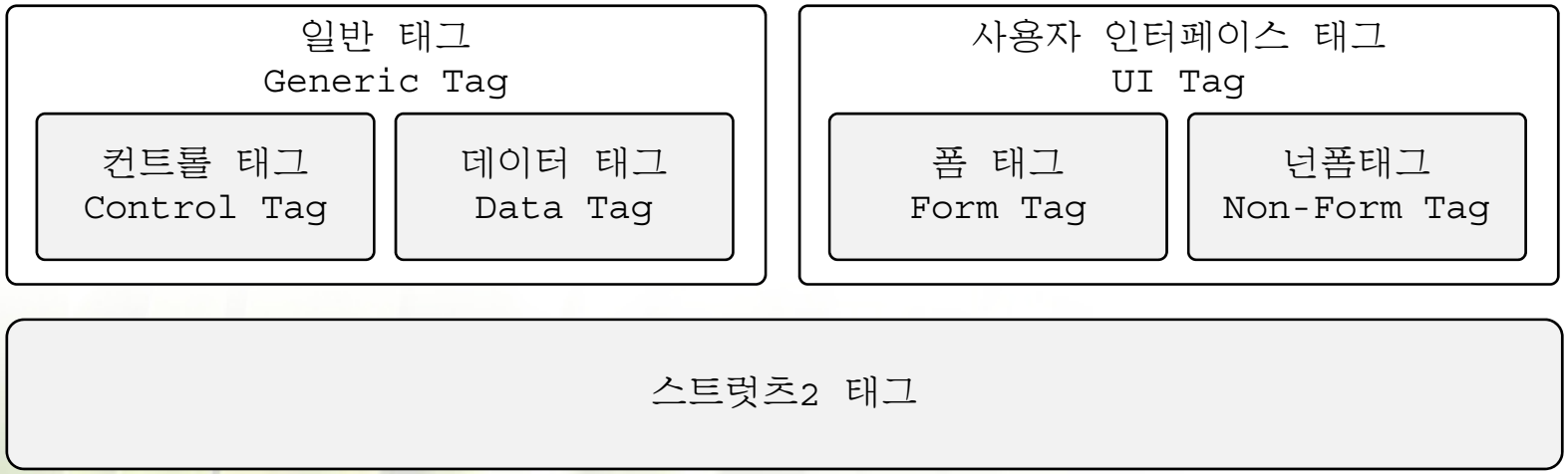
스트럿츠2의 아키텍처 (계속)

- 태그 라이브러리: **Tag Library**

✱ 스트럿츠2 태그들은 스트럿츠1.x와 많은 변화

✱ 최소의 코딩으로 풍부한 기능을 가진 웹 애플리케이션을 만들 수 있는 태그를 제공

✱ 스트럿츠의 태그에서는 **OGNL** 표현식을 제공
: 밸류스택에 저장되어있는 다중 객체의 프로퍼티들을 손쉽게 엑세스 할 수 있다.



스트럿츠2의 아키텍처 (계속)

● 일반 태그의 컨트롤 태그

컨트롤 태그들은 조건 처리와 반복적인 데이터를 처리하기 위한 태그들로 구성 되어있다.

태그 이름	설명
if	기본적인 조건 흐름을 수행한다. 'if' 태그는 그 자체로 사용되거나 'elseif' 태그, 'else' 태그와 함께 사용될 수 있다.
elseif	
else	
append	여러 리스트들을 리스트 순으로 하나의 iterator로 묶어준다.
generator	generator 태그의 val 속성에 정의된 값으로 iterator를 생성한다.
iterator	Iterator는 값들을 반복 처리 한다.반복 될 수 있는 값은 java.util.Collection, java.util.Iterator이다.
merge	여러 리스트들을 아이템 순으로 하나의 iterator로 묶어준다.
sort	태그 속성으로서 보내지는 Comparator를 사용해 목록을 정렬하는 태그이다.
subset	iterator의 서브셋을 취하는 태그이다.

스트럿츠2의 아키텍처 (계속)

● 일반 태그의 데이터 태그

데이터 태그들은 링크, 객체, 국제화, 액션처리, 파라미터, 메시지 번들 처리, 객체의 프로퍼티 처리 등을 위한 태그들로 구성되어있다.

태그 이름	설명
a	a 태그는 클릭 시에 dojo 프레임워크를 이용해 원격 XMLHttpRequest 호출을 하는 HTML <a href=''/>를 생성한다.
action	action 태그는 개발자가 액션 이름과 네임스페이스(선택사양)를 선언하여 JSP로부터 직접 액션을 호출할 수 있게 한다.
bean	자바빈 규정에 따라 클래스를 인스턴스화 한다.
date	date 태그는 신속하고 쉬운 방법으로 Date의 형식을 줄 수 있다.
debug	화면상에 [Debug] 하이퍼링크를 출력한다. 이 하이퍼링크를 클릭하면 밸류스택의 내용과 Stack Context 내용을 화면에 출력한다.
i18n	리소스 번들을 얻고 그 내용을 밸류스택에 저장한다.
include	서블릿의 출력(서블릿의 결과 또는 JSP 페이지)을 현재 페이지에 포함시킨다..

스트럿츠2의 아키텍처 (계속)

- 일반 태그의 데이터 태그 (계속)

태그 이름	설명
param	param 태그는 다른 태그의 파라미터로서 사용된다.
push	스택 안에 값을 저장한다.
set	특정 스코프의 변수에 값을 할당한다.
text	i18n 텍스트 메시지를 렌더링 한다.
url	이 태그는 url을 생성할 때 사용한다.
property	특정 값을 정의하지 않는다면 스택의 맨 위의 값을 프로퍼티로 취한다. 액션의 프로퍼티의 값을 화면에 출력할 수 있다.

스트럿츠2의 아키텍처 (계속)

● 사용자 인터페이스 태그의 폼 태그

사용자 인터페이스 태그는 화면에서 사용할 컨트롤들을 다루는 태그로 구성된다. 예를 들면 체크박스, 체크박스의 목록, 콤보박스, 등과 같은 컨트롤들로 이루어져 있다. 컨트롤들에 기능을 추가하기 위하여 개발자가 자바스크립트를 이용하여야 했던 것들을 손쉽게 사용할 수 있도록 태그 라이브러리 안에 포함되어 있다.

태그 이름	설명
checkbox	밸류스택으로부터 특정 프로퍼티에 의해 파퓰레이트되는 input 요소의 타입이 체크박스인 요소로 렌더링된다.
checkboxlist	list로부터 체크박스 시리즈를 생성한다. 사용방법은 <s:select /> 또는 <s:radio />와 비슷하다. 그러나 체크박스 태그를 생성한다.
combobox	콤보박스는 기본적으로 텍스트를 입력할 수 있는 HTML input 과 HTML select를 그룹으로 함께 제공한다.
datetimepicker	datetimepicker 요소를 렌더링한다. DateTimePicker 위젯은 연도, 월, 주를 증가 시키면서 날짜를 쉽게 선택할 수 있게 해주는 기능을 갖는다.
doubleselect	첫 번째 리스트박스의 선택된 내용에 따라 두 번째의 내용이 변경되어 출력하는 HTML select 요소를 렌더링한다.

스트럿츠2의 아키텍처 (계속)

- 사용자 인터페이스 태그의 폼 태그 (계속)

태그 이름	설명
head	HTML 파일에 대한 HEAD 섹션의 부분을 렌더링한다. 어떤 테마가 CSS를 요구하고 자바스크립트를 포함할 때 유용하다.
file	HTML file 요소를 렌더링한다.
form	HTML form 요소를 렌더링한다.
hidden	type이 hidden인 HTML input 요소로 렌더링된다.
label	HTML label 로 렌더링된다.
optiontransferselect	기본적으로 2개의 select 컴포넌트 사이에 option을 이동할 수 있는 컴포넌트를 생성한다.
optgroup	select 태그 내에 삽입할 optgroup 컴포넌트를 생성한다.
password	type이 password인 HTML input 요소로 렌더링된다.
reset	리셋 버튼으로 렌더링된다. 폼을 리셋하기 위하여 form 태그와 함께 사용된다.
select	type이 select인 HTML input 요소로 렌더링된다.

스트럿츠2의 아키텍처 (계속)

- 사용자 인터페이스 태그의 폼 태그 (계속)

태그 이름	설명
submit	서브밋 버튼으로 렌더링된다. 비동기적인 폼 제출을 위해 form 태그와 함께 사용된다.
textarea	HTML textarea 요소로 렌더링된다.
textfield	type이 text인 HTML input 요소로 렌더링된다.
token	폼의 이중 제출을 멈추게 한다.
updownselect	선택박스의 요소를 위, 아래로 이동 시킬 수 있는 버튼을 가진 select 컴포넌트를 생성한다.

스트럿츠2의 아키텍처 (계속)

● 사용자 인터페이스 태그의 넌-폼 태그

✱ 넌폼 태그(**Non-Form Tag**)는 폼에 종속되지 않고 단독으로 처리할 수 있는 태그들로 구성.
✱ 비동기적인 비즈니스 로직을 수행할 수 있는 **Ajax** 기능을 포함한 태그들이 포함

예를 들면, **<s:div/>, <s:tabbedPanel/>, <s:tree/>** 등을 대표로 들 수 있다. 일반적인 웹 애플리케이션에서 **Ajax**를 구현하기 위하여 **XmlHttpRequest**를 처리하기 위한 자바스크립트 코드가 필요하고, 서버 측에서 클라이언트로 비즈니스 데이터를 **XML**로 변환하여 전송해야 하며, 전송된 **XML** 데이터를 **DHTML**을 이용하여 동적으로 컴포넌트를 구성해야 한다. 그러나 스트럿츠2 프레임워크에서 제공하는 **ajax** 테마와 몇 몇 태그들을 이용하면 스트럿츠2의 다른 개발과 동일한 방식으로 **JSP**와 **Action** 클래스, 비즈니스 로직을 구현하면서도 이를 가능하게 해주는 매우 강력한 기능을 제공한다.

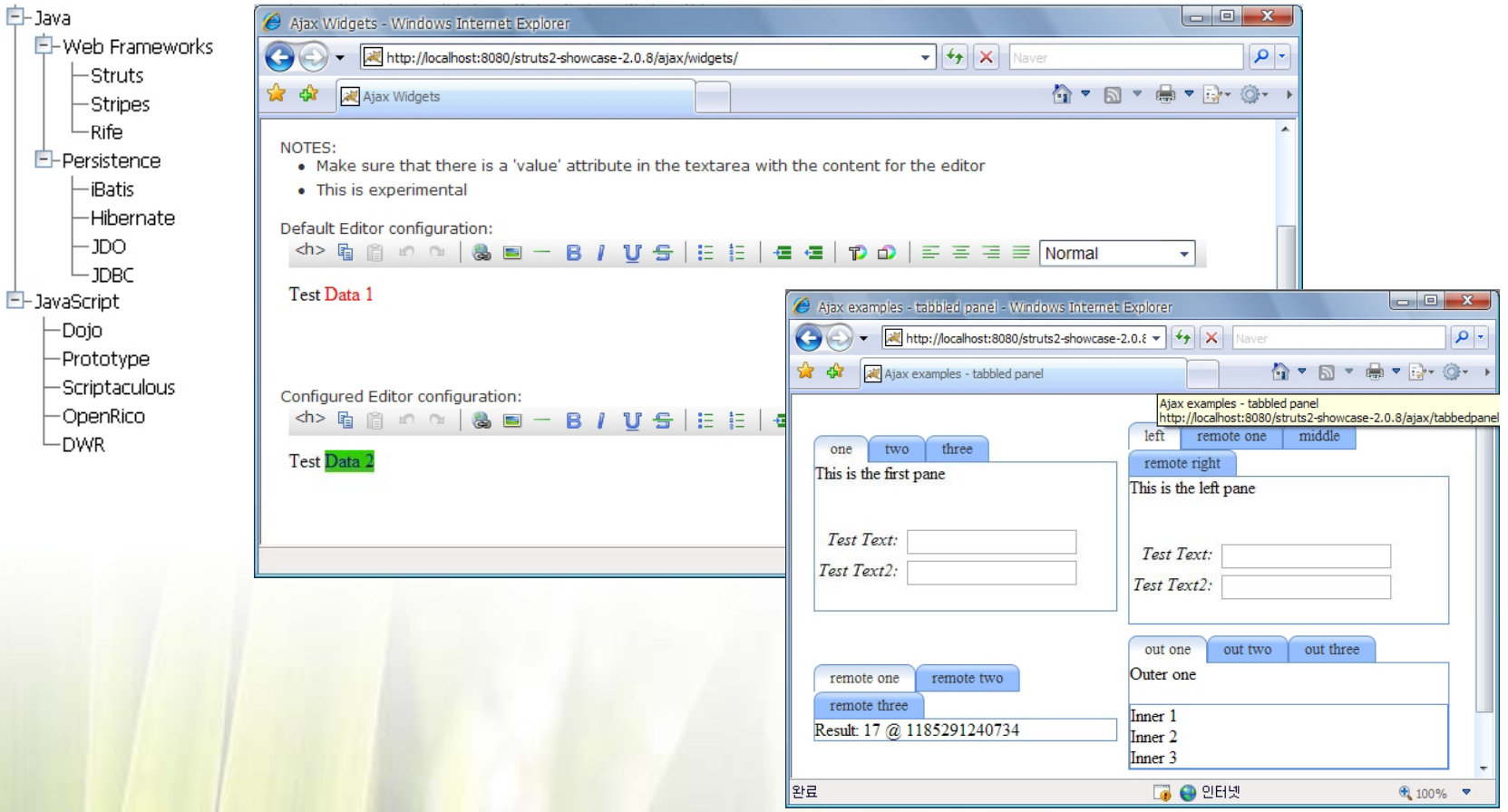
스트럿츠2의 아키텍처 (계속)

- 사용자 인터페이스 태그의 넌-폼 태그 (계속)

태그 이름	설명
actionerror	action error가 존재한다면 actionerror을 렌더링한다.
actionmessage	action message가 존재한다면 actionmessage를 렌더링한다.
component	특정 템플릿을 사용하는 커스텀 UI 위젯을 렌더링한다.
div	ajax테마를 사용할 때 div 태그는 전체 페이지를 갱신하지 않고 해당 내용을 갱신하기 위하여 현재 페이지로부터 원격호출을 제공한다.
fielderror	field error가 존재한다면 fielderror를 렌더링한다.
table	모델링 테이블을 렌더링하기 위한 태그
tabbedPanel	tabbedpanel 위젯은 AJAX 컴포넌트이다. 각 탭은 로컬 또는 리모트 컨텐츠가 될 수 있다. (사용자가 탭을 선택할 때 갱신된다.)
tree	AJAX를 지원하는 트리 위젯을 렌더링한다.
treenode	AJAX를 지원하는 트리 위젯 안의 트리 노드를 렌더링한다.

스트럿츠2의 아키텍처 (계속)

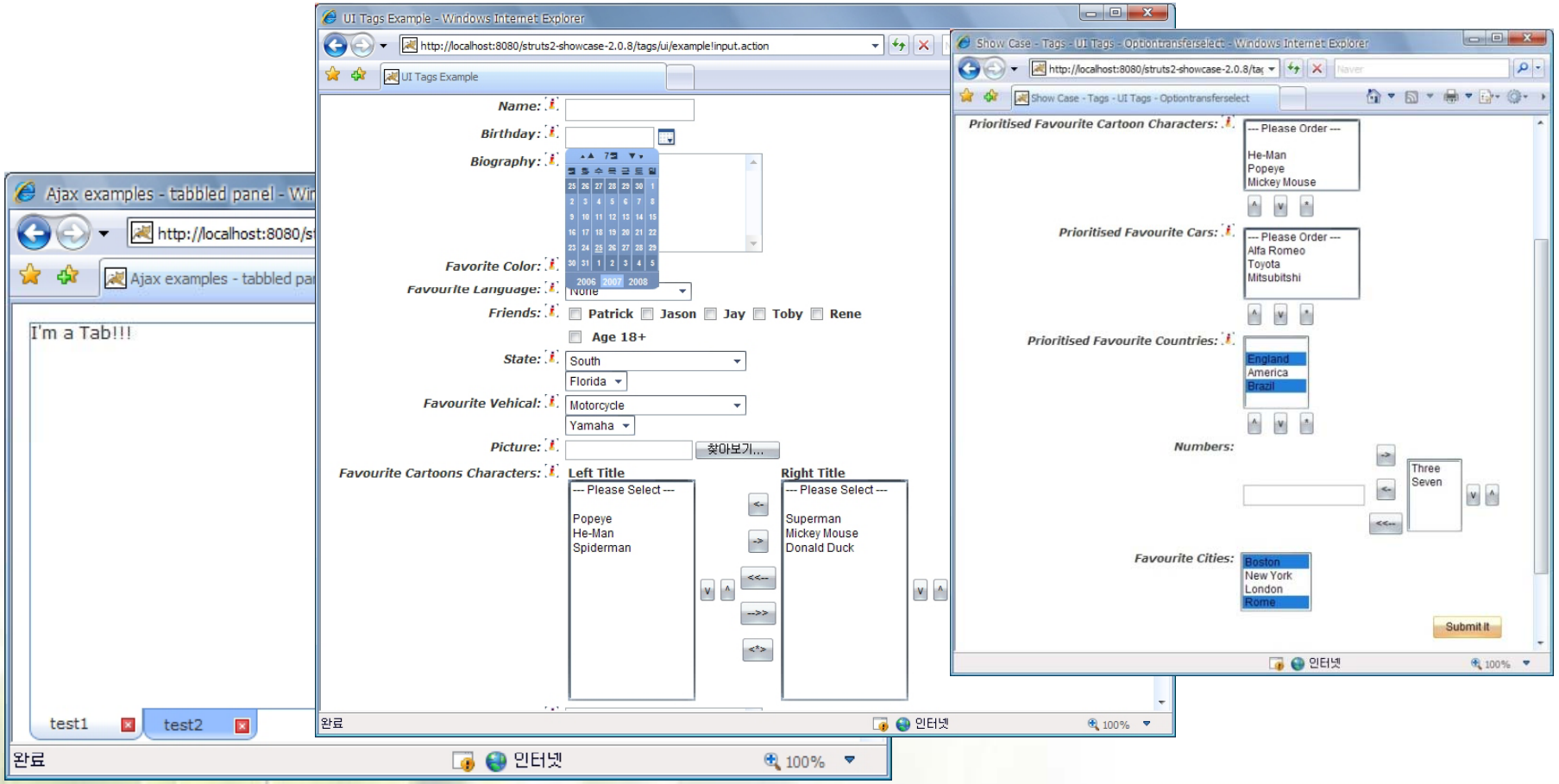
- 스트럿츠2의 태그 라이브러리 샘플



Spring

스트럿츠2의 아키텍처 (계속)

- 스트럿츠2의 태그 라이브러리 샘플 (계속)



Spring

스트럿츠2의 아키텍처 (계속)

● 의존성 주입: Dependency Injection

✱ 의존성 주입(**Dependency Injection**)은 객체를 생성해야 하는 책임과 객체 그 자신들과 팩토리와의 연결을 제거한다. **IoC(Inversion of Control** 컨테이너가 팩토리를 제공한다.

✱ 내부적으로 프레임워크는 **Google Guice** 기반의 의존성 주입 컨테이너를 사용한다. 스프링(**Spring**) 플러그인 **Plexus** 플러그인을 포함한 다른 컨테이너와 함께 애플리케이션을 구축할 수 있도록 플러그인을 사용할 수 있다.

✱ **WebWork 2.1**에 의해서 사용되는 **WebWork/XWork IoC** 컨테이너는 스트럿츠2에서 지원되지 않는다.

✱ 의존성 주입을 사용하게 되면 애플리케이션을 개발할 때 소스가 간결해질 수 있다. 의존성 주입 컨테이너에 설정 파일에 빈을 등록하고 그 빈을 사용하고자 하는 클래스에서 객체를 선언만 해놓고 사용할 수 있다. 객체를 생성하는 코드를 작성할 필요가 없어진다.

스트럿츠2의 아키텍처 (계속)

● 손쉬운 Ajax 지원

✱ 스트럿츠2는 애플리케이션에 **Ajax**기능을 제공하기 위하여 **Dojo** 프레임워크를 기반으로 한 태그들을 제공한다.

✱ **Ajax** 태그를 사용하기 위해서는 "**theme**" 태그 속성을 "**ajax**"로 설정해야 한다.
또한 **ajax** 테마를 위한 페이지를 설정하기 위해서는 <**s:head/**> 태그를 이용하여 테마를 "**ajax**"로 설정한다.

```
<html>
<head>
    <s:head theme="ajax"/>
</head>

<body>
    ...
    <s:div theme="ajax" .../>
    ...
</body>
</html>
```

스트럿츠2의 아키텍처 (계속)

● 손쉬운 Ajax 지원 (계속)

✱ Ajax를 지원하는 태그

기본 Ajax 태그	설명
<s:div>	Ajax를 통한 컨텐츠를 로드할 수 있는 영역을 생성한다. 선택적으로 영역을 갱신할 수 있는 기능을 제공한다.
<s:submit>	Ajax를 통해서 요소들을 갱신하거나 폼을 서브밋할 수 있다.
<s:a> (Anchor)	Ajax를 통해서 요소들을 갱신할 수 있다.
<s:tabbedPanel>	정적 또는 동적 <s:div.../> 탭 컨텐츠를 포함하는 탭 패널을 생성한다.
<s:autocompleter>	제안을 제공하거나 현재 값을 기반으로 요소를 갱신하는 Autocompleter 태그

스트럿츠2의 아키텍처 (계속)

- 플러그인: **Plugins**

✱ **Spring Plugin**

✱ **JasperReports Plugin**

✱ **JFreeChart Plugin**

✱ **Sitemesh Plugin**

✱ **Tiles Plugin**

✱ **Codebehind Plugin**

✱ **Config Browser Plugin**

✱ **JSF Plugin**

✱ **Plexus Plugin**

✱ **SiteGraph Plugin**

✱ **Struts 1 Plugin**

애플리케이션의 국제화

- 스트럿츠2의 국제화 지원

✱ **UI Tag**
✱ **ValidationAware** 인터페이스의 메시지와 오류 값
✱ **ActionSupport**를 상속받은 액션 또는 결과 페이지에서 **getText()** 메소드를 통하여 가능
✱ 리소스 번들 검색 순서

1. **ActionClass.properties**
2. **BaseClass.properties (Object.properties** 방향으로 진행)
3. **Interface.properties (**모든 인터페이스와 서브 인터페이스)
4. **ModelDriven's model (ModelDriven**을 구현하여다면)
5. **package.properties (**클래스의 디렉토리부터 루트 디렉토리 방향으로 진행)
6. 글로벌 리소스 번들

```
com/
  example/
    package.properties
    actions/
      package.properties
      SampleAction.java
      SampleAction.properties
```

SampleAction.properties 파일이 존재하지 않는다면 **com.example.package.properties** 파일이 검색된다.

만약 **com.example.package.properties** 파일이 존재하지 않는다면 **com.package.properties** 파일을 찾는다.

애플리케이션의 국제화

- 태그에서 **getText()**를 사용할 경우

```
<s:property value="getText('some.key')" />
```

- **text** 태그를 사용할 경우

```
<s:text name="some.key" />

<s:text name="some.invalid.key" >
   The Default Message That Will Be Displayed
</s:text>
```

- **i18n** 태그를 사용할 경우

```
<s:i18n name="some.package.bundle" >
     <s:text name="some.key" />
</s:i18n>
```

애플리케이션의 국제화

- **UI 태그에서 key 속성을 사용할 경우**

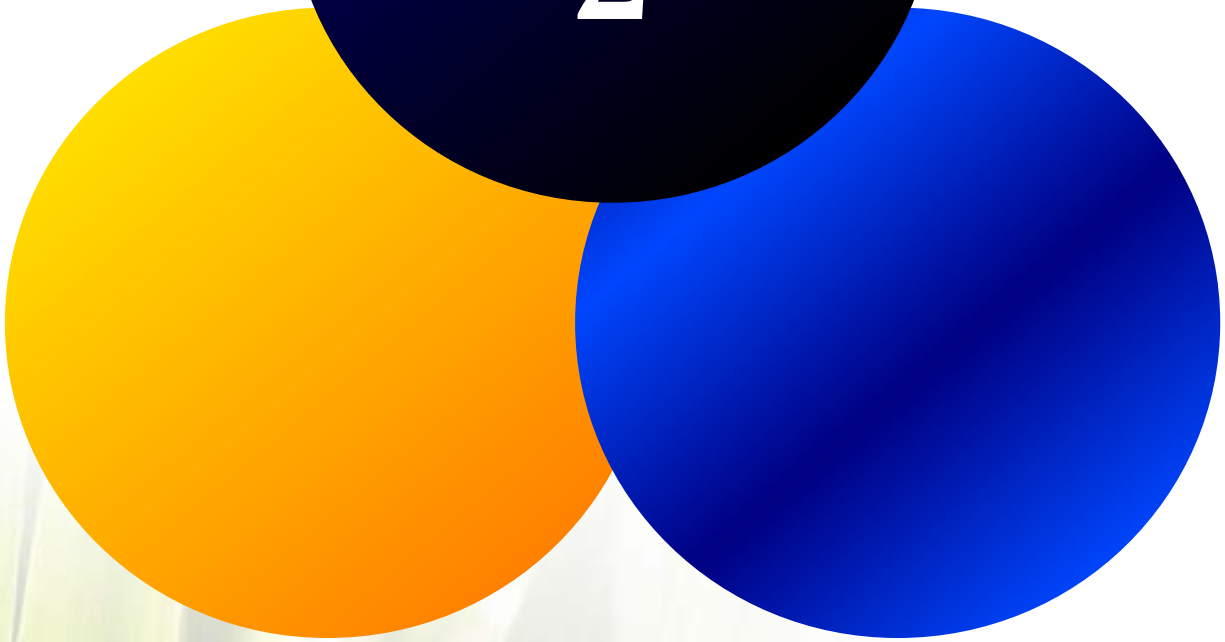
```
<s:textfield key="some.key" name="textfieldName"/>
```

- **struts.properties 안의 struts.custom.i18n.resources 엔트리에 정의된 글로벌 리소스 번들 파일을 사용할 경우**

```
struts.custom.i18n.resources=globalMessage
```

위와 같이 정의되어있다면 프레임워크는 **/classes/globalMessage.properties** 파일을 사용한다. 만약 우리나라에 적합한 글로벌 리소스 파일은 **globalMessage_ko.properties** 파일을 사용하면 된다.

Struts 2



Spring