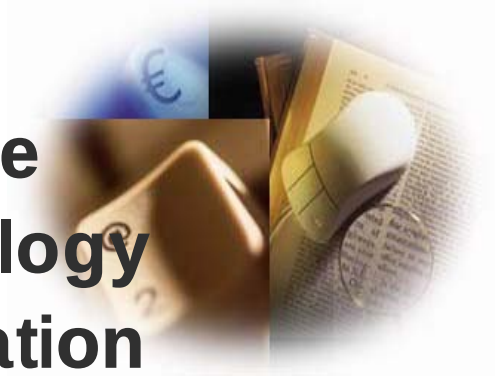
Korea
Software
Technology
Association

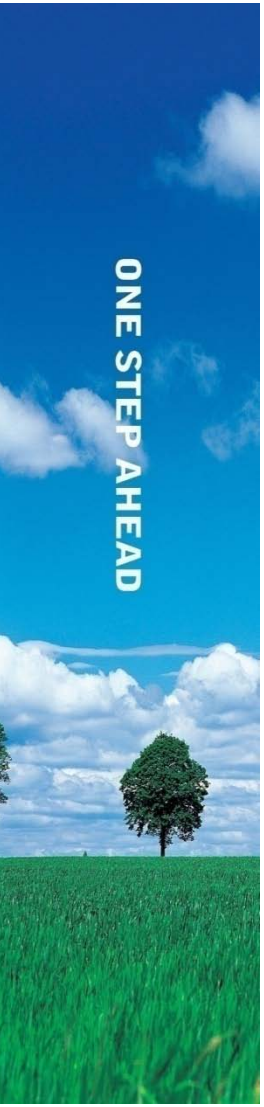


과정명 : 웹프레임워크의 활용
스트럿츠2 와 AJAX

강사명: 김 희 숙(넥스트리소프트 hskim@nextree.co.kr)

2.0.14 version으로 작성되었으며,
2.1.x version에서 사용하기 위해서는
별도의 Migration작업이 필요합니다.

2. 스트럿츠2 아키텍처

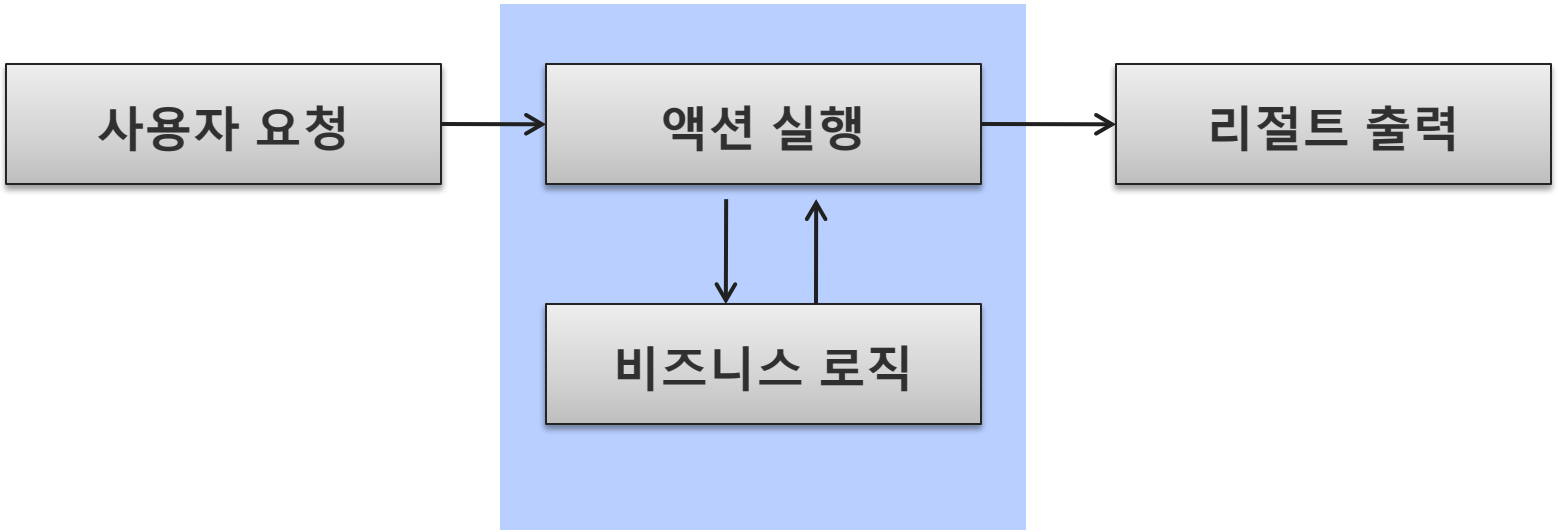


2.1 스트럿츠2 핵심구성요소
2.2 스트럿츠2 아키텍처

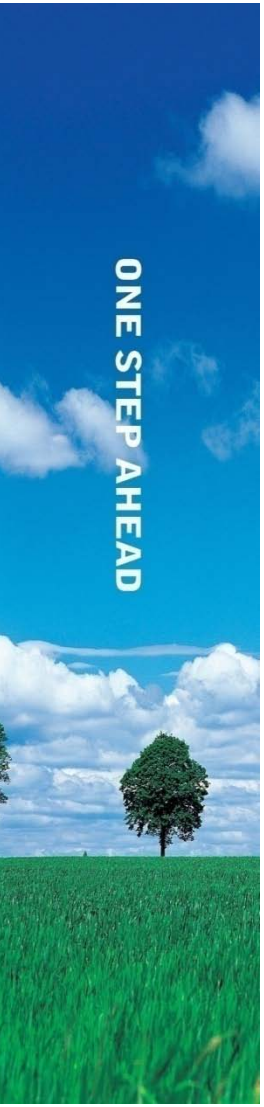
2.1 스트럿츠2 핵심구성요소

❑ 기본 흐름

- 사용자가 웹 서버에 요청시에 해당작업을 처리 후 결과 페이지를 출력



2.2 스트럿츠2 아키텍처

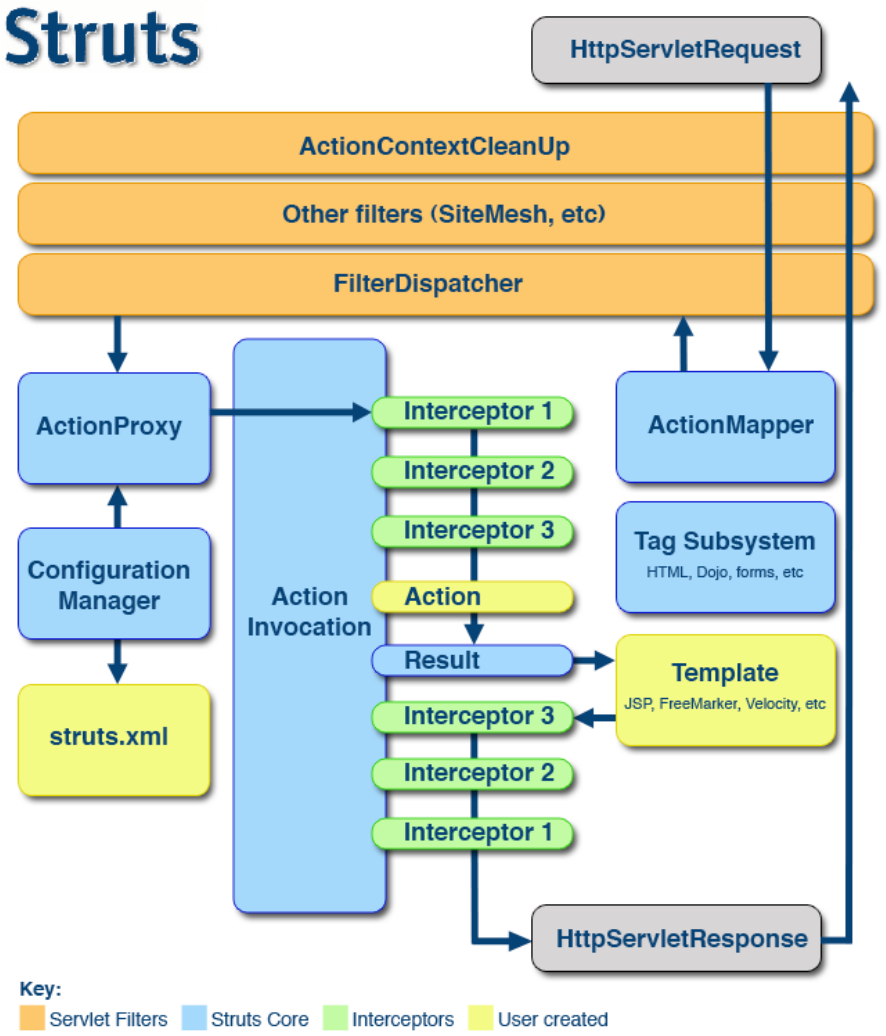


- ❑ 스트럿츠2 아키텍처 개념도
- ❑ 스트럿츠2 구성요소

스트럿츠2 아키텍처 개념도

❑ 스트럿츠2 아키텍처 구성요소

- 필터 디스패처
- 액션 매퍼
- 액션 프록시 / 액션 인보케이션
- 인터셉터
- 액션
- 결과
- 밸류 스택
- 환경설정
- 태그 라이브러리
- 의존성 주입
- AJAX 지원
- 플러그인



스트럿츠2 구성요소

❑ 필터 디스패처 (Filter Dispatcher)

- 스트럿츠2 프레임워크에서 요청에 대한 **메인 엔트리 포인트**
- 일반적으로 *.action 이라는 확장명과 매핑
- 어떤 액션을 실행해야 되는지 결정하기 위해서 요청의 경로를 사용
- 필터디스패처는 서블릿과 스트럿츠2의 HTTP 요청/응답과 XWork의 일반적인 커맨드 패턴(Command Pattern) 액션/결과(Result) 사이의 어댑터로서 작동
- 애플리케이션 스코프, 세션 스코프, 파라미터 스코프의 속성들을 래핑하는 java.util.Map 구현체를 설정하여 액션 실행 컨텍스트를 생성
- 액션 프록시를 생성하기 위해서 액션 프록시 팩토리를 사용
- 만약 요청된 이름과 일치하는 액션이 존재하지 않으면 사용자에게 오류 반환
- 인터셉터, 액션, 액션 실행 후 반환되는 코드 값과 매핑된 결과를 실행하는 액션 프록시를 실행

스트럿츠2 구성요소

❑ 액션 매퍼 (Action Mapper)

- 필터 디스패처가 액션 매퍼를 통해서 액션 매핑(ActionMapping)을 생성
- 액션 매핑은 method, params, result 등의 정보는 갖지 않음
- 액션 매퍼는 HTTP 요청과 액션 실행 사이에 매핑을 제공
- HttpServletRequest가 주어졌을 때 일치하는 액션이 없다면 액션 매퍼는 널(null)을 반환하거나 프레임워크가 시도할 액션 호출에 관한 내용을 담고 있는 액션 매핑을 반환
- 액션 매퍼가 반환하는 액션 매핑은 실제 액션이거나 유효한 요청일 것을 보장하도록 요구되지 않음
- 액션 매퍼 종류
  - 디폴트 액션 매퍼
  - 커스텀 액션 매퍼
  - 레스트풀 액션 매퍼
  - 레스트풀2 액션 매퍼
  - 혼합 액션 매퍼

스트럿츠2 구성요소

❑ 액션 프록시 / 액션 인보케이션 (Action Proxy / Action Invocation)

- 액션 프록시는 액션을 수행하기 위하여 제공되는 대행자
  - 액션은 프레임워크를 통해 실행되기 때문에 인터셉터, 결과 등의 나머지 기능들을 캡슐화하기 위해서 액션 인스턴스 그 자체 보다 이 대행자를 사용
- 액션 프록시는 static ActionProxyFactory 인스턴스를 사용하여 필터디스패처에 의해 생성
- 액션 프록시는 액션 실행의 현재 상태를 나타내는 액션 인보케이션을 가짐
- 액션 인보케이션은 액션 인스턴스, 순서대로 적용될 인터셉터들, 결과의 맵과 액션 컨텍스트를 가짐
- 컨텍스트와 함께 액션 프록시를 생성한 후에 필터 디스패처는 execute() 메소드를 호출하여 액션 프록시를 실행
- 액션 프록시는 액션 인보케이션의 실행 컨텍스트를 설정한 후 액션 인보케이션의 invoke() 메소드를 호출
- 액션 인보케이션의 invoke() 메소드는 실행된 다음 인터셉터를 찾고 interceptor() 메소드를 호출

스트럿츠2 구성요소

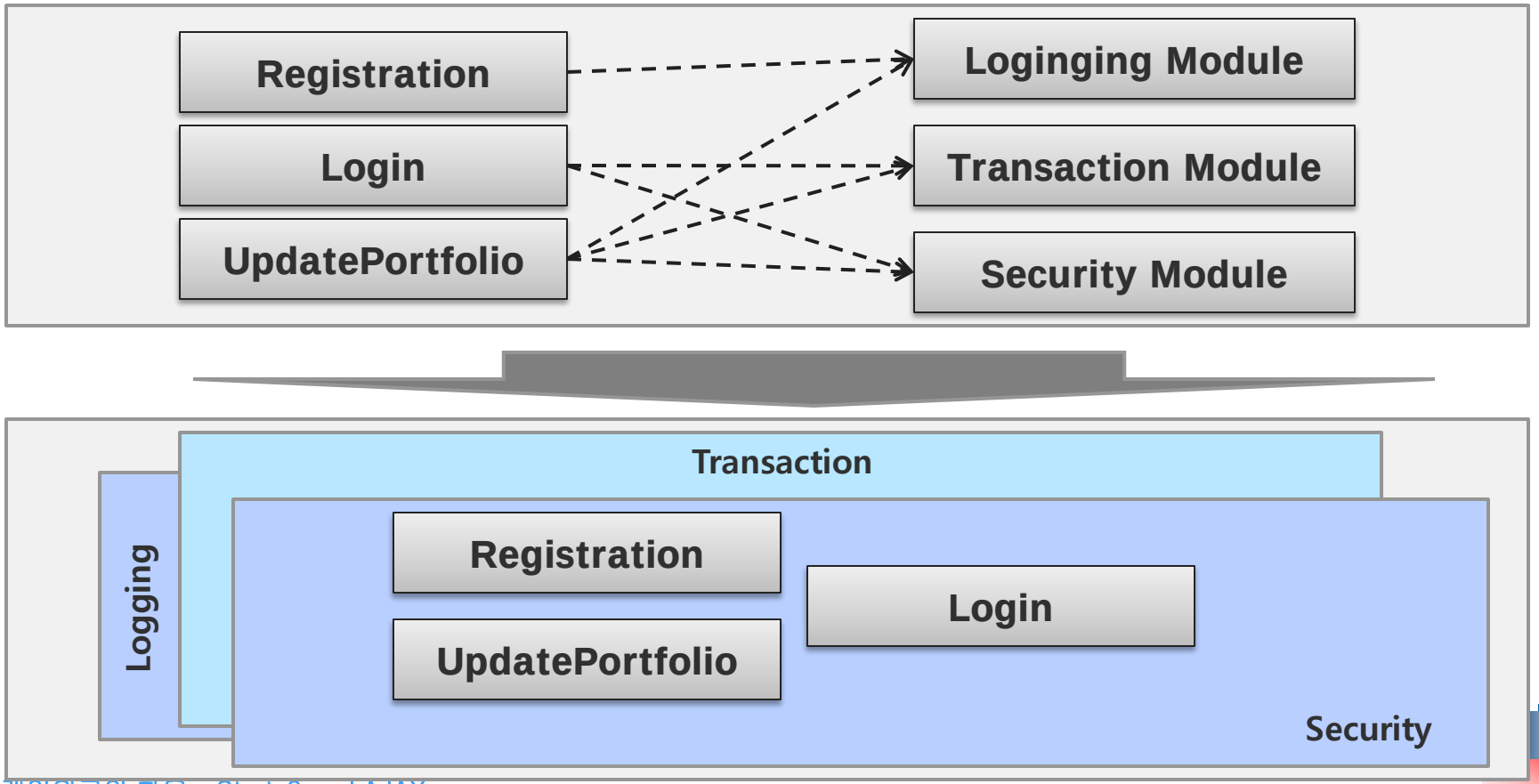
❑ 인터셉터 (Interceptor)

- 액션 실행 주위로 실행될 수 있는 코드를 캡슐화할 수 있게 함
- 액션 실행에 투명성을 제공할 수 있는 커맨드 패턴의 추가 서비스
- 인터페이스는 액션 외부에 정의되고, 런타임 시에 액션과 액션 실행 환경을 액세스할 수 있어, **관심사 분리와 크로스 커팅** 코드를 할 수 있게 해줌
- 액션이 호출되기 전과 후에 명령코드를 실행
- 대부분의 프레임워크 핵심 기능들은 인터셉터로 구현
  - 이중 서브밋 방지
  - 타입 변환
  - 객체 파퓰레이션
  - 유효성 검사
  - 파일 업로드
  - 출력 페이지 준비
- 모든 인터셉터는 플러그인 방식이며 인터셉터 스택으로 적용

스트럿츠2 구성요소 : 참조

❑ AOP(Aspect Oriented Programming)

- 공통의 관심 사항을 적용하여 발생하는 의존관계의 복잡성과 코드 중복을 해소해 주는 프로그래밍 기법



스트럿츠2 구성요소

❑ 액션 (Action)

- 하나의 작업 단위, 액션은 하나의 URL, 하나의 클래스 또는 메소드
- 액션은 한 종류의 비즈니스 로직을 수행하기 위한 통로이며 POJO 클래스임
- 액션은 struts.xml 에서 정의
- 액션 메소드는 하나의 제어 문자열을 반환
- 액션체인 : 액션을 수행한 결과로서 다른 액션을 호출할 수 있음
- <s:action /> 태그를 이용한 다중 액션 호출
- 액션의 프로퍼티는 태그와 매핑 (params 인터셉터의 역할)

스트럿츠2 구성요소

❑ 결과 (Result)

- 대부분의 유스케이스들은 2개의 관점으로 분리
- 첫 번째는 애플리케이션의 상태를 변경 요구, 다음은 애플리케이션의 갱신된 뷰(View)를 출력
- 액션 클래스는 애플리케이션의 상태를 관리하고 결과 타입은 뷰를 관리
- 결과는 액션 인보케이션에 의해서 인터셉터 체인의 제일 안쪽에서 액션이 수행된 후에 실행

스트럿츠2 구성요소

❑ 밸류 스택 (ValueStack)

- XWork와 스트럿츠2의 동적 컨텍스트 기반의 핵심
- 객체의 스택으로 객체 이름의 프로퍼티를 갖는 첫번째 객체를 검색함으로써 동적으로 프로퍼티 값을 찾기 위하여 다뤄짐
- 스트럿츠2는 액션이 실행되는 동안 액션을 스택상에 저장함으로서 밸류 스택을 구축
- OGNL(Object Graph Navigation Language) 기반으로 구축
- 다중 객체 스택을 지원하기 위하여 OGNL의 단일 객체 루트 개념을 확장하는 방식으로 작동

스트럿츠2 구성요소

- **OGNL (Object Graph Navigation Language)**
  - XWork와 스트럿츠2의 동적 컨텍스트 기반의 핵심
  - 자바 객체의 프로퍼티 값을 얻거나 저장하기 위하여 자바빈들 상의 프로퍼티들을 추적하는 표현식을 다룰 수 있게 함
  - 정적 또는 인스턴스 메소드 실행
  - OGNL 언어의 기본은 단순하며 기본적인 빈의 프로퍼티들은 프로퍼티이름으로 액세스

스트럿츠2 구성요소

❑ 환경설정 (Configuration)

- 프레임워크와 애플리케이션을 설정할 때 필요한 파일 목록

파일	필수	위치 (상대경로)	목적
web.xml	Yes	/WEB-INF/	모든 필수 프레임워크 컴포넌트를 포함하기 위한 웹 배치 디스크립터
struts.xml	no	/WEB-INF/classes	result/view type, action mapping, interceptor 등을 포함한 메인 환경설정
struts.properties	no	/WEB-INF/classes	프레임워크 속성
struts-default.xml	no	/WEB-INF/lib/struts2-core.jar	스트럿츠에 의해 제공되는 기본 환경설정
struts-default.vm	no	/WEB-INF/classes	velocity.properties에 의해 참조되는 기본 매크로
struts-plugin.xml	no	플러그인 jar 파일의 루트경로	struts.xml과 같은 형식으로 되어있는 플러그인을 위한 선택적 환경설정 파일
velocity.properties	no	/WEB-INF/classes	기본 Velocity 환경설정 파일 오버라이드

스트럿츠2 구성요소

❑ 태그 라이브러리 (Tag Library)

- 스트럿츠2 태그들은 스트럿츠1.x와 많은 변화
- 최소의 코딩으로 풍부한 기능을 가진 웹 애플리케이션을 만들 수 있는 태그 제공
- 스트럿츠의 태그에서는 OGNL 표현식을 사용
  - 밸류스택에 저장되어있는 다중 객체의 프로퍼티들을 손쉽게 엑세스

스트럿츠2 구성요소

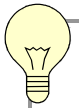
❑ 의존성 주입 (Dependency Injection)

- 객체를 생성해야 하는 책임과 객체 그 자신들과 팩토리와의 연결을 제거
- IoC(Inversion of Control) 컨테이너가 팩토리를 제공
- 내부적으로 프레임워크는 Google Guice 기반의 의존성 주입 컨테이너를 사용
- 스프링(Spring) 플러그인, Plexus 플러그인을 포함한 다른 컨테이너와 함께 애플리케이션을 구축할 수 있도록 플러그인을 사용
- 의존성 주입을 사용하게 되면 애플리케이션을 개발할 때 소스가 간결
- 의존성 주입 컨테이너에 설정 파일에 빈을 등록
- 그 빈을 사용하고자 하는 클래스에서 객체를 선언만 해놓고 사용
- 객체를 생성하는 코드를 작성할 필요가 없어짐

스트럿츠2 구성요소

❑ AJAX 지원

- 스트럿츠2는 애플리케이션에 Ajax기능을 제공하기 위하여 Dojo 프레임워크를 기반으로 한 태그들을 제공
- Ajax 태그를 사용하기 위해서는 "theme" 태그 속성을 "ajax"로 설정
- 또한 ajax 테마를 위한 페이지를 설정하기 위해서는 <s:head/> 태그를 이용하여 테마를 "ajax"로 설정



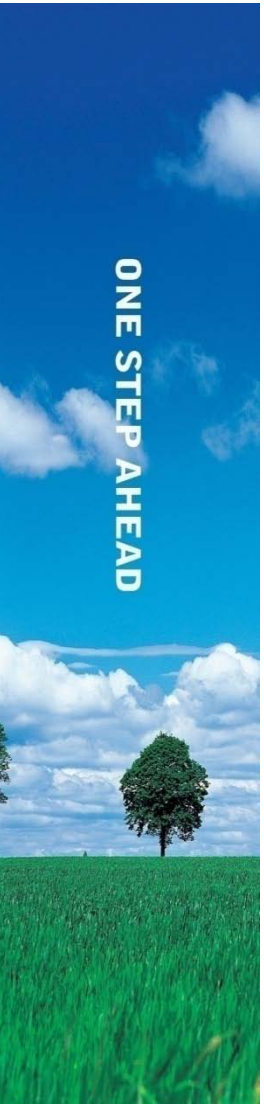
struts-2.1.x 버전부터 struts-tags와 struts-dojo-tags로 분리
library도 **struts2-core-2.1.x .jar**와 **struts-dojo-plugin-2.1.x.jar** 로 분리
상세내용은 부록 참조

스트럿츠2 구성요소

❑ 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

3. Action

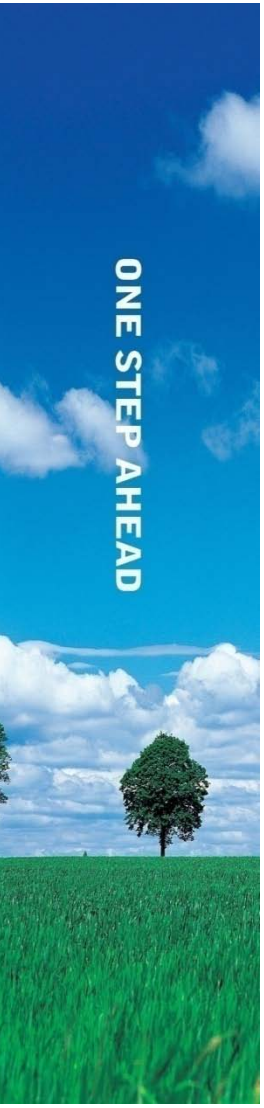


3.1 Action package

3.2 Action 구현

3.3 오브젝트 프로퍼티와 ModelDriven Action을 이용한 데이터전송

3.1 Action package



- ❑ package 선언
- ❑ package 속성

3.1 Action package

❑ package 선언

- 기능 혹은 영역의 일반성에 근거하여 Action을 그룹핑하는 매커니즘 제공
- XML 또는 Java에서 애플리케이션 구조를 선언
- 특별한 URL namespace를 action에 할당 할 수 있음

```
<package name="professor" namespace="/professor"  extends="struts-default">

    <action name="registForm"
                class="kr.nextree.nexschool.action.regsit.RegistProfessorFormAction">
      <result>pages/professor/professor_regist.jsp</result>
    </action>

    <action name="regist"
                class="kr.nextree.nexschool.action.regist.RegistProfessorAction" >
      <result>/common/regist_success.jsp</result>
    </action>

</package>
```

3.1 Action package

❑ 스트럿츠2 package의 속성

- package에서 다음과 같은 4개의 어트리뷰트가 쓰임
- 필수는 name 뿐이며 해당 package를 참조하기 위한 논리명
- namespace가 지정되지 않으면 default namespace 사용
- default namespace는 빈 문자열(empty string) "" 임
- root namespace는 "/" 로 지정
- struts-default 패키지를 상속 받음으로써 프레임워크에서 제공되는 interceptor 를 사용 할 수 있음

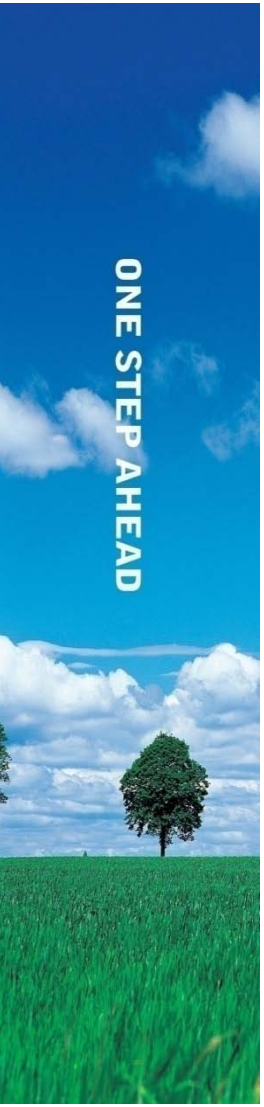
Attribute	설 명
name(required)	패키지명
namespace	패키지내에서의 Action에 대한 namespace
extends	상위 패키지 상속, superclass 패키지의 모든 멤버들을 상속
abstract	true 이면 action이 아닌 상속을 위해 사용

3.1 Action package

❑ 자주 사용되는 스트럿츠 내장 package

```
<package name="struts-default">
        . . .
    <interceptor-stack name="defaultStack">
        <interceptor-ref name="exception"/>
        <interceptor-ref name="alias"/>
        <interceptor-ref name="servlet-config"/>
        <interceptor-ref name="prepare"/>
        <interceptor-ref name="i18n"/>
        <interceptor-ref name="chain"/>
        <interceptor-ref name="debugging"/>
        <interceptor-ref name="profiling"/>
        <interceptor-ref name="scoped-model-driven"/>
        <interceptor-ref name="model-driven"/>
        <interceptor-ref name="fileUpload"/>
        <interceptor-ref name="checkbox"/>
        <interceptor-ref name="static-params"/>
        <interceptor-ref name="params">
            <param name="excludeParams">
                dojo\  ..*
            </param>
        </interceptor-ref>
        <interceptor-ref name="conversionError"/>
        <interceptor-ref name="validation">
            <param name="excludeMethods">
                input,back,cancel,browse
            </param>
        </interceptor-ref>
        <interceptor-ref name="workflow">
            <param name="excludeMethods">
                input,back,cancel,browse
            </param>
        </interceptor-ref>
    </interceptor-stack>
    . . .

    <default-interceptor-ref name="defaultStack"/>
    . . .

</package>
```

3.2 Action 구현



- ❑ 선택적인 Action 인터페이스
- ❑ ActionSupport 클래스
- ❑ 기본적인 유효성 기능
- ❑ 리소스 번들을 이용한 안내 문자 사용

3.2 Action 구현

❑ 선택적인 Action 인터페이스

- 어떤 클래스든 Action이 됨
- Action이 호출될 때 프레임워크에서 호출될 수 있는 메소드 필요
- 프레임워크는 어떤 형태도 요구하지 않으며, 단지 실행 메소드만을 구현하면 됨(action에서 method를 지정하지 않을 경우 default로 execute()를 실행)
- Action 인터페이스 에서는 반환값으로 쓰일 String 상수 제공
- Action 인터페이스의 "success" 상수를 넘기는 execute 메소드를 구현한 디폴트 action을 제공

com.opensymphony.xwork2.Action 인터페이스에는 하나의 메소드만 정의되어 있다.

```
String execute() throws Exception
```

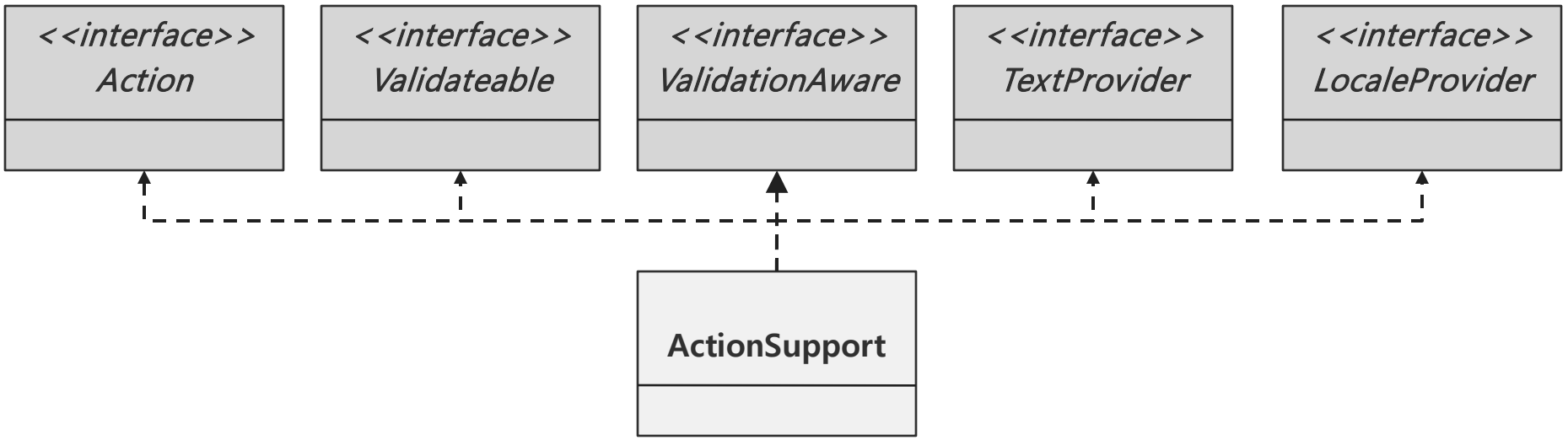
Action 인터페이스는 **execute** 메소드에서 반환값으로 이용되는 유용한 문자 상수를 제공

```
public static final String ERROR "error"
public static final String INPUT "input"
public static final String LOGIN "login"
public static final String NONE "none"
public static final String SUCCESS "success"
```

3.2 Action 구현

❑ ActionSupport 클래스

- validation과 text localization 서비스가 interceptors와 인터페이스의 조합을 통해 제공
- Action은 interceptor에 의해 호출될 메소드만을 가진 인터페이스를 구현
- interceptor는 서비스의 실행을 제어



3.2 Action 구현

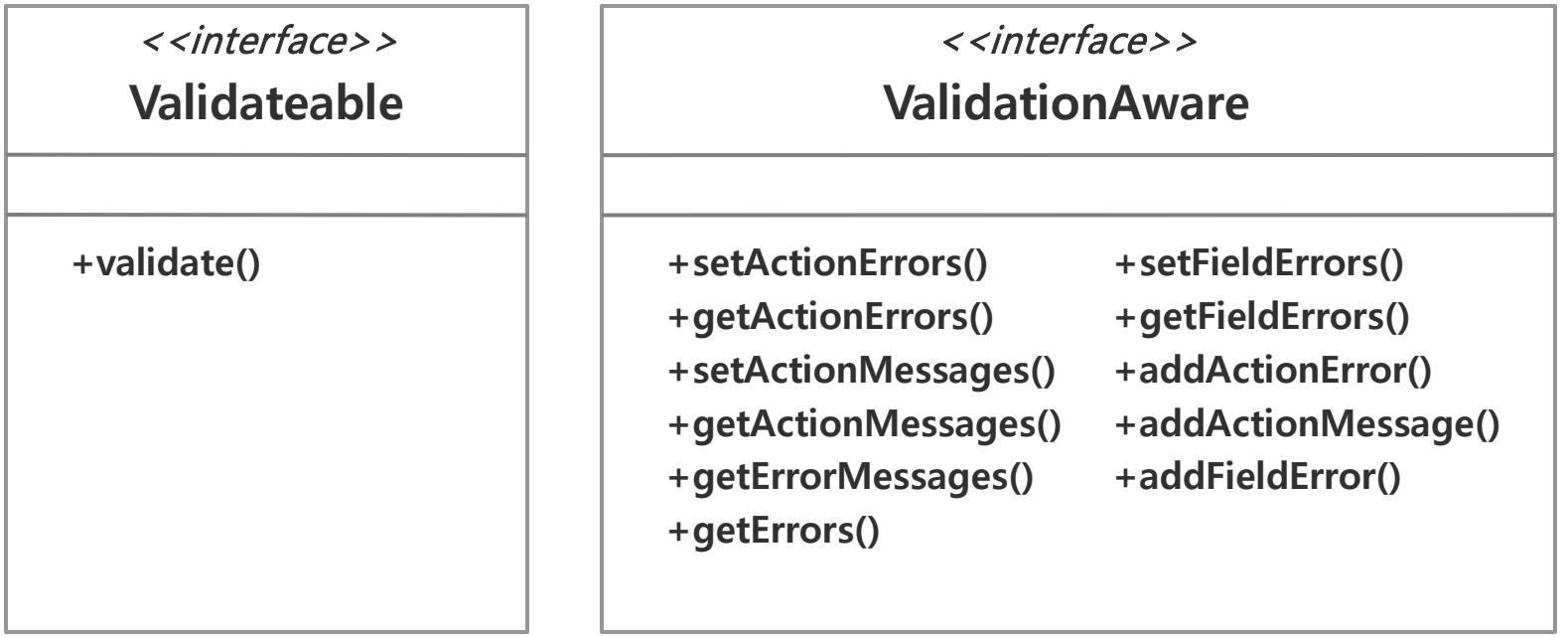
❑ 기본적인 유효성(1)

- ActionSupport는 몇 개의 중요한 서비스 구현을 제공
- ActionSupport를 사용하기 위해서는 Action과 Interceptors의 상호작용을 알아야 함
- Interceptor와 Interface들의 조합을 통해서 Validation과 Text localization services 제공
- Action이 Interceptors에 의해 메소드와 연결되는 동안 Interceptors가 서비스의 실행을 조절
- 스트럿츠2 에서는 풍부하고 수준높은 유효성 설정 프레임워크를 제공
- struts-default 패키지를 확장했다면 ActionSupport를 확장하는 것만으로 유효성을 이용하기 위한 조건 충족

3.2 Action 구현

❑ 기본적인 유효성(2)

- struts-default 패키지로 확장하면 기본적인 Interceptor 스택 상속
- ActionSupport를 사용할 경우 validate() 메서드 구현 필요
- ActionSupport를 사용하지 않을 경우 Validateable, ValidationAware 인터페이스를 구현해야함



3.2 Action 구현

❑ 기본적인 유효성(3)

- workflow interceptor는 crosscutting 영역에서 해당 로직 검색

```
public void validate(){

    if ( this.userId.trim().length() == 0 ){
        addFieldError( "userId", "아이디는 공백일수 없습니다." );
    }

    if (getUserService().isExistUserId(this.userId ) ){
        addFieldError("userId", "이미 존재하는 아이디입니다.");
    }

    if ( this.password.trim().length()  == 0 ){
        addFieldError( "password", "패스워드는 공백일수 없습니다.");
    }

}
```

3.2 Action 구현

❑ 기본적인 유효성(4)

- error 메시지는 com.opensymphony.xwork2.ValidationAware 인터페이스를 통해 설정

error 발생시 collection에 error 저장

```
addFieldError ( String fieldName, String errorMessage )
addActionError ( String errorMessage )
```

workflow interceptor가 에러를 만나게 되면, Control-String(Action.INPUT)을 반환

```
<action name="regist" class="kr.nextree.nexschool.action.regist.RegistProfessorAction">
    <result>/common/regist_success.jsp</result>
    <result name="input" type="redirectAction">
        <param name="namespace">/professor</param>
        <param name="actionName">registFrom</param>
    </result>
</action>
```

3.2 Action 구현

❑ 리소스 번들을 이용한 안내 문자 사용(1)

- 프로퍼티 파일을 생성하여 Action 클래스에 안내문자를 제공
- ActionSupport 는 Localize를 위해 2개의 인터페이스를 구현
  - com.opensymphony.xwork2.TextProvider
    - getText(), getTexts()를 통해 메시지 획득
    - TextProvider는 프로퍼티 파일에서 Key값으로 메시지를 찾음
  - com.opensymphony.xwork2.LocaleProvider
    - ActionSupport는 브라우저에서 보내는 지역 정보를 통해 locale을 가져오도록 인터페이스 구현
    - getLocale() 메소드만 제공
    - TextProvider는 항상 이 locale을 검사하여 적합한 메시지를 반환

RegistProfoessorAction.properties

```
userId.exists=아이디가 이미 존재합니다.
userId.required=아이디는 공백일 수 없습니다.
password.required=패스워드는 공백일 수 없습니다.
```

properties파일에서 한글편집을 지원하는 도구
properties editor : http://sourceforge.jp/projects/propedit/releases/

3.2 Action 구현

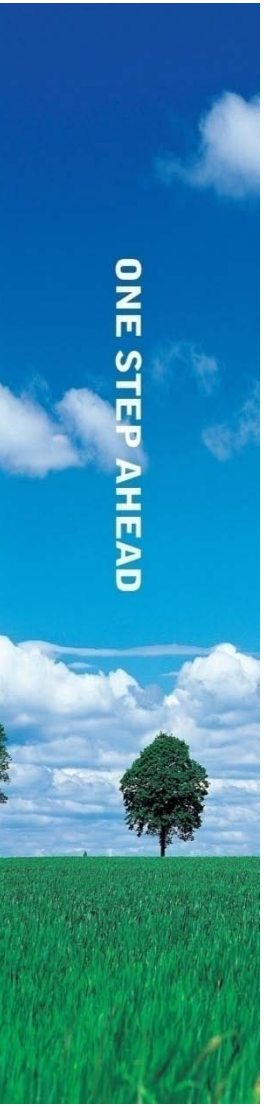
❑ 리소스 번들을 이용한 안내 문자 사용(2)

리소스 번들을 이용한 에러 메시지 사용

```
public void validate(){

    if ( this.userId.trim().length() == 0 ){
        addFieldError( "userId", getText("userId.required"));
    }

    if (getUserService().isExistUser(this.userId ) ){
        addFieldError("userId", getText("userId.exists"));
    }

    if ( this.password.trim().length() <= 0 ){
        addFieldError( "password", getText("password.required"));
    }

}
```

Key값을 이용하여 프로퍼티 파일에 설정되어 있는 메시지를 가져온다

3.3 프로퍼티와 ModelDriven을 이용한 데이터전송



- ❑ 자바빈즈 프로퍼티
- ❑ ModelDriven Action
- ❑ 데이터 전송을 위한 도메인 오브젝트 사용

3.3 프로퍼티와 ModelDriven을 이용한 데이터전송

❑ 자바빈즈 프로퍼티

- 프레임워크에게 오브젝트 자체를 단일 프로퍼티로 제공하면 오브젝트를 생성조차 하지 않아도 됨

자바 빈의 프로퍼티를 action의 프로퍼티로 사용했을 경우(domain object로 매핑 작업 필요)

```
public String execute(){
        Professor prof= new Professor();
        prof.setUserId( userId);
        prof.setPassword(password);
        prof.setName(name);
        prof.setEmail(email);
        prof.setAddress(address);
        getUserService().createUser(prof);
        return "success";
}

private String userId;
private String password;
.....
```

3.3 프로퍼티와 ModelDriven을 이용한 데이터전송

❑ 자바빈즈 프로퍼티

자바 빈 자체를 action의 프로퍼티로 사용하는 경우

```
public String execute(){
    getUserService().createUser(professor);
    return SUCCESS;
}
private Professor professor;
//getters and setters

public void validate(){
    . . .
    if ( getUser().getPassword().length() == 0 ){
        addFieldError( "user.password",  getText("password.required") );
    }
    . . .
}
```

3.3 프로퍼티와 ModelDriven을 이용한 데이터전송

❑ 자바빈즈 프로퍼티

- OGNL EL을 사용하여 객체의 필드에 값을 설정

객체의 프로퍼티를 액션의 필드로 사용할 경우

```
<s:textfield name="name" label="사용자명"/>

<s:property value="name" />님의 가입을 환영합니다.
```

객체자체를 액션의 필드로 사용할 경우

```
<s:textfield name="professor.name" label="이름"/>

<s:property value="professor.name " />님의 가입을 환영합니다.
```

3.3 프로퍼티와 ModelDriven을 이용한 데이터전송

ModelDriven Action(1)

- 도메인 데이터를 제공하는 인터페이스
- getModel() 메소드를 통해 제공, Action에서 구현

```
public class ModelDrivenRegistProfessorAction extends ActionSupport
                                                    implements ModelDriven {
        public String execute(){
                getUserService().createAccount( professor);
                return SUCCESS;
        }

        private Professor professor= new Professor();

        public Object getModel() {
                return professor;
        }

        . . .
}
```

3.3 프로퍼티와 ModelDriven을 이용한 데이터전송

ModelDriven Action(2)

- execute() 메소드안에서 도메인 오브젝트의 필드값을 바꾸는 일은 권장 않음
- 데이터 불일치를 발생 시킬 수 있음

```
public String execute(){
        professor = new Professor();
        user.setSomething();
        getUserService().createUser ( professor );
        return SUCCESS;
}
private Professor professor= new Professor();
public Object getModel() {
        return professor;
}
```

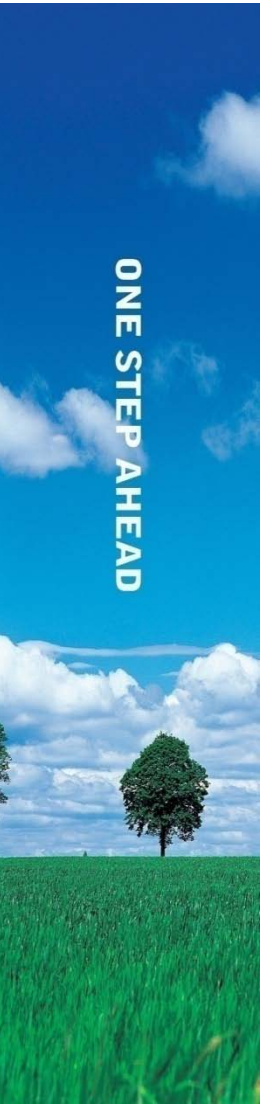
```
<s:textfield name="name" label="사용자명"/>

 <s:property value="name" /> 님의 가입을 환영합니다.
```

3.3 프로퍼티와 ModelDriven을 이용한 데이터전송

❑ 데이터전송을 위한 도메인 오브젝트 사용

- 도메인 오브젝트 사용시 속성값이 노출되는 문제점이 있음
- 사용자가 queryString에 id값을 셋팅하여 넘길 경우 자동적으로 이 값이 도메인 오브젝트에 셋팅
- id 속성을 오브젝트로부터 제거할 수도 있지만 이는 오브젝트의 재사용성을 상실
- Action을 만들때 이점을 숙지

4. Results



4.1 기본 Result 타입
4.2 기타 Result 타입
4.3 전역 Result 선언
4.4 Results 실습

4.1 기본 Result 타입

❑ 기본 Result 타입(1)

- struts-default.xml 에 다수의 기본 result 타입이 선언되어 있음
- 기본 result 타입 사용을 위해선 package가 struts-default를 상속하여야 함

struts-default.xml의 freemarker result 타입 선언

```
<result-type name="freemarker"
    class="org.apache.struts2.views.freemarker.FreemarkerResult" />
```

struts-default를 상속한 package에서 result 타입 사용

```
<package name="some" namespace="/some" extends="struts-default">

    <action name="someAction" class="example.someAction">
        <result type="freemarker">/some/someAcion.jsp</result>
    </action>

</package>
```

4.1 기본 Result 타입

❑ 기본 Result 타입(2)

Result 타입	용도	파라미터
dispatcher	JSP, Servlet등의 웹 애플리케이션	location(default), parse
redirect	다른 URL로의 redirect	location(default), parse
redirectAction	다른 Struts2 action으로 redirect	actionName(default), namespace, 기타 파라미터

8.1 기본 Result 타입

❑ RequestDispatcher(dispatcher)

- action에 대한 result로 JSP page로 렌더링 할때 사용
- struts-default.xml의 struts-default package에 선언
- struts-default를 상속한 package를 사용할때는 DispatcherResult를 기본형으로 사용함.(package당 한개의 default result 타입만 허용)
- javax.servlet.RequestDispatcher를 사용

struts-default.xml의 dispatcher result 타입 선언

```
<result-type name="dispatcher"
    class="org.apache.struts2.dispatcher.ServletDispatcherResult"
    default="true" />
```

dispacher result 타입 용법

```
<action name="home"
            class="kr.nextree.nexschool.action.professor.ProfessorManageraction">
    <result type="dispatcher">/pages/professor/professor_home.jsp</result>
</action>
```

4.1 기본 Result 타입

❑ 다른 서블릿으로 Forward

- action에 대한 result type으로 servlet을 사용이 가능
- servlet result도 dispatcher를 사용하므로 result 태그의 설정은 동일
- servlet에서는 ActionContext를 통해 ValueStack에 접근

AnotherServlet.java

```
public class AnotherServlet extends HttpServlet{
   protected void doPost(HttpServletRequest req, HttpServletResponse res) throws
                    ServletException, IOException{

      PrintWriter out = res.getWriter();
      ...
      ValueStack valueStack = ActionContext.getContext().getValueStack();
      String someProperty = valueStack.findValue("someProperty");

      out.println(someProperty);
      ....
   }
}
```

4.1 기본 Result 타입

ForwardToAnotherServlet.java

```
public class ForwardToAnotherServlet extends ActionSupport
                    implements RequestAware{

   public String execute(){
      getRequest().put("someAttribute", "some attribute from request attribute set.");
      return SUCCESS;
   }

   private Map request;
   .... getter and setter ....

   public String getSomeProperty(){ return "some property from action.";}
}
```

4.1 기본 Result 타입

result 선언 – 기본값 사용

```
<action name="ForwardToAnotherServlet"
                class="chapter8.ForwardToAnotherServlet">

    <result>/anotherServlet</result>

</action>
```

result 선언 – 기본값 미사용

```
<action name="ForwardToAnotherServlet"
                class="chapter8.ForwardToAnotherServlet">

    <result type="dispatcher">
            <param name="location">/anotherServlet</param>
            <param name="parse">true</param>
    </result>

</action>
```

4.1 기본 Result 타입

ServletRedirectResult(redirect)

- 현재 request를 종료하고 새로운 요청을 보내도록 브라우저에 응답
- request와 response가 유지되지 않기 때문에 이전 action에서 값을 참조 할 수 없음
- redirect에서도 parameter를 유지하기 위해서는 session에 저장하거나, redirect url 뒤에 querystring으로 전달하는 방법이 있음
- dispatcher가 server-side작업인것에 비해, redirect는 client-side작업임

struts-default.xml의 redirect result 타입 선언

```
<result-type name="redirect"
    class="org.apache.struts2.dispatcher.ServletRedirectResult" />
```

redirect result 타입 용법

```
<action name="SendUserToSearchEngineAction" class="myActionClass">
    <result type="redirect">http://www.google.com</result>
</action>
```

4.1 기본 Result 타입

OGNL escape를 이용한 동적 **location**설정

```
<action name="SendUserToSearchEngineAction" class="myActionClass">
    <result type="redirect">
        http://www.google.com?myParam=${defaultUsername}
    </result>
</action>
```

OGNL escape는 '%'를 사용하지만, struts-xml과 다른 xml문서에서는 '$'를 사용하며 OGNL escape는 ValueStack을 참조한다.

4.1 기본 Result 타입

ServletActionRedirectResult(redirectAction)

- action으로 redirect를 하는 것을 제외하고는 redirect와 동일
- param 를 url에 querystring으로 전달 가능

```
<action name="Login" class="chapter8.Login">
    <result type="redirect">/chapter8/secure/AdminPortfolio.action</result>
    <result name="input">/chapter8/Login.jsp</result>
</action>
```

```
<action name="Login" class="chapter8.Login">
    <result type="redirectAction">
        <param name="actionName">AdminPortfolio</param>
        <param name="namespace">/kosta/chapter8/secure</param>
        <param name="param1">hardCodedValue</param>
        <param name="param2">${someProperty}</param>
    </result>
    <result name="input">/chapter8/Login.jsp</result>
</action>
```

Struts 2.1.x – actionName과 namespace를 제외한 나머지 param 엘리먼트는 인식되지 않으므로, 파라미터 전달을 위해서는 actionName에 쿼리스트링 형식으로 전달하여야 한다.

AdminPortfolio?**param1=hardcodedValue&param2=${someProperty}**

4.2 기타 Result 타입

❑ VelocityResult(velocity)

- 뷰 레이어 페이지 렌더링시에 동적 데이터 삽입에 유용한 경량 템플릿
- 태그로 OGNL을 이용하여 ActionContext와 ValueStack의 값을 접근

struts-default.xml의 velocity result 타입 선언

```
<result-type name="velocity"
    class="org.apache.struts2.dispatcher.VelocityResult" />
```

velocity result 타입 용법

```
<action name="ViewPortfolioVM" class="chapter8.ViewPortfolio">
    <result type="velocity">/chapter8/ViewPortfolio.vm</result>
</action>
```

4.2 기타 Result 타입

ViewPortfolio.vm

```
<html>
  <head>
    <title>view Portfolio</title>
  </head>
  <body>
    <h5>
      이 것은 #sproperty ("value=username") 님의 포트폴리오
      #sproperty ("value=portfolioName") 입니다.
    </h5>
    <a href='#surl ("action=PortfolioHomePage")'>Home</a>
  </body>
</html>
```

4.2 기타 Result 타입

❑ FreemarkerResult(freemarker)

- Velocity templates와 유사항 성능을 보유
- Velocity와 마찬가지로 OGNL을 통해 ValueStack의 data에 접근
- 원본 요청의 response stream에 직접 접근 가능

struts-default.xml의 freemarker result 타입 선언

```
<result-type name="freemarker"
    class="org.apache.struts2.views.freemarker.FreemarkerResult" />
```

velocity result 타입 용법

```
<action name="ViewPortfolioFM" class="chapter8.ViewPortfolio">
    <result type="freemarker">/chapter8/ViewPortfolio.ftl</result>
</action>
```

4.2 기타 Result 타입

ViewPortfolio.ftl

```
<html>
  <head>
    <title>view Portfolio</title>
  </head>
  <body>
    <h5>
      이 것은 <@s.property value="username" /> 님의 포트폴리오
      <@.property value="portfolioName") 입니다.
    </h5>
    <a href='<@s.url ("action=PortfolioHomePage")'>Home</a>
  </body>
</html>
```

4.3 전역 Result 선언

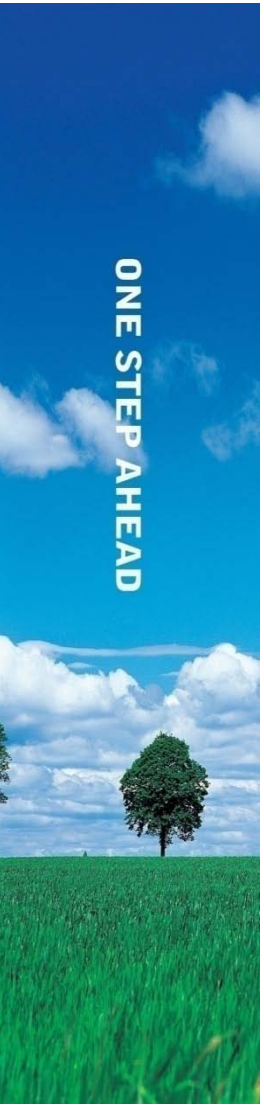
❑ global-results

- 동일한 package의 모든 action에서 같은 타입의 result를 사용시
- action에 해당 result가 정의되어 있지 않을 경우 전역 result 영역 참조
- action 엘리먼트보다 먼저 정의되어야 함.

struts.xml

```
<package name="chapter8" namespace="/chapter8" extends="struts-default">

   <global-results>
      <result name="error">/common/Error.jsp</result>
   </global-results>

   <action name="ViewPortfolio" class="chapter8.ViewPortfolio">
      <result type="velocity">/chapter8/ViewPortfolio.jsp </result>
   </action>

</package>
```

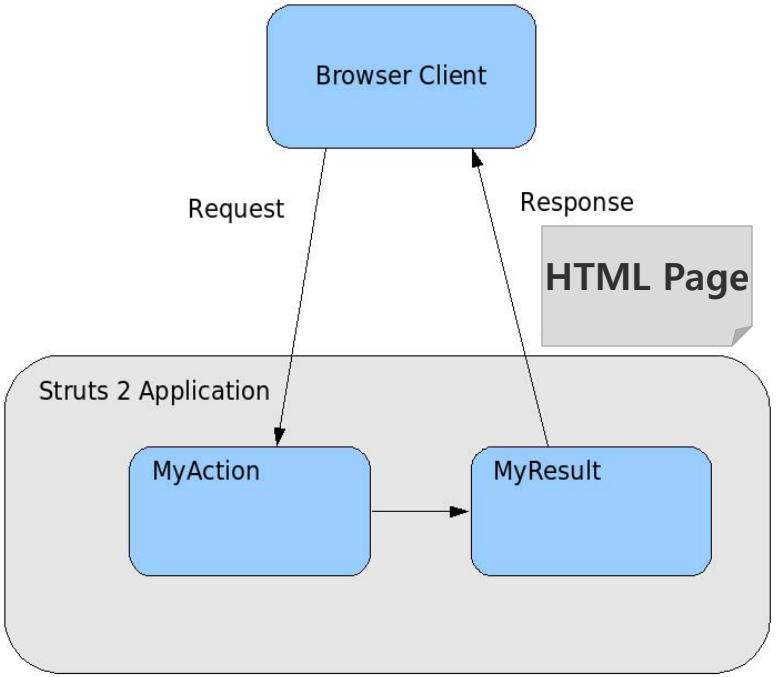
4.4 Results 실습

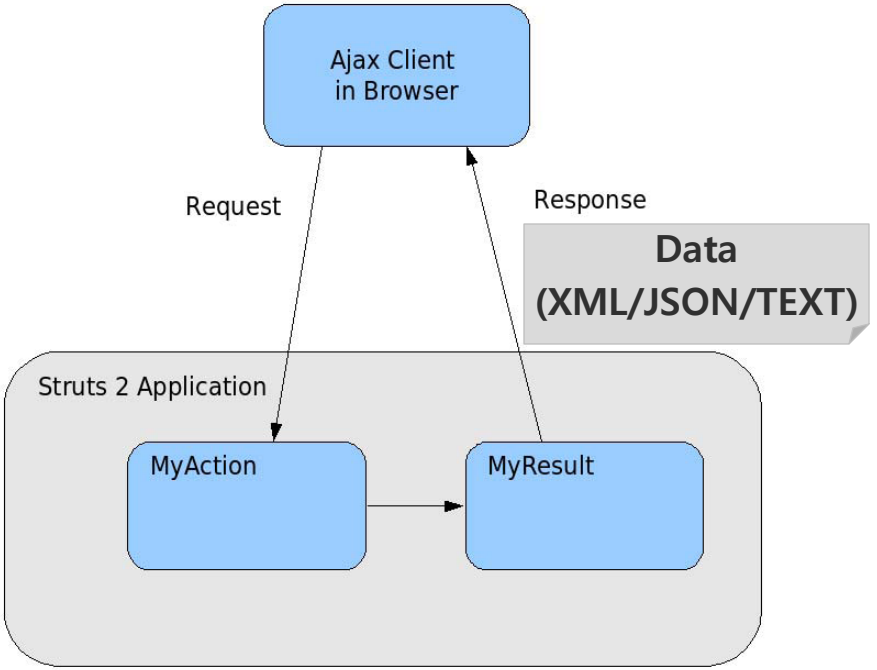


- ❑ Ajax 애플리케이션을 위한 custom results 작성
- ❑ JSON result type 구현

Ajax 애플리케이션을 위한 custom results 작성

❑ Request 처리 구조





Ajax 애플리케이션을 위한 custom results 작성

❑ JSONResult 생성

- com.opensymphony.xwork2.Result interface를 구현

```
public interfact Result extends Serializable{
        public void execute(ActionInvocation invocation) throws Exception;
}
```

```
public JSONResult implements Result{

    public void execute(ActionInvocation invocation) throws Exception{
        ServletActionContext.getResponse().setContentType("text/plain");
        PrintWriter out = ServletActionContext.getResponse().getWriter();

        ValueStack valueStack = invocation.getStack();
        Object jsonModel = valueStack.findValue("jsonModel");

        XStream xstream = new XStream(new JettisonMappedXmlDriver());
        out.println(xstream.toXML(jsonModel));
    }
    ...
}
```

Ajax 애플리케이션을 위한 custom results 작성

JSON Result

```
{
   “artist” : {
      “username” : “Mary”,
      “password” : “max”,
      “portfolioName” : “Mary’s Portfolio”,
      “firstName” : “Mary”,
      “lastName” : “Greene”,
      “protfolios” : {
         “Oil Painting” : {“name” : “noname No1”,“owner” : “Mary”},
         “Wood cuts”:  {“name” : “woodcuts No1”, “owner”: “Mary”}
      }
}
```

Ajax 애플리케이션을 위한 custom results 작성

❑ Action 구현

- Ajax를 위한 액션도 일반 액션과 동일

```
public class ProfessorManagerAction extends ActionSupport {

    public String execute(){
        User user = getUserService.find(selectedProfessorId);
        jsonModel = user;
        return SUCCESS;
    }

    private long selectedProfessorId;
    private Object jsonModel;

    ... getters and setters...
}
```

Ajax 애플리케이션을 위한 custom results 작성

❑ JSONResult 를 위한 custom result type 선언

- JSONResult를 위한 result type은 기본값이 아님.
- package내에 custome result type 선언 필요

```
<package name="professor" namespace="/professor">
  <result-types>
    <result-type name="customJSON"
                 class="kr.nextree.nexschool.result.JSONResult" />
  </result-types>

  <action name="findPrfoessor"
                 class="kr.nextree.nexschool.action.professor.ProfessorManagerAction"
                 method ="find">
    <result type="customJSON">artist</result>
  </action>
  ....
</package>
```