

Anyframe Struts Plugin



Version 1.0.2

저작권 © 2007-2011 삼성SDS

본 문서의 저작권은 삼성SDS에 있으며 Anyframe 오픈소스 커뮤니티 활동의 목적하에서 자유로운 이용이 가능합니다. 본 문서를 복제, 배포할 경우에는 저작권자를 명시하여 주시기 바라며 본 문서를 변경하실 경우에는 원문과 변경된 내용을 표시하여 주시기 바랍니다. 원문과 변경된 문서에 대한 상업적 용도의 활용은 허용되지 않습니다. 본 문서에 오류가 있다고 판단될 경우 이슈로 등록해 주시면 적절한 조치를 취하도록 하겠습니다.

I. Introduction	1
II. Struts	2
1. Architecture	3
1.1. Controller Structure	3
1.2. Request의 흐름	3
2. Configuration	4
2.1. web.xml	4
2.1.1. servlet, servlet-mapping 설정	4
2.1.2. taglib 설정	5
2.2. struts-config.xml	6
2.2.1. controller	6
2.2.2. message-resources	7
2.2.3. plug-in	8
2.2.4. form-beans	9
2.2.5. action-mappings	10
2.2.6. global-forwards	12
3. Controller	14
3.1. ActionServlet	14
3.1.1. ActionServlet의 역할	14
3.1.2. 초기화 프로세스	14
3.1.3. 실행 시(ActionServlet 인스턴스가 HTTP Request를 받을 때)	14
3.1.4. ShutDown 프로세스	14
3.2. RequestProcessor	15
3.2.1. RequestProcessor의 역할	15
3.2.2. process() 메소드의 Request 처리 절차	15
3.2.3. Sample	16
3.3. Action	16
3.3.1. Action의 역할	16
3.3.2. Action의 구현	16
3.3.3. Sample	16
3.4. ActionForward	18
3.4.1. ActionForward의 역할	18
3.5. Actions Package	18
3.5.1. org.apache.struts.actions 패키지에 미리 정의되어 있는 Action	18
3.5.2. org.apache.struts.actions.ForwardAction	19
3.5.3. org.apache.struts.actions.IncludeAction	19
3.5.4. org.apache.struts.actions.DispatchAction	19
3.5.5. org.apache.struts.actions.LookupDispatchAction	20
3.5.6. org.apache.struts.actions.SwitchAction	20
4. View	22
4.1. Taglib	25
4.1.1. Taglib의 특징	25
4.1.2. Struts Taglib	26
4.1.3. JSP Standard Tag Library	29
4.1.4. 기타 Taglib	30
4.2. Tiles	30
4.2.1. Page Layout 구성 방법	31
4.2.2. Tiles 설치	31
4.2.3. Tiles 사용	32
4.2.4. Tiles Layout 정의	32
5. Internationalization	34
5.1. Internationalization의 특징	34
5.1.1. Internationalization의 필요성	34
5.1.2. 지역 (Locale)	34
5.2. Internationalization Sample	35

5.2.1. Sample	35
6. Validator	37
6.1. Plug-in 등록	37
6.1.1. struts-config.xml에 plug-in 등록	37
6.1.2. Samples	37
6.2. Validator Rules	37
6.2.1. Struts Validator Rules 기본 기능	37
6.3. ActionForm	38
6.3.1. ValidatorForm의 상속	38
6.3.2. Samples	38
6.4. formset 설정	39
6.4.1. formset 설정 방법	39
6.4.2. Sample	39
6.5. Action 매핑 설정	40
6.5.1. struts-config.xml의 Action 매핑 설정	40
6.5.2. Sample	40
7. Exception Handling	41
7.1. Global Level Exception Handling	41
7.1.1. Global Level Exception Handling의 특징	41
7.1.2. Samples	41
7.2. Action Level Exception Handling	42
7.2.1. Action Level Exception Handling의 특징	42
7.2.2. Samples	42
III. Struts Extensions	43
8. Controller	44
8.1. DefaultActionServlet	44
8.2. DefaultRequestProcessor	44
8.2.1. DefaultRequestProcessor 기능	45
8.3. AbstractActionSupport	46
8.3.1. Action Sample	47
8.4. DefaultDispatchActionSupport	48
8.4.1. Action Sample	48
8.5. DefaultForwardAction	48
8.6. AnyframeMiPAction	48
8.6.1. Sample Action	49
9. View	50
9.1. Tag library	50
9.1.1. Page Navigator Tag	50
10. Preventing Double Form Submission	51
10.1. Double Submit의 개념	51
10.2. 일반적인 Token 처리	51
10.3. 선언적인 Token 처리	52
10.3.1. Samples	52
10.3.2. 참고 사항	53
11. Exception Handling	55
11.1. 선언적인 Exception Handling	55
11.1.1. Samples	55
11.2. DefaultBaseExceptionHandler 확장	56
12. Authentication and Authorization	58
12.1. Authentication	58
12.1.1. Samples	58
12.2. Authorization	60
12.2.1. 접근 권한 제어 프로세스	60
12.2.2. Samples	61
13. Spring Integration	62

13.1. Configuration	62
13.1.1. ContextLoaderListener, ContextConfigLocation 정의	62
13.2. Action	62

I.Introduction

Struts plugin은 J2EE 어플리케이션 개발 시 Web framework로 많이 사용되고 있는 Apache Struts [<http://struts.apache.org/>]와 Spring 간의 연계 방법을 가이드하기 위한 샘플 코드와 이 오픈 소스들을 활용하는데 필요한 참조 라이브러리들로 구성되어 있다.

Installation

Command 창에서 다음과 같이 명령어를 입력하여 struts plugin을 설치한다.

```
mvn anyframe:install -Dname=struts
```

installed(mvn anyframe:installed) 혹은 jetty:run(mvn clean jetty:run) command를 이용하여 설치 결과를 확인해볼 수 있다.

Dependent Plugins

Plugin Name	Version Range
Query [http://dev.anyframejava.org/docs/anyframe/plugin/optional/query/1.1.2/reference/htmlsingle/query.html]	2.0.0 > *

II.Struts

Apache Struts Framework는 Java web application을 개발하기 위한 open-source framework이다. Struts는 software application의 separate concerns중 하나인 Model-View-Controller(MVC) architecture를 기반으로 web application을 개발할 수 있도록 도와주고 있다. Struts에서는 Controller를 ActionServlet형태로 제공하고, JSP Taglib를 사용하여 View 레이어를 구현하도록 가이드하고 있다. 또한 Spring Framework의 Web Struts에서는(spring-webmvc-struts.jar), Spring Framework을 기반으로 비즈니스 레이어를 구성할 경우 Struts의 WebApplicationContext에 Spring Bean으로 등록된 비즈니스 레이어의 인터페이스에 접근할 수 있는 환경을 제공한다.

Struts의 특징은 다음과 같다.

- **MVC architecture**를 따르고 있기 때문에 역할 분리가 명백하다.

Model과 View를 연결하는 Controller인, ActionServlet은 입력된 HTTP Request에 따라 Action클래스를 실행하고, Action클래스는 Model에 해당하는 비즈니스 레이어나 Database관련 로직을 수행한다. 그리고 Model과 View사이의 data 전달을 위한 ActionForm 클래스를 활용할 수 있다.

- **JSP**로 구현하는 **View** 개발을 편리하게 도와주는 **Taglib**를 제공한다.

Struts에서 기본으로 제공하는 Taglib에는 Bean, HTML, Logic, Nested가 있다.

- **Configuration** 설정으로 **Exception Handling**이 가능하다.

struts-config.xml파일의 exception handling을 이용하여 Exception종류에 따른 Exception page를 간편하게 설정할 수 있다.

- **QConfiguration** 설정으로 권한처리가 가능하다.

특정 URL에 권한을 부여하여 허가된 사용자만이 해당 URL에 접근이 가능하도록 설정할 수 있다.

- **Controller**에서 **Validation Check**가 가능하다.

사용자의 입력값을 View가 아닌 Controller에서 Validation Check를 할 수 있다.

- **MessageResource**를 이용한 국제화(**I18N**)기능을 지원한다.

사용자 Locale정보에 따라 다양한 언어로 web page 출력이 가능하다

2.Configuration

Struts를 사용하기 위해서는 기본적인 환경 설정이 필요하다. 먼저 웹 어플리케이션의 배포지시자인 web.xml 파일에 <servlet>설정 등이 필요하고 Struts의 Controller가 어떤 Action을 실행할 것인지 어떤 화면으로 이동할 것인지 등에 대한 설정 정보를 설정하는 struts-config.xml이 필요하다. 본 문서에서는 Struts를 사용해 웹 어플리케이션을 개발하기 위한 최소 조건과 Struts에서 많이 사용하는 기능에 대한 환경 설정에 대해서 중점적으로 다루겠다.

2.1.web.xml

웹 어플리케이션의 배포 지시자(Deployment Descriptor)로 Java EE 환경에서 웹 어플리케이션이 어떻게 배포되어야 하는지를 기술하는 파일이다. XML 구문으로 기술되며 웹 어플리케이션 root 바로 아래 서브 디렉토리인 WEB-INF에 위치한다. 본 문서에서는 ActionServlet을 확장한 Anyframe의 DefaultActionServlet을 사용 할 경우 web.xml 작성법을 중심으로 설명하겠다.

ActionServlet에 대한 내용은 Apache Struts User Guide [<http://struts.apache.org/1.3.10/userGuide/configuration.html>]를 참조하기 바란다.

- servlet, servlet-mapping 설정
- taglib 설정

2.1.1.servlet, servlet-mapping 설정

2.1.1.1.<servlet>설정

- org.apache.struts.action.ActionServlet 또는 서브 클래스를 <servlet-class>에 등록
- 해당 Servlet을 통해 로드되어 할 Struts 속성 정의 파일 목록 정의
- Servlet 초기화에 필요한 속성 정의

다음은 <servlet>하위에 <init-param>으로 정의할 수 있는 초기화 파라미터들이다.

Name	Purpose / Default Value
config	Default application의 struts 설정 파일이 있는 상대(module-relative)경로를 나타낸다. Default value는 /WEB-INF/struts-config.xml이다.
config/sub1	Sub-application을 사용할 때, config/와 sub-application의 prefix를 사용하여 기술한다. 예로, hello라는 prefix를 가진 sub-application이 있다면, config/hello라는 이름으로 <init-param>을 기술해야 한다. 또한, sub-application이 여러 개 있다면 그 각각에 대해서 <init-param>을 기술해야 한다. (여기서, config와 config/sub1, config/sub2 등을 각각 하나의 module이라고 부른다.)
convertNull	강제로 forms의 property 값들을 null로 populate한다. 예를 들어, convertNull 값이 true이면, java.lang.Integer type의 property들은 디폴트 값으로 0이 아닌 null이 셋팅된다. convertNull의 디폴트 값은 false이다.
chainConfig	action에서 definition명으로 forwarding할 때 설정을 해야 한다. Struts 1.3에서 새롭게 추가된 내용이므로 Struts Tiles를 이용할 시 주의해야 한다. 디폴트 값은 org/apache/struts/chain/chain-config.xml이다.
configFactory	ModuleConfig interface의 implementation을 생성하기 위한 ModuleConfigFactory의 Java class name이다. (Struts 1.3 이상)

Name	Purpose / Default Value
debug	Servlet의 Logging 수준을 결정한다. 디폴트 값은 0이고, 0~6 사이의 수를 넣을 수 있다. 6이 가장 많은 양의 logging 정보를 출력할 것이다.

2.1.1.2. <servlet-mapping> 설정

- Request의 URL 패턴을 <servlet>과 매핑

2.1.1.3. Samples

다음은 Struts를 기반으로 구성된 웹 어플리케이션의 web.xml 파일에서 <servlet>, <servlet-mapping>을 설정한 예제이다.

*.do URL 패턴의 request에 대하여 front controller 서블릿인 ActionServlet이 처리하도록 설정되어 있으며, 초기화 파라미터로 해당 웹 어플리케이션에서 사용하는 struts-config 설정 파일이 복수개로 등록되어 있다.(','로 복수개 설정) convertNull의 초기화 파라미터 설정도 확인할 수 있다. load-on-startup 설정은 서블릿 엔진이 시작될 때 로드될 우선 순위를 지정한 값이다.

```

<servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
  <init-param>
    <param-name>config</param-name>
    <param-value>
      /config/struts/struts-config.xml,
      /config/struts/struts-config-login.xml
    </param-value>
  </init-param>
  <init-param>
    <param-name>debug</param-name>
    <param-value>3</param-value>
  </init-param>
  <init-param>
    <param-name>convertNull</param-name>
    <param-value>true</param-value>
  </init-param>
  <load-on-startup>0</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>action</servlet-name>
  <url-pattern>*.do</url-pattern>
</servlet-mapping>

```

2.1.2. taglib 설정

2.1.2.1. JSP에서의 설정

Struts 1.3에서는 Servlet 2.3과 2.4 모두 tag library의 설정이 간단해 졌다. struts-taglib.jar파일을 /WEB-INF/lib에 복사한 후 아래 sample과 같이 사용할 tag에 대한 정의를 JSP에 추가하면 된다.

2.1.2.2. Samples

다음은 'bean', 'html', 'logic' tag library를 사용하는 경우 JSP에 어떻게 정의해야 하는지를 보여주는 예제이다.

```
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic"%>
```

2.2.struts-config.xml

Struts 기반의 웹 어플리케이션을 위한 배포 지시자로 Model, View, Controller를 함께 연결시켜 주는 주된 설정 파일이다. 본 문서에서는 자주 사용하는 attribute들에 대해서만 소개하고 자세한 내용은 Apache Struts User Guide [<http://struts.apache.org/1.3.10/userGuide/configuration.html#struts-config>] 와 struts-config online DTDDoc [http://struts.apache.org/1.3.10/struts-core/dtdoc/struts-config_1_3.dtd.html] 를 참고한다.

- 코드에서 설정 정보 분리
- Struts DTD : struts-config_1_3.dtd를 준수
- Action, ActionForm, JSP 간의 매핑 정보 저장
- 하나 이상의 configuration file 사용 가능
- 서브 모듈 별 설정 파일 정의 가능
- Struts configuration 파일의 설정 요소

2.2.1.controller

2.2.1.1.<controller>설정

- Request를 받은 ActionServlet은 실제 처리를 RequestProcessor에게 위임함
- 각각의 sub-application에 대한 controller를 다르게 설정하여 모듈별로 분리할 수 있음
- org.apache.struts.action.RequestProcessor 또는 서브 클래스를 <controller>에 등록

다음은 <controller>의 attribute들이다.

Name	Description
bufferSize	파일 업로드 시 사용하는 입력 버퍼의 크기. 이 속성은 선택적으로 사용할 수 있다. 디폴트 값 : 4096
className	컨트롤러 정보를 포함한 Config 빈을 구현하는 클래스의 full name. org.apache.struts.config.ControllerConfig를 상속받은 클래스이어야 한다. 디폴트 값 : org.apache.struts.config.ControllerConfig
contentType	Response 결과를 보낼 때 사용하는 콘텐츠 타입. 이 속성은 선택적으로 사용. 이 속성에 지정한 값이 있더라도 액션이나 JSP 페이지에서 지정한 콘텐츠 타입이 우선한다. 디폴트 값 : text/html
forwardPattern	/로 시작하는 (contextRelative 속성이 false일 경우) context-relative URL과 <forward>의 'path'속성이 어떻게 매핑되는지에 대한 대체 패턴. 이 값은 아래 항목의 조합으로 이루어진다. 디폴트 값 : \$M\$P <ul style="list-style-type: none"> • \$M – Module의 prefix값으로 대체 • \$P – 선택된 <forward>의 'path' 속성값으로 대체

Name	Description
	<ul style="list-style-type: none"> • \$\$ - URL 에서 \$ 표시 그대로 표현됨 • \$x(x는 위에 정의되지 않은 다른 문자) - 추후 사용을 위해 예약됨
inputForward	<action>의 input attribute를 최종 URL로 사용할 forward 또는 global-forward의 이름이길 원하면 true로 설정한다. false 이면 sub application의 상대경로로 간주한다. 디폴트 값 : false
locale	사용자의 locale정보를 사용자 세션에 저장할지 여부 디폴트 값 : true
maxFileSize	파일 업로드 시 파일의 최대 용량으로 K,M,G 단위를 붙여 사용한다. 디폴트 값 : 250M
multipartClass	Multipart 요청에 대한 처리를 담당하는 클래스의 전체 이름. 파일 업로드 시 사용. 디폴트 값 : org.apache.struts.upload.CommonsMultipartRequestHandler
nocache	Response에서 HTTP 헤더를 'nocache'로 설정 할 지에 대한 Boolean 값. Optional 디폴트 값 : false
pagePattern	<p>커스텀 태그를 사용하는 페이지의 page 속성이 컨텍스트 상대 URL 경로에 매핑 방법에 대한 대체 패턴. 이 속성의 값은 아래 항목들로 구성된다.</p> <p>디폴트 값 : \$M\$P</p> <ul style="list-style-type: none"> • \$M - Module 접두어에 의한 대체 • \$P - 선택한 forward 요소의 path 속성에 의한 대체 • \$\$ - URL 에서 \$ 표시 그대로 표현됨 • \$x(x는 위에 정의되지 않은 다른 문자) - 추후 사용을 위해 예약됨
processorClass	사용자의 요청을 처리할 클래스의 full name. 여기에 지정한 클래스는 org.apache.struts.action.RequestProcessor 클래스의 하위 클래스가 된다. 디폴트 값 : org.apache.struts.chain.ComposableRequestProcessor
tempDir	파일 업로드 시 사용할 임시 디렉터리. Optional 디폴트 값 : Servlet container가 제공하는 디렉터리

2.2.1.2.Samples

다음은 struts-config.xml 파일에서 controller 설정에 대한 예제이다.

```
<controller
    contentType="text/html;charset=utf-8" locale="true" nocache="true"
    processorClass="org.apache.struts.action.RequestProcessor"/>
```

2.2.2.message-resources

2.2.2.1.<message-resources>설정

- 메시지 리소스 번들과 관련된 사항들을 정의
- 국제화 지원(I18N)
- Application에 여러 개의 <message-resources>를 등록해 사용 할 수 있음
- 개별 Module은 각각이 사용하는 resource bundle을 정의할 수 있음

- Application에 여러 개의 resource bundle을 사용하려고 할 때는 key값을 설정해 줘야함

다음은 <message-resources>의 attribute들이다.

Name	Description
className	message-resources의 정보를 포함하는 설정 빈을 구현하는 클래스 이름. 여기에 지정한 클래스는 org.apache.struts.config.MessageResourcesConfig 클래스의 하위 클래스여야 한다. Optional. 디폴트 값 : org.apache.struts.config.MessageResourcesConfig
factory	사용할 MessageResourcesFactory 클래스의 이름. 디폴트 값 : org.apache.struts.util.PropertyMessageResourcesFactory
key	메시지 리소스 번들이 저장될 ServletContext 속성. 이 속성은 optional. 디폴트 값 : org.apache.struts.action.MESSAGE
null	정의되지 않은 메시지 키가 사용된 경우 이를 MessageResources의 하위 클래스에서 어떻게 처리할지 나타내는 Boolean 값. false로 설정하면 null이 아닌 "???keyname???" 과 같은 문자열을 되돌려준다. 디폴트 값 : true
parameter	리소스 번들의 이름. 예를 들어, 리소스 번들 이름이 ApplicationResources. 이 속성의 값은 ApplicationResources가 된다. 이 속성은 반드시 정의해야 한다. 만일 리소스 번들이 패키지가되어 있다면 해당 패키지의 이름을 포함한 전체 이름을 지정해야 한다.

2.2.2.2.Samples

다음은struts-config.xml 파일에서 message-resources 설정에 대한 예제이다.

```
<message-resources
    parameter="message.message-productmgmt"/>
```

2.2.3.plug-in

2.2.3.1.<plug-in>설정

- Struts Application 이 구동 시에 동적인 자원을 처리하게 해주는 강력한 기능임
- org.apache.struts.action.PlugIn 인터페이스를 구현하는 자바 클래스를 생성한 후 설정 파일에 plug-in 요소를 추가하여 사용 가능
- plug-in 에 대한 예 : validator, tiles 등

다음은 <plug-in>의 attribute이다.

Name	Description
className	Plug-In 클래스의 전체 이름을 나타내며, 해당 클래스는 반드시 org.apache.struts.action.PlugIn 인터페이스를 구현해야 한다.

2.2.3.2.Samples

다음은 struts-config.xml 파일에서 plug-in 설정에 대한 예제이다.

```
<plug-in
    className="org.apache.struts.tiles.TilesPlugin">
    <set-property property="definitions-config" value="/WEB-INF/struts-tiles-defs.xml"/>
</plug-in>
```

2.2.4.form-beans

2.2.4.1.<form-beans>설정

- Action 수행에 사용되는 form bean의 정보 설정
- <form-beans>하위에 여러 개의 <form-bean> 구성 가능
- <form-bean>은 하위에 여러 개의 <form-property>를 가질 수 있음

다음은 <form-bean>의 attribute들이다.

Name	Description
className	Form bean들의 configuration 정보를 담고 있을 객체이다. 반드시 org.apache.struts.config.FormBeanConfig를 상속 받은 클래스여야 한다. 디폴트 값 : org.apache.struts.config.FormBeanConfig
dynamic	Form bean의 type attribute가 org.apache.struts.action.DynaActionForm이거나 org.apache.struts.action.DynaActionForm을 상속 받아 구현한 클래스이면, 이 attribute는 true여야 한다. 디폴트 값 : false
name	이 form bean의 이름이고, 다른 form bean들과 구분될 수 있는 identifier이다. [required]
type	이 form bean의 구현 클래스를 나타낸다. 이 클래스는 ActionForm의 서브클래스여야 한다. [required]

다음은 <form-property>의 attribute들이다.

Name	Description
className	Form property들의 configuration 정보를 담고 있을 객체이다. 반드시 org.apache.struts.config.FormPropertyConfig 또는 이를 상속 받은 클래스여야 한다. 디폴트 값 : org.apache.struts.config.FormPropertyConfig
initial	이 property의 initial 값을 나타내고, 문자로 표현된다.
name	Form bean이 사용하는 이 property의 이름이다. [required]
size	Form property의 type attribute가 array일 경우에, 그 array elements의 개수를 나타낸다.
type	Form bean이 사용하는 이 property의 type이다. 다음은 DynaActionForm에서 지원하는 type이다. <ul style="list-style-type: none"> • java.math.BigDecimal • java.math.BigInteger • boolean • byte • char • java.lang.Class • double, int, long, short • java.lang.String

Name	Description
	<ul style="list-style-type: none"> • java.sql.Data • java.sql.Time • java.sql.Timestamp

2.2.4.2.Samples

다음은 struts-config.xml 파일에서 form-beans 설정에 대한 예제이다.

```

<form-beans>
  <form-bean
    name="productForm"
    type="org.anyframe.struts.sample.sales.web.form.ProductForm">
  </form-bean>
  <form-bean
    name="employeeForm"
    type="org.apache.struts.action.DynaActionForm">
    <form-property name="name" type="java.lang.String"/>
    <form-property name="age" type="int"/>
    <form-property name="department" type="java.lang.String" initial="2"/>
    <form-property name="flavorIDs" type="java.lang.String[]"/>
  </form-bean>
</form-beans>

```

2.2.4.3.DynaActionForm

Struts 어플리케이션에서 각 Form에 대한 별개의 실재하는 ActionForm 클래스를 관리하는 것은 많은 시간을 요한다. DynaActionForm을 사용하게 되면, 위의 두 번째 form-bean 설정과 같이 struts-config.xml 상에 bean의 프로퍼티, 타입, 디폴트 값을 나열함으로써 ActionForm을 직접 작성하지 않아도 된다. 하지만 설정이 복잡하며 성능이 저하되는 문제점이 존재한다.

2.2.5.action-mappings

2.2.5.1.<action-mappings>설정

- 컨트롤러가 요청을 받았을 때 어떤 Action 인스턴스를 실행할 것인가에 대한 설정 정보
- <action-mappings>하위에 <action>을 이용해서 여러 개의 action 설정 가능
- <action> : 특정 request URI와 대응하는 Action 매핑 정의

2.2.5.2.<action>의 주요 attribute

- path : 확장자를 제외한 "/"로 시작하는 경로명
- type : action클래스의 이름
- scope : form bean이 저장되어 있는 context의 scope
- name : action과 연결된 form bean의 name
- role : Action 객체에 접근할 수 있는 권한을 설정
- input : form bean에서 validation error가 발생한 경우 되돌아가거나 상황을 표시할 수 있는 경로

다음은 <action>의 attribute들이다.

Name	Description
attribute	Form bean에 접근하기 위한, request-scope 또는 session-scope attribute의 name 값이다. 사용할 form bean을 다른 attribute의 이름으로 사용하고자 할 때 사용한다. Form bean이 name attribute에 기술되어 있을 때에만 기술될 수 있다.
className	Action들의 configuration 정보를 담고 있을 객체이다. 반드시 org.apache.struts.config.ActionMapping 또는 이를 상속 받은 클래스여야 한다. 디폴트 값 : org.apache.struts.config.ActionMapping
forward	요청된 request를 Action 클래스 대신하여 수행할 resource(*.do, *.jsp 등)의 상대(module-relative) 경로를 나타낸다. [required: 반드시 forward, include, type attribute 중의 하나만 기술되어야 한다.]
include	요청된 request를 Action 클래스 대신하여 수행할 resource(*.do, *.jsp 등)의 상대(module-relative) 경로를 나타낸다. [required: 반드시 forward, include, type attribute 중의 하나만 기술되어야 한다.]
input	Form bean에서 validation error가 발생했을 때, 이를 나타낼 resource(*.do, *.jsp 등)의 상대(module-relative) 경로를 가리킨다. Form bean이 name attribute에 기술되어 있을 때에만 기술될 수 있다. [required: form bean이 name attribute에 기술되어 있고 validation error들을 리턴할 경우]
name	이 action 매핑 사용하는 form bean의 이름을 나타낸다.
path	Submit된 request의 상대(module-relative)경로를 나타낸다. 이 attribute는 반드시 "/"으로 시작해야 하고, filename의 확장자 없이 기술되어야 한다. 예를 들어, "/main.do"은 적절한 path attribute의 기술 방법이 아니다. 왜냐하면 이미 do라는 확장자가 action 매핑에 사용되고 있는 것을 알고 있기 때문에, "/main"이라고만 기술하는 것이 옳다. [required]
parameter	Action 객체에 특별한 어떤 값을 넘겨주기 위한 설정 parameter이다. 현 Action 클래스에서는 이 attribute를 이용하지 않고 있기 때문에, 값을 넣는다 해도 처리되지 않는다. 만약 이 attribute를 사용하고자 하면, Action 클래스의 서브클래스를 만들어 구현해야 한다.
prefix	Request parameter name을 form bean property name에 매치시키는 데 사용되는 prefix를 나타낸다. Form bean이 name attribute에 기술되어 있을 때에만 설정할 수 있다.
roles	Action 객체에 접근할 수 있는 권한을 설정한다. 여러 role 이름들은 콤마(,)로 구분하여 쓸 수 있다. 예를 들어, "admin, master, user"라고 써주면 admin, master, user의 세 가지 권한 중 어느 한가지 권한이라도 가진 사용자는 이 action을 사용할 수 있게 된다.
scope	이 action이 사용하는 form bean이 저장되어 있는 context의 scope를 나타낸다. request 또는 session. 디폴트 값 : session
suffix	Request parameter name을 form bean property name에 매치시키는 데 사용되는 suffix를 나타낸다. Form bean이 name attribute에 기술되어 있을 때에만 설정할 수 있다.
type	요청된 request를 수행할 Action 클래스를 나타낸다. 이 클래스는 org.apache.struts.action.Action의 서브클래스여야 한다. [required: 반드시 forward, include, type attribute 중의 하나만 기술되어야 한다.]
unknown	설정 파일에 정의되지 않은 request를 처리하는 default action 매핑인지 여부를 나타낸다. 요청된 request를 수행할 action 매핑 객체가 없을 경우에, unknown이 true로 설정된 action 매핑 객체에게 이 request를 넘겨 처리하게 한다. 각각의 module마다 unknown이 true인 action 매핑은 하나만 있을 수 있다. 디폴트 값 : false
validate	Form bean에서 validation을 수행할지 여부를 나타낸다. 이 값이 true이면, form bean의 validate 메소드가 실행된다. 디폴트 값 : true

Name	Description
cancellable	Struts 1.3에 추가된 attribute로 Struts 1.2.9에서 <set-property>로 설정했던 것이 1.3부터 바뀌었다. Cancel Process를 사용하기 위해서 설정해야 한다.

2.2.5.3.Samples

다음은 struts-config-login.xml 파일에서 action-mappings 설정에 대한 예제이다.

```
<action-mappings>
  <action
    path="/login"
    type="org.anyframe.sample.struts.action.LoginAction"
    name="userForm"
    scope="request"
    input="/basic/login.jsp">
    <exception key="error.password.mismatch" path="/basic/login.jsp"
      type="javax.security.auth.login.LoginException" />
    <forward name="success" path="/basic/main.jsp" />
  </action>
  ...
</action-mappings>
```

'/login.do' 의 request에 대해 LoginAction 이 처리하도록 매핑되어 있으며, 이 때 action에 연결된 form bean은 UserForm 이다. request scope 동안 form bean 이 유지되며 forward 경로는 Action클래스에서 "success"라는 이름으로 forward name을 세팅 했기 때문에 /basic/main.jsp로 forwarding한다. exception 발생 시 /basic/login.jsp로 돌려진다.

<action>의 작성은 개발자가 반드시 숙지해야할 부분으로, request의 처리를 담당하는 Action을 매핑하고 페이지 네비게이션을 제어하는 등 웹 어플리케이션 개발의 중요한 작업이다.

2.2.6.global-forwards

2.2.6.1.<global-forwards> 설정

- 실제 forward 또는 redirect 할 수 있는 URI를 논리적인 이름으로 맵핑
- <global-forwards>하위에 <forward>를 이용해서 여러 개의 URI 매핑 설정
- 하나의 <forward>는 하나의 논리적인 이름을 module-relative 또는 context-relative URI 경로로 매핑함 (URI 경로를 직접 사용하는 것 보다 logic 내부적으로 정해진 이름을 사용함으로써, view로부터 controller와 model을 분리)
- 모든 action에서 사용할 수 있는 global level의 forward를 정의

forward 의 우선순위

<forward>은 전역(global) level과 action level 에서 정의될 수 있는데 action level 에서 선언된 것이 더 우선순위가 높다. 다음은 <forward>의 attribute들이다.

Name	Description
className	Forward들의 configuration 정보를 담고 있을 객체이다. org.apache.struts.config.ActionForward 또는 이를 상속 받은 클래스여야 한다. 디폴트 값 : org.apache.struts.config.ActionForward
name	현재 forward의 이름이고, 다른 forward들과 구분될 수 있는 identifier이다. [required]

Name	Description
path	Forward 또는 redirect할 resource(*.do, *.jsp 등)의 상대(module-relative or context-relative)경로를 나타낸다. [required]
redirect	RequestProcessor가 이 forward에 대해 redirect할 필요가 있을 때, true로 설정한다. 디폴트 값 : false

2.2.6.2.Samples

다음은 struts-config.xml 파일에서 global-forwards 설정에 대한 예제이다.

```
<global-forwards>
  <forward name="login" path="/login.jsp"/>
  <forward name="main" path="/main.do" redirect="true"/>
</global-forwards>
```

3.Controller

Controller는 MVC의 Model과 View사이의 중계자 역할을 한다. Struts의 Controller는 사용자의 Request를 처리하고 리소스의 초기화 등을 담당하고 있는 ActionServlet, 사용자 Request로 부터 ActionForm 객체를 생성하고 Action클래스의 execute()를 실행하는 RequestProcessor, 그리고 비즈니스 로직을 호출하고 성공과 실패에 대한 forward 정보를 설정하는 Action클래스로 이루어져있다.

3.1.ActionServlet

ActionServlet은 Struts에서 Controller역할을 담당하는 주요한 클래스이다.

3.1.1.ActionServlet의 역할

- 사용자의 요청을 받는 단일 진입점의 역할 (Front Controller)
- 리소스의 초기화와 clean-up 을 담당

3.1.2.초기화 프로세스

ActionServlet의 init() 메소드에서 다음과 같이 초기화 절차가 진행된다.

- Struts 내부에서 사용되는 예러나 경고에 사용되는 메세지 초기화
- web.xml에서 init-param으로 정의된 정보(debug, config, detail 등)들을 초기화
- web.xml에서 설정한 servlet-mapping 정보 초기화
- ServletContext에 ActionServlet 객체 저장
- init-param의 'config'에 의해 정의된 디폴트 모듈 정보(정의되지 않은 경우 '/WEB-INF/struts-config.xml')를 이용하여 ApplicationConfig 객체를 생성
- struts-config.xml에 메세지 리소스 관련 설정이 있을 경우 초기화
- struts-config.xml에 DataSource가 설정되어 있을 경우 초기화
- struts-config.xml에 정의된 Plug-In정보를 초기화

3.1.3.실행 시(ActionServlet 인스턴스가 HTTP Request를 받을 때)

- Request의 prefix에 따라 해당되는 서브 어플리케이션을 찾음
- RequestProcessor를 찾아 process() 메소드 호출

3.1.4.ShutDown 프로세스

- RequestProcessor의 destroy() 메소드 호출
- <plug-in>에 의해 정의된 값이 존재하는 경우 해당 destroy() 메소드 호출
- <data-sources>에 의해 정의된 값이 존재하는 경우 해당 close() 메소드 호출

3.2.RequestProcessor

RequestProcessor는 개발자가 필요에 따라 Request 처리에 대한 내용을 확장할 수 있도록 Struts 1.1 부터 제공되기 시작했다. 이로 인해 ActionServlet과 RequestProcessor 클래스가 분리되어 어플리케이션 모듈별로 각각의 RequestProcessor를 가질 수 있다는 장점이 있다. RequestProcessor는 기존에 ActionServlet이 제공하던 기본적인 기능에 더하여 확장 가능한 다양한 메소드를 제공한다.

Anyframe 에서는 RequestProcessor를 상속받아 DefaultRequestProcessor로 확장한 클래스를 제공한다.

3.2.1.RequestProcessor의 역할

- Request로부터 데이터를 받아서 ActionForm을 생성
- Action의 execute() 메소드실행
- ActionForm 전달
- Configuration에 정의된 대로 forward나 redirect 수행
- 어플리케이션의 configuration정보 유지

3.2.2.process() 메소드의 Request 처리 절차

- processMultipart(): HTTP Request의 content type이 multipart/form-data일 경우 새로운 Request Wrapper 생성
- processPath(): Request의 URI에서 ActionMapping처리를 위한 "path" 값을 추출
- processLocale(): Request로 부터 Locale 정보를 추출하여 session에 저장
- processContent(): Request의 content type과 encoding 정보를 설정
- processNoCache(): struts-config.xml에서 <controller>의 nocache 값이 true로 설정되었을 경우 HTTP Response의 header에 브라우저의 cache를 사용하지 않도록 설정
- processPreprocess(): Request가 처리되기 전에 수행되어야 하는 작업이 있을 경우 RequestProcessor를 상속받아서 확장하는 클래스에서 이 메소드를 **override**하여 구현
- processMapping(): 앞에서 추출한 "path"정보를 이용하여 ActionMapping 검색
- processRoles(): 사용자가 현재 Request를 수행할 수 있는 Role을 가지고 있는지 확인
- processActionForm(): ActionMapping에 설정된 ActionForm이 존재할 경우, 설정 파일에서 정의한 scope(session 또는 request)에서 ActionForm을 검색하고, 없을 경우 새로운 ActionForm을 생성하여 해당 scope에 저장
- processPopulate(): HTTP Request 파라미터들을 ActionForm에 저장
- processValidate(): 설정 파일에서 'validate' 값이 true로 설정된 경우, ActionForm의 validate() 메소드를 호출.
- processForward(): 설정 파일에서 <action>에 forward나 include 속성이 정의되어 있는 경우, RequestDispatcher의 forward(), include() 메소드를 호출
- processActionCreate(): Request에 해당하는 Action 객체 생성. 최초에 한번 생성된 후 ServletContext에 저장되어 재사용 됨
- processActionPerform(): Action 객체의 execute() 메소드 호출

- processActionForward(): Action 객체의 execute() 메소드의 리턴 값인 ActionForward에 해당하는 URL로 forward

3.2.3.Sample

아래는 RequestProcessor 를 struts-config.xml의 <controller>로 설정한 예이다.

```
<controller contentType="text/html;charset=UTF-8"
    debug="3" locale="true" nocache="true"
    processorClass="org.apache.struts.action.RequestProcessor"/>
```

위의 processorClass 에는 필요에 따라 RequestProcessor를 확장한 Controller를 설정할 수 있다. Anyframe 에서는 RequestProcessor 를 확장한 DefaultRequestProcessor 를 제공한다.

3.3.Action

Action 클래스는 비즈니스 로직을 호출하고 성공과 실패에 대한 적절한 forward 정보를 설정한다.

3.3.1.Action의 역할

- Action의 execute()는 클라이언트의 요청과 Business Logic을 연결
- Action은 MVC 구조에서 Controller와 Model사이를 이어주는 역할

3.3.2.Action의 구현

- 하나의 인스턴스가 그 action으로 매핑된 모든 request를 처리하므로 execute() 메소드를 thread-safe 메소드로 구현해야 함
- Action 클래스에서 실행하는 Business Logic은 한 업무 처리에만 관련이 있어야 함
- execute()는 항상 ActionForward를 리턴해야함
- 비즈니스 로직을 호출하는 로직만 존재해야 함, 비즈니스 로직은 모델 객체에 구현해야 함

Action 인스턴스는 단일 인스턴스로 모든 클라이언트 요청이 같은 인스턴스를 공유하므로 특정 클라이언트의 정보를 Action 클래스의 멤버변수로 저장하는 것은 잘못된 구현이다. 클라이언트의 정보가 필요한 경우 request나 session에 저장하도록 하고 특정 클라이언트의 상태를 나타내기 위해서는 execute() 메소드내에서 지역변수를 사용하면 각 스레드별로 영향을 끼치지 않는다. Action 클래스의 작성은 개발자가 반드시 숙지해야 할 부분으로, request로 전달된 사용자 입력 데이터를 추출하여 비즈니스 로직을 호출하고 그 결과를 다시 클라이언트로 포워드하는 중요한 작업이다.

3.3.3.Sample

- Action 클래스의 작성 예

아래는 로그인에 필요한 사용자 입력 값을 받아 체크한 후 Session에 저장하는 LoginAction.java 의 소스코드이다.

```
public class LoginAction extends Action{

    public ActionForward execute(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response)
        throws Exception {

        UserForm userForm = (UserForm) form;
```

```

String userId = userForm.getUserId();
String password = userForm.getPassword();

//사용자 Id, Password 체크
if ((userId != null && userId.equals("anyframe"))
    && (password != null && password.equals("anyframe"))) {

    //로그인 성공시 Session에 사용자 정보를 저장한다.
    Set principals = new HashSet();
    Set credentials = new HashSet();

    //사용자의 이름과 권한을 저장한다.
    principals.add(new TypedPrincipal("Anyframe", TypedPrincipal.USER));
    principals.add(new TypedPrincipal("ADMIN", TypedPrincipal.GROUP));
    Subject subject = new Subject(false, principals, credentials,
        credentials);

    HttpSession session = request.getSession();
    session.setAttribute("subject", subject);

} else {
    throw new FailedLoginException();
}
return (mapping.findForward("success"));
}
}

```

execute()메소드가 호출되면 ActionForm에 저장된 userid 와 password 값을 받아오고 사용자 인증이 성공하면(위에서는 비즈니스 서비스없이 임시로 anyframe/anyframe 에 대해 ADMIN 사용자로 체크 함) Subject객체를 Session에 설정한 후 success로 지정된 ActionForward를 리턴한다. 인증에 실패할 경우에는 FailedLoginException을 발생시켜 struts-config.xml에 설정된 Exception처리 설정을 따른다.

- Action 매핑 예

아래는 위의 LoginAction.java의 매핑 정보를 설정한 struts-config-login.xml의 일부이다.

```

<action
  path="/login"
  type="org.anyframe.sample.struts.action.LoginAction"
  name="userForm"
  scope="request"
  input="/basic/login.jsp">
  <exception key="error.password.mismatch" path="/basic/login.jsp"
    type="javax.security.auth.login.FailedLoginException" />
  <forward name="success" path="/basic/main.jsp" />
</action>

```

- Spring Bean 형태의 비즈니스 서비스 Invoke

Spring Framework에서는 Struts와의 통합을 위해 spring-webmvc-struts의 ActionSupport(내부적으로 Struts의 Action을 extends하고 있음)를 제공하고 있다. Struts의 Action클래스 대신 ActionSupport 클래스를 extends하면 Spring Framework Bean형태의 비즈니스 서비스에 쉽게 접근이 가능하다. 사용 방법은 다음과 같다.

```

public class SampleAction extends ActionSupport {
    public ActionForward execute(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response)
        throws Exception {

        ApplicationContext ctx = getWebApplicationContext();
        BusinessService businessService =
            (BusinessService)ctx.getBean("businessService");
    }
}

```

```

//종락
return mapping.findForward("success");
}
}

```

3.4.ActionForward

3.4.1.ActionForward의 역할

- JSP 페이지나 서블릿 같은 웹 리소스의 논리적인 추상화
- 물리적인 리소스에 대한 정보는 설정 파일에 저장

Action 클래스에서 execute() 메소드는 ActionForward 객체를 리턴한다. ActionForward는 리소스를 감싸서 어플리케이션과 물리적인 리소스를 분리하며 설정 파일에 forward의 name, path, redirect 속성과 같은 요소들로 정의하고 코드에는 포함하지 않는다. RequestDispatcher는 redirect 요소의 값에 따라 ActionForward의 포워드나 리다이렉트를 실행하게 된다. Action에서 ActionForward를 반환할 때 설정 파일에 미리 정의된 ActionForward를 알아내기 위해 일반적으로 ActionMapping을 사용한다. 다음 코드는 ActionMapping을 사용하여 논리적 이름에 근거해 ActionForward를 찾는 방법을 보여준다.

```
return mapping.findForward("success");
```

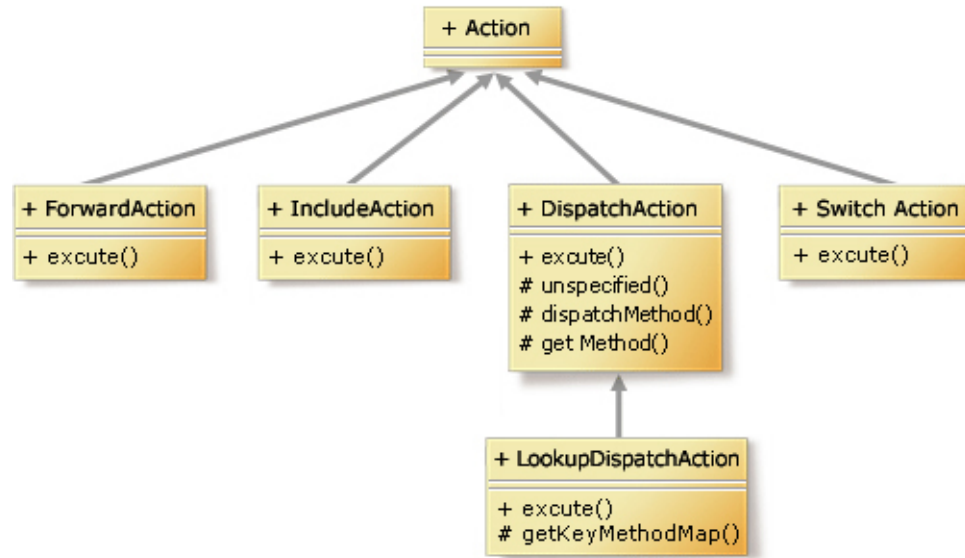
3.5.Actions Package

Struts 에는 어플리케이션에서 쉽게 통합할 수 있는 out-of-the-box 형태의 Action 클래스 5개를 포함하고 있으며 이를 이용해 개발 시간을 단축할 수 있다.

3.5.1.org.apache.struts.actions 패키지에 미리 정의되어 있는 Action

- org.apache.struts.actions.ForwardAction
- org.apache.struts.actions.IncludeAction
- org.apache.struts.actions.DispatchAction
- org.apache.struts.actions.LookupDispatchAction
- org.apache.struts.actions.SwitchAction

다음 그림은 Struts 의 org.apache.struts.actions 패키지에 해당하는 클래스들의 간단한 클래스 다이어그램이다.



3.5.2.org.apache.struts.actions.ForwardAction

단순히 포워딩만 실행하는 경우 Action 클래스를 직접 구현하지 않고 struts에서 미리 구현된 ForwardAction을 사용한다. 이 Action 클래스는 파라미터 속성에 정의된 URI로 포워드를 실행한다.

```

<action
  path="/loginView"
  type="org.apache.struts.actions.ForwardAction"
  parameter="/basic/login.jsp">
</action>
  
```

request url이 loginView.do이면 /basic/login.jsp로 포워드하게 된다. Action 클래스를 통하지 않고 JSP를 직접 호출하는 것은 바람직하지 못하다. MVC 구조에서 Controller의 책임을 위반할 뿐만 아니라 Struts에서 처리하는 중간 단계를 건너뛰게 되므로 문제가 발생할 수 있다.(ex. 리소스 번들에서 올바른 메시지를 가져오지 못함)

3.5.3.org.apache.struts.actions.IncludeAction

ForwardAction과 비슷하며 기존에 있던 서블릿 기반 컴포넌트들을 스트럿츠 기반 웹 어플리케이션과 쉽게 통합하기 위해 제공하는 것이다.

```

<action
  input="/subscription.jsp"
  name="subscriptionForm"
  path="/saveSubscription"
  parameter="/path/to/processing/servlet"
  scope="request"
  type="org.apache.struts.actions.IncludeAction" />
</action>
  
```

3.5.4.org.apache.struts.actions.DispatchAction

기능마다 각각의 Action 클래스를 만들게 되면 클래스 수가 많아져서 관리하기가 어렵다. Struts에서는 관련된 기능을 하나의 Action으로 모을 수 있는 DispatchAction을 제공한다. 예를 들면 add, update, view 와 같은 여러 기능을 하나의 Action(extends DispatchAction) 안에 메소드로 구현하면 (cf. 일반 Action에서는 execute() 메소드 구현) 설정 파일의 parameter 속성값으로 지정한 문자열 키값(아래에

서는 "method"로 지정하였음. 다른 문자열을 써도 됨.)으로 메소드를 호출할 수 있게 된다. 아래는 DispatchAction을 extends한 ShoppingCartAction.java 의 소스코드이다.

```
public class ShoppingCartAction extends DispatchAction {

    public ActionForward add (
        ActionMapping mapping,
        ActionForm form,
        HttpServletRequest request,
        HttpServletResponse response) throws Exception{
        // TODO : add 기능 관련 토직
        return mapping.findForward("add");
    }

    public ActionForward update (
        ...
        // TODO : update 기능 관련 토직
        return mapping.findForward("update");
    }

    public ActionForward search (
        ...
        // TODO : search 기능 관련 토직
        return mapping.findForward("search");
    }

    public ActionForward view (
        ...
        // TODO : view 기능 관련 토직
        return mapping.findForward("view");
    }
}
```

ShoppingCartAction에 대한 action 매핑 정보를 설정한 struts-config-dispatch.xml 의 일부이다.

```
<action
    path="/shoppingCart"
    type="org.anyframe.sample.struts.web.action.ShoppingCartAction"
    name="dispatchForm"
    parameter="dispatchMethod"
    scope="request">
    <forward name="add" path="/dispatchActionView.do" />
    <forward name="list" path="/dispatchActionView.do" />
    <forward name="update" path="/dispatchActionView.do" />
    <forward name="delete" path="/dispatchActionView.do" />
</action>
```

3.5.5.org.apache.struts.actions.LookupDispatchAction

DispatchAction과 비슷하나 메소드를 찾을 때 parameter 속성에서 찾는 것이 아니라 리소스 번들에 키 값을 이용해 찾게 된다. LookupDispatchAction을 사용하게 되면 하나의 HTML Form에 submit 버튼이 여러 개 있는 경우 더 쉽게 처리할 수 있다.

3.5.6.org.apache.struts.actions.SwitchAction

이 클래스는 하나의 어플리케이션 모듈에서 다른 모듈로 스위칭을 지원하고 어플리케이션의 리소스를 컨트롤한다. prefix, page 파라미터를 통하여 어플리케이션 모듈을 전환할 수 있다. 사용 예는 다음과 같다.

```
<action path="/toModule"  
        type="org.apache.struts.actions.SwitchAction"/>
```

```
/toModule.do?prefix=/moduleB&page=/index.do
```

디폴트 모듈로 전환하고자 하는 경우 prefix를 null String으로 설정하여 호출할 수 있다.

```
/toModule.do?prefix=&page=/index.do
```

4.View

View 는 클라이언트가 모델의 상태를 보기 위해 사용하는 창(window)이다. 하나의 모델은 여러 창, 즉 뷰를 포함할 수 있으며 클라이언트가 어떤 view를 통해 모델을 보느냐에 따라 화면이 달라진다. 표시 방법으로 XML, XSLT, SOAP, HTML 등 다양한 방법을 택할 수 있으며 Anyframe 에서는 주로 자바 코드와 태그로 View를 구성하며 JSP를 기반으로 클라이언트에 동적인 콘텐츠를 제공하게 된다.

View의 역할

- 사용자 입력 데이터 수용 및 데이터 표시
- 입력 데이터 검증
- 에러처리
- 국제화

자주 사용되는 View 형태

- HTML : 브라우저를 통해 최종사용자가 보는 페이지이다.
- JSP Custom Tag : Struts 어플리케이션에서 매우 중요한 역할을 한다, 필수는 아님.
- JavaScript and StyleSheet : 사용을 금지하지 않으며 적절히 사용하면 효과적인 view를 만들 수 있다.
- MessageResource Bundle : Localization 기능을 제공하며 유지보수 시간을 절약할 수 있다.
- Multimedia file

View의 구성 요소는 JSP 및 관련 기술과 ActionForm이 있으며 해당 내용은 아래와 같다.

- JSP
 - 서버에서 동적 웹 콘텐츠를 생성하는 자바 플랫폼 기술, Servlet Container 상에서 수행되는 서버 스크립트 언어
 - 서블릿으로 변환되어 수행되며 최초 요청 시 해당 페이지에 대한 컴파일 된 서블릿 인스턴스는 한 번만 생성되므로 서버 자원을 효율적으로 사용
 - 강력한 이식성, 빠른 수행속도, 프리젠테이션 로직과 비즈니스 로직 분리, 컴포넌트의 재사용, 커스텀 태그의 사용으로 인한 개발 편의성 등이 장점
- Javascript
 - 넷스케이프에서 만든 인터프리터 형 스크립트 언어
 - 자바스크립트 코드는 HTML 페이지 내에 삽입될 수 있으며, 클라이언트 측인 웹 브라우저에 의해 해석됨
 - 객체 지향 모델로써 구조화된 문서를 표현하는 표준 형식인 DOM(Document Object Model) 객체를 다루고 프로그램으로 조작(manipulate)함
- CSS
 - Cascading Style Sheet는 마크업 언어(HTML, XHTML, XML)가 실제 표시되는 방법을 기술하는 언어로 W3C의 표준임
 - 스타일 정보의 수정 시 홈페이지 전체에서 이에 해당하는 요소들이 한꺼번에 반영됨
 - 각기 다른 사용자 환경에서도 동일한 형태의 문서를 제공

DHTML : 정적 마크업 언어인 HTML, 클라이언트 기반 스크립트 언어(ex. Javascript)와 스타일 정의 언어인 CSS를 조합하여 대화형 웹 사이트를 제작하는 기법을 의미함

ActionForm

웹 어플리케이션에서 사용자의 입력을 받을 때 페이지에는 텍스트 박스, 버튼 등과 같은 컴포넌트들이 HTML의 폼 요소 내에 포함되어 있고 사용자가 버튼을 누르게 되면 필드 내에 있는 값들이 HTTP request와 함께 서버로 submit 된다. 서버 어플리케이션은 request에서 이 입력 값들을 꺼내어 올바른 데이터를 입력했는지 validation을 수행하고 나서 실제 비즈니스를 수행하기 위해 Action으로 데이터를 넘기게 된다. 만일 입력 데이터가 validation rule을 통과하지 못한 경우 에러 메시지를 설정하여 입력 페이지로 돌아가게끔 처리해야 한다. 이처럼 요청에서 입력 값을 꺼내어 검증 수행하고 실패에 대한 에러 메시지를 출력하는 등의 기능을 직접 구현하는 것은 쉬운 일이 아니다. 또한 이런 작업은 전체 어플리케이션 내에서 반복해서 일어나므로 재사용하는 것이 좋다. 이러한 작업들을 해주는 것이 org.apache.struts.action.ActionForm 클래스이다.

- ActionForm의 역할

- 요청에서 입력 값을 꺼내어 검증 수행, 실패에 대한 에러 메시지를 출력하는 등의 일련의 처리 과정을 재사용
- ActionForm은 클라이언트의 입력 값을 Action으로 전달하고, 결과를 되돌려줄 수 있음
- 입력 데이터들을 검증하는 동안 상태를 보관하는 버퍼로 동작
- 확실하지 않은 입력 값들을 검증 룰을 통해 세밀하게 조사하기 전까지 비즈니스 계층 밖에 위치하도록 해주는 firewall 역할
- ActionForm을 화면 표시 데이터로 설정하여 HTML 폼의 입력 필드를 쉽게 표시할 수 있음

Html 입력 Form으로 부터 받은 parameter 들은 자동으로 ActionForm 객체에 채워진다. 검증을 위한 validate() 메소드와 parameter가 ActionForm에 채워지기 전에 초기화 하는 reset() 메소드를 구현해야 한다. struts-config.xml 에 ActionForm 에 대한 <form-beans> 정의가 필요하다. ActionForm은 Model의 부분이 아니다. 비즈니스 처리를 수행하기 위한 Model 영역은 Controller / View 와 완전히 분리하여야 하며 직접 비즈니스 계층으로 전달해서는 안되고 ValueObject 나 Parameters 같은 형태의 Data Transfer Object를 생성하여 전달하도록 해야 한다.

- ActionForm의 단점

- 어플리케이션 개발자가 ActionForm의 서브클래스를 직접 구현해야 함
- 많은 수의 클래스가 생겨날 수 있어서 유지보수 관리 어려움
- validate() 메소드를 구현하려면 DynaActionForm을 상속받아 직접 구현해야함. Validator 프레임워크를 이용하는 것이 좋음.

- DynaActionForm

- 실제 구현 클래스들을 만들 필요가 없음
- Property는 configuration파일에서 설정
- validate() 메소드를 구현하려면 DynaActionForm을 상속받아 직접 구현해야함. Validator 프레임워크를 이용하는 것이 좋음.

- ActionForm의 scope

- ActionForm 객체가 저장되어 유지되는 context의 scope를 나타낸다.
- session, request 2가지 레벨이 있다.
- struts-config.xml 의 <action>의 scope 속성으로 설정한다. default는 session이다. session scope 일 경우 ActionForm의 제거에 유의해야 한다.

- ActionForm의 LifeCycle

1. 액션의 매핑 정보를 확인하고 ActionForm이 설정되어 있는지 검사한다.
2. 액션에 ActionForm이 설정되어 있다면, 폼 빈의 설정 정보에서 action요소의 name속성을 찾는데 사용한다.
3. 이미 만든 ActionForm 인스턴스가 있는지 검사한다.
4. ActionForm 인스턴스가 적합한 scope에 있고 요청에 필요한 타입과 같다면 재사용한다.
5. ActionForm 인스턴스가 적합한 scope내에 없다면 새로운 인스턴스를 만들고 action 요소의 scope 속성에 따른 scope에 저장한다.
6. ActionForm 인스턴스의 reset() 메소드를 호출한다.
7. 요청 파라미터의 이름에 따른 ActionForm의 setter 메소드를 통해서 요청 파라미터의 값을 ActionForm에 입력한다. (populate 라고 한다.)
8. 마지막으로 validate 속성이 "true"로 설정되어 있다면 ActionForm 인스턴스의 validate()메소드를 수행하고 검증과정에 에러가 있다면 에러들을 반환한다.

다음은 ActionForm 작성의 예를 보여주는 UserForm.java 의 일부 소스코드이다.

```
public class UserForm extends ActionForm{

    private String userId;

    private String password;

    public String getUserId() {
        return userId;
    }

    public void setUserId(String userId) {
        this.userId = userId;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public void reset(ActionMapping mapping, HttpServletRequest request) {
        this.password = null;
        this.userId = null;
    }

    public ActionErrors validate(ActionMapping mapping,
                                HttpServletRequest request) {
        ActionErrors errors = new ActionErrors();
        if ((userId == null) || (userId.length() < 1))
            errors.add("userId", new ActionMessage("error.userid.required"));
        if ((password == null) || (password.length() < 1))
            errors.add("password", new ActionMessage("error.password.required"));
        return errors;
    }
}
```

화면에서 입력받을 요소에 대한 attribute를 정의하고 해당 getter/setter 메소드를 작성한다. 또한 검증을 위한 validate 메소드와 초기화를 위한 reset 메소드를 구현해야 한다.

- ActionErrors 사용하기

위 ActionForm 소스의 validate 메소드에서 ActionErrors 객체를 반환하는 것을 보았다. 여기서는 ActionErrors의 사용에 대해 알아본다.

- ActionErrors는 어플리케이션에서 발견한 에러를 하나 이상 캡슐화한다.

- request에 저장된 ActionErrors는 이후 JSP 에서 custom tag를 통해 사용자들에게 에러 메시지로 보여진다.

다음은 ActionMessage 생성의 예이다.

```
ActionMessage message
    = new ActionMessage("global.error.login.requiredfield", "email");
```

위의 첫번째 인자는 리소스 번들 내의 키 중 하나와 일치하는 문자열이고, 두번째 인자는 메시지를 위한 parameter이다. 아래는 리소스 번들 내의 관련 메시지 정의이다.

```
global.error.login.requiredfield=The {0} field is required.
```

{0} 부분에는 email 이란 글자가 찍혀 표시된다. 위의 형태 외에도 복수개의 메시지 파라미터를 처리할 수 있는 몇가지 생성자 유형이 더 있다. ActionMessage는 ActionForm의 validate 에서만 생성할 수 있는 것은 아니다. 예를 들어, Action에서 호출한 비즈니스 처리에서 예외가 발생했고 이를 사용자에게 알리기 위한 에러 메시지를 추가하려고 할 때도 ActionMessage를 사용할 수 있다. JSP 에서 Taglib를 이용해 메세지로 ActionErrors를 보여주는 예는 다음과 같다.

```
<%@ page contentType="text/html; charset=euc-kr" %>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>

<html:html>
  <head>
    <title>Error Page</title>
  </head>
  <body>
    <html:errors/>
  </body>
</html:html>
```

4.1.Taglib

Struts Framework 는 몇몇 종류의 태그들을 포함하고 있으며 이 Tag Library 기능을 이용하면 프리젠테이션 계층을 더 쉽게 제어할 수 있고 재사용이 용이하다. 제공하는 Tag library를 사용하여 JSP 페이지에서 자바 코드를 일체 사용하지 않고도 개발이 가능하다.

4.1.1.Taglib의 특징

4.1.1.1.Tag library의 필요성

- GUI 제작시 재사용을 통해 생산성 향상
- scripting 요소의 제거로 개발자와 디자이너간 역할 분담에 도움을 줌
- 전체 업무 영역에서 많이 사용되는 공통 기능을 커스텀 태그로 구현하면 생산성 향상에 도움이 될 수 있음

4.1.1.2.Tag library의 구성요소

- Tag Handler : Tag 가 어떤 식으로 동작하는지 정의하는 클래스, javax.servlet.jsp.tagext.Tag 인터페이스를 구현한 javax.servlet.jsp.tagext.TagSupport 나 BodyTagSupport를 상속(extends)하여 구현함
- Tag Library Descriptor (TLD) : Tag Handler 클래스로 구현한 Custom Tag 들에 대한 XML 형식의 기술문서
- taglib 지시자 (JSP 페이지 내에서) : JSP 페이지에서 해당 Tag Library를 사용하기 위한 지시자

4.1.1.3.Tag library의 종류

- Struts Tag Library : HTML, Logic, Bean, Nested
- JSTL : core, fmt, xml, sql
- Jakarta taglibs
- 해당 프로젝트에 맞게 작성한 Custom Tag Library

4.1.2.Struts Taglib

Struts Tag library의 종류는 아래와 같다.

- HTML tag : HTML 입력 폼을 작성하거나 HTML 기반 사용자 인터페이스를 작성하는데 일반적으로 쓰이는 태그
- Logic tag : 조건 처리, Collection 객체를 loop을 돌면서 출력, 흐름 제어 등에 쓰이는 태그
- Bean tag : 자바 빈과 관련 프로퍼티들에 접근하는 데 이용되는 태그. 변수의 기술을 통해 쉽게 접근할 수 있는 새 빈을 정의할 수 있음
- Template tag : layout를 공유하는 동적인 JSP 템플릿을 작성할 때 유용하게 쓸 수 있는 태그
- Nested tag : Struts 태그들을 중첩해서 사용할 수 있게 해줌

사용법은 다른 Tag Library와 같다. 아래는 JSP에 taglib 선언한 예이다.

```
<%@ taglib
    uri="http://struts.apache.org/tags-bean"
    prefix="bean"%>
```

많은 경우 Tag Library 들은 자바빈즈와 함께 사용된다. 자바 빈은 HTML 폼의 입력 필드에 대응하는 프로퍼티들을 포함하는 ActionForm일 수도 있지만 Value Object들도 사용할 수 있다.

4.1.2.1.HTML

다음은 HTML Tag Library의 태그들에 대한 설명이다.

Name	Description
base	HTML의 <base>를 표시한다.
button	button 입력 필드를 표시한다.
cancel	cancel 버튼을 표시한다.
checkbox	checkbox 입력필드를 표시한다.

Name	Description
errors	모든 일련의 에러 메시지를 조건적으로 표시한다.
file	file 선택 입력 필드를 표시한다.
form	HTML <form>을 정의한다.
frame	HTML의 <frame>을 정의한다.
hidden	hidden 필드를 표시한다.
html	HTML의 <html>을 표시한다.
img	HTML의 를 표시한다.
javascript	validator 플러그 인이 로딩한 validation-rule에 기반한 javascript를 표시한다.
link	HTML의 앵커나 하이퍼링크를 표시한다.
messages	모여진 일련의 메시지들을 조건적으로 표시한다.
multibox	멀티플 checkbox 입력 필드를 표시한다.
option	select의 option을 표시한다.
options	select의 option들의 집합을 표시한다.
optionsCollection	select의 option들의 집합을 표시한다.
password	password 입력필드를 표시한다.
radio	radio 버튼 입력 필드를 표시한다.
reset	reset 버튼 입력 필드를 표시한다.
rewrite	URI를 표시한다.
select	<select>를 표시한다.
submit	submit 버튼을 표시한다.
text	"text" 타입의 입력 필드를 표시한다.
textarea	textarea 입력 필드를 표시한다

다음은 HTML 의 link, password 태그의 예이다.

```
<tr>
  <td colspan="4" align="center">
    <html:link page="/html-link.do?doubleProperty=321.321&longProperty=321321">
      Double and long via hard coded changes
    </html:link>
  </td>
</tr>
```

```
<html:password property="password"
  size="15" maxlength="16" redisplay="false" />
```

4.1.2.2.Logic

다음은 Logic Tag Library의 태그들에 대한 설명이다.

Name	Description
empty	요청한 변수가 null 또는 빈 문자열인 경우 이 태그의 바디 콘텐츠를 수행한다.
equal	요청한 변수가 지정한 값과 같을 경우 이 태그의 바디 콘텐츠를 수행한다.

Name	Description
forward	ActionForward 엔트리를 통해 지정한 페이지로 포워드를 수행한다.
greaterEqual	요청한 변수가 지정한 값보다 크거나 동일한 경우 이 태그의 바디 콘텐츠를 수행한다.
greaterThan	요청한 변수가 지정한 값보다 큰 경우...
iterate	지정한 컬렉션으로 이 태그 내의 바디 콘텐츠를 반복한다
lessEqual	요청한 변수가 지정한 값보다 작거나 동일한 경우
lessThan	요청한 변수가 지정한 값보다 작을 경우
match	지정한 값이 요청한 변수의 부분 문자열에 일치하는 경우
messagesNotPresent	지정한 메시지가 이 요청에 없는 경우
messagesPresent	지정한 메시지가 이 요청에 있는 경우
notEmpty	요청한 변수가 null도, 빈 문자열도 아닌 경우
notEqual	요청한 변수가 지정한 값과 동일하지 않은 경우
notMatch	지정한 값이 요청한 변수의 부분 문자열에 일치하지 않는 경우 이 태그의 바디 콘텐츠를 수행한다.
notPresent	지정한 값이 이 Request에 없는 경우
present	지정한 값이 이 Request에 있는 경우
redirect	HTTP Redirect를 표시한다.

다음은 notEmpty, iterate 태그의 예이다.

```
<logic:notEmpty name="userSummary" property="addresses">
<!--이 부분은 address Collection 의 모든 객체들을 돌려 반복 출력하는 logic 태그로 구성하면 됨
-->
</logic:notEmpty>
```

```
<logic:iterate id="address" name="usersSummary" property="addresses">
<!--address 객체를 테이블 형태로 출력한다. -->
</logic:iterate>
```

4.1.2.3.Bean

다음은 Bean Tag Library의 태그들에 대한 설명이다.

Name	Description
cookie	지정한 요청 쿠키의 값에 근거해 변수를 정의한다
define	지정한 빈 프로퍼티의 값에 근거해 변수를 정의한다
header	지정한 요청 헤더의 값에 근거해 변수를 정의한다
include	동적인 어플리케이션 요청의 응답을 로드해 빈으로 이용할 수 있도록 한다
message	응답이 되는 국제화된 메시지 문자열을 표시한다
page	지정한 아이템을 빈으로써 페이지 문맥에서 꺼낸다
parameter	지정한 요청 파라미터의 값에 근거해 변수를 정의한다
resource	웹 어플리케이션의 자원을 로드 해 빈으로 이용할 수 있도록 한다
size	Collection 또는 Map의 요소의 갯수를 포함한 빈을 정의한다.
struts	지정한 Struts 내부 설정 객체를 빈으로 꺼낸다.

Name	Description
write	지정한 빈 프로퍼티의 값을 표시한다.

다음은 message, write 태그의 예이다.

```
<td><bean:message key="global.user.firstName"/>:</td>
```

위와 같이 사용하면 global.user.firstName 에 해당하는 메시지를 가져와 보여준다.

```
<td>Hello <bean:write name="user" property="firstName"/>:</td>
```

위와 같이 사용하면 user 라는 빈에서 firstName을 꺼내 Hello 옆에 붙여준다.

Nested

한 태그를 다른 태그에 중첩하여 사용하고자 할 경우 적용할 수 있다. Struts에서 지원하는 현재 태그와 매칭되는 HTML Nested Tag, Logic Nested Tag, Bean Nested Tag가 존재하며 사용 방법은 원래 태그와 같다.

4.1.3.JSP Standard Tag Library

- JSR52, JSP Standard Tag Library 스펙
- 어떤 컨테이너에서도 사용 가능한 표준 태그 집합을 정의
- core, fmt, xml, sql 태그가 있음
- Servlet 2.3, JSP 1.2 이상을 지원하는 컨테이너 필요(Tomcat 을 비롯하여 대부분 지원함)

JSTL은 데이터의 포맷, 반복 처리, 조건 처리 등 전형적인 프리젠테이션 레이어를 위한 표준 구현을 제공하기 때문에, JSP 작성자들이 어플리케이션 개발에 집중하는데 도움이 되며 일반적인 기능을 커스텀 태그 라이브러리의 표준 세트로 패키징했기 때문에 JSP 작성자들이 스크립팅 엘리먼트에 대한 필요를 줄이고 관련된 관리 비용을 피할 수 있도록 한다. 이에 반해 pure 자바 코드에 비해 시스템 리소스를 많이 사용하며 극한 부하 상황에서는 2~3배의 성능 저하가 발생할 수 있으므로 성능이 이슈가 되는 경우 사용에 유의하도록 한다.

Struts Bean, Logic 태그들은 JSTL로 바꾸어 더 쉽게 사용할 수 있다. 다음은 JSTL Core 태그 중 조건 분기 및 Collection loop 처리의 예이다.

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
..
<!-- 테이블의 리스트 반복부 -->
<c:choose>
  <c:when test="${page.totalCount <= 0}">
    <tr class="ct_list_pop">
      <td colspan="11" align="center">::: 조회된 사용자 정보가 없습니다. :::</td>
    </tr>
  </c:when>
  <c:otherwise>
    <c:forEach var="userVO" items="${page.list}" varStatus="status">
      <tr class="ct_list_pop">
        <td align="center">
          <c:out value="${status.count + ((page.currentPage - 1) * pageSize) }"/>
        </td>
      </tr>
    </c:forEach>
  </c:otherwise>
</c:choose>
```

```

</td>
<td></td>
<td align=" left">
    <a href="javascript:fncGetUser(' <c:out value="\${userVO.userId}"/> ');">
        <c:out value="\${userVO.userId}"/>
    </a>
</td>
<td></td>
<td align=" left"><c:out value="\${userVO.userName}"/></td>
<td></td>
<td align="center" style="padding-right:3px;"><c:out value="\${userVO.ssn}"/>
</td>
<td></td>
<td align="center"><c:out value="\${userVO.cellPhone}"/></td>
<td></td>
<td align=" left"><c:out value="\${userVO.email}"/></td>
</tr>
<tr>
    <td colspan="11" bgcolor="D6D7D6" height="1"></td>
</tr>
</c:forEach>
</c:otherwise>
</c:choose>
..

```

4.1.4. 기타 Taglib

4.1.4.1. Jakarta taglibs

- 자카르타 Taglibs 프로젝트에서 25개 정도의 태그 라이브러리 제공
- <http://jakarta.apache.org/taglibs/> [<http://tomcat.apache.org/taglibs/index.html>]참조

JSP Standard Tag Library (JSTL) 의 구현인 Standard Taglib 1.1(JSTL 1.1 - Servlet 2.4, JSP 2.0 이상)을 비롯한 많은 태그 라이브러리들을 활용할 수 있다.

4.1.4.2. Custom Tags

- Tag interface를 구현한 클래스를 만들고 XML 형식의 tag library descriptor (TLD) 파일을 제공해야 함
- 보통 관련된 Custom Tag의 묶음인 Tag Library 형태로 제공됨
- 전체 업무 영역에서 많이 사용되는 공통 기능을 커스텀 태그로 직접 구현

예를 들면 코드 테이블로부터 코드리스트를 조회하여 select box 형식으로 표출하는 Custom Tag 를 적용하면 전체 업무의 생산성 향상에 도움이 될 것이다. Anyframe 에서 제공하는 pagenavigator 태그도 다중 행의 리스트성 자료의 페이지 처리 기능을 돕기 위해 직접 구현한 커스텀 태그이다.

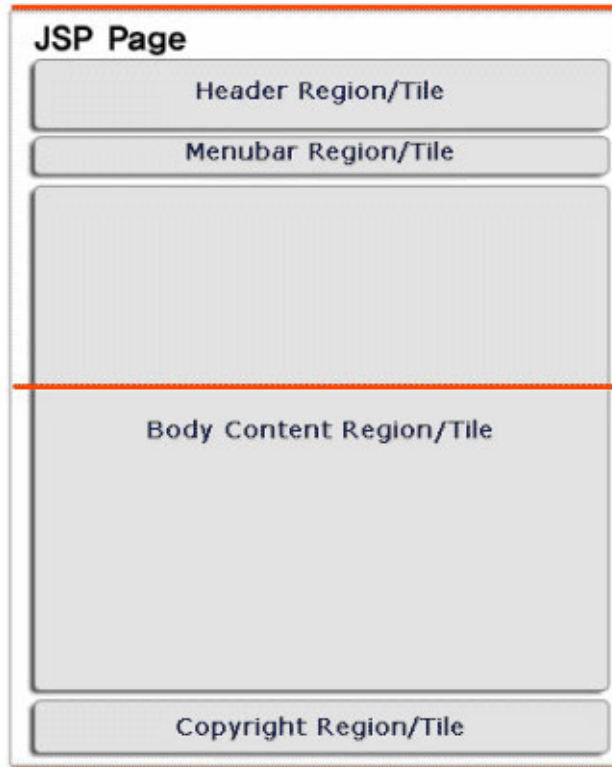
4.2. Tiles

JSP Page의 Layout을 구성하는 방법에는 몇 가지가 있다. 대표적인 것이 include 지시어를 사용하여 중복 되는 코드를 줄여주는 방법이다. 그러나 여전히 한계가 존재하며 이를 해결하기 위한 더 나은 접근법은 템플릿 중심 아키텍처를 적용하는 것이다. Tiles 는 이를 지원하는 templating system으로 웹 어플리케이션의 유저 인터페이스를 단순화 하기위해 만들어졌다. Struts 에는 Plug-in 형태로 내장되어 있다. Struts 없이도 Tiles를 독립적으로 적용할 수 있으며 이는 Apache Tiles 프로젝트 [<http://tiles.apache.org/>] (<http://tiles.apache.org/>)에서 확인할 수 있다.

4.2.1. Page Layout 구성 방법

4.2.1.1. 구성 방법

- JSP based approach : 페이지에 삽입할 기능이 많아질수록 복잡해짐. 소규모 어플리케이션에만 적합
- include 지시어 사용 : 중복되는 코딩 부분의 재사용. 여전히 페이지에서 콘텐츠와 레이아웃이 혼재됨.
- Template based approach : 페이지의 물리적인 영역을 은닉화하는 방법 제공. 콘텐츠와 레이아웃의 분리



위 그림은 Tiles를 적용하여 페이지 레이아웃을 구성한 예이다.

4.2.2. Tiles 설치

- struts-tiles.jar을 WEB-INF/lib 디렉토리에 복사한다.
- web.xml파일의 Action Servlet 정의 부분에 다음과 같이 추가한다.

```
<init-param>
  <param-name>chainConfig</param-name>
  <param-value>org/apache/struts/tiles/chain-config.xml</param-value>
</init-param>
```

- struts-config.xml의 plug-in에 TilesPlugin을 다음과 같이 등록한다.

```
<plug-in className="org.apache.struts.tiles.TilesPlugin">
  <set-property
    property="definitions-config"
    value="/WEB-INF/tiles-defs.xml"/>
</plug-in>
```

- Tiles를 사용하는 JSP에 다음과 같은 코드를 추가 한다.

```
<%@ taglib
    uri="http://struts.apache.org/tags-tiles" prefix="tiles" %>
```

4.2.3. Tiles 사용

4.2.3.1. Tiles 적용 시 고려점

- Tiles 단독으로도 사용이 가능하다.
- Tiles plug-in은 Tiles definitions를 사용할 때만 필요하다. (plug-in 설정 없이도 Tiles 라이브러리를 사용할 수 있음)
- Tiles definition은 JSP로 정의할 수도 있으나 일반적으로 layout과 Tiles definition (xml) 을 별도로 지정한다.
- Tiles는 기본적 Template 으로 많은 Layout을 제공하지만 개발에 들어가기 전에 Layout 에 대한 충분한 준비가 필요하다.

4.2.3.2. Tiles Tag Library의 속성

다음은 Tiles Tag Library의 태그들의 목록과 간단한 설명이다.

Attribute	Description
add	element를 추가
definition	titles component 정의
get	<template:put>을 통해 JSP로 전달된 자원을 얻는다.
getAsString	Tile/Component/Template 속성값을 JspWriter에 출력
importAttribute	정의된 context안에 타일의 속성을 추가한다.
initComponentDefinitions	정의 팩토리(definitions factory)를 초기화한다.
Insert	JSP 페이지 내에서 동적 템플릿을 추가
put	<template:insert>태그 내에서 템플릿에 삽입할 자원을 지정
putList	속성으로 전달할 리스트를 선언
useAttribute	JSP 페이지에서 속성 값을 사용한다.

보통 모든 페이지에서 적용되는 헤더나 저작 관련 내용들을 한곳에 모아 미리 Layout를 정의해둔다. 이를 Definition 이라 하는데 JSP 나 XML 로 만들 수 있으며 예는 다음과 같다.

4.2.4. Tiles Layout 정의

4.2.4.1. JSP 로 레이아웃을 정의한 예

다음은 storefront-defs.jsp 라는 jsp 에 tile definition을 정의한 예이다.

```
<%@ taglib
    uri="http://struts.apache.org/tags-tiles" prefix="tiles" %>
<tiles:definition id="storefront.default"
    pgae="/layouts/storefrontdefaultLayout.jsp" scope="request">
    <tiles:put name="header" value="/common/header.jsp"/>
    <tiles:put name="menubar" value="/common/menubar.jsp"/>
    <tiles:put name="copyright" value="/common/copyright.jsp"/>
</tiles:definition>
```

definition들을 이용하려면, 타일 컴포넌트들이 definition 에 접근할 수 있어야 한다. 다음은 definition을 사용하는 JSP 의 예이다. include 를 사용하여 storefront-defs.jsp 를 참조하고 있다.

```
<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles" %>
<%@ include file="../common/storefront-defs.jsp" %>

<tiles:insert beanName="storefront.default" beanScope="request">
  <tiles:put name="body-content" value="../security/signin-body.jsp"/>
</tiles:insert>
```

4.2.4.2.XML 로 레이아웃을 정의한 예

위에서는 tile definition 을 jsp 에 설정한 예를 보였지만, xml 로 정의하고 struts-config.xml 에 plug-in으로 정의해 놓고 쓰는 것이 일반적이다. 다음은 tiles-defs.xml의 일부분이다. index 라는 definition을 기본으로 하여 extends 해서 사용함으로 반복된 코딩을 줄여준다.

```
<!-- Doc index page description -->
<definition name="index" path="/layout.jsp">
  <put name="title" value="Anyframe Sample" />
  <put name="header" value="/header.jsp" />
  <put name="menu" value="/menu.jsp" />
  <put name="body" value="/body.jsp" />
  <put name="footer" value="/bottom.jsp" />
</definition>

<!-- view order information page description -->
<definition name="list" extends="index">
  <put name="body" value="/user/listUser.jsp" />
</definition>
```

Tiles를 사용하는 경우 action-mapping 에서는 forward path를 Tiles definition에 정의된 definition name으로 주어야 한다.

```
<action
  name="userForm"
  path="/listUser"
  type="org.anyframe.sample.struts.action.user.GetUserListAction"
  scope="request"
  validate="false"
  roles="admin,user">
  <forward name="success" path="list" />
</action>
```

forward 의 path 부분에 tiles 의 Definition name 이 들어가면 '/'를 사용하지 않음에 유의한다.

5.Internationalization

국제화(Internationalization, I18N)란 미리 다양한 언어와 지역을 지원하도록 소프트웨어를 설계하고 쉽게 리엔지니어링을 할 수 있게 지원하는 일련의 과정이다.

5.1.Internationalization의 특징

5.1.1.Internationalization의 필요성

- 코드 수정 없이 지원하는 언어를 추가
- 텍스트 요소들, 메시지들, 이미지 소스들과 소스코드의 분리
- 날짜,시간,숫자, 통화 등과 같은 문화에 종속적인 데이터들을 언어와 위치에 따라 달리 포매팅
- 비 표준 문자 집합들을 지원
- 어플리케이션을 새로운 언어와 지역에 쉽고 빠르게 적용

어플리케이션이 국제화를 지원한다고 해서 모든 언어와 지역에서 곧바로 쓸 수 있는 것을 의미한다는 것은 아니다. 국제화를 지원한다는 것은 언어나 지역에 영향을 받는 부분과 영향을 받지 않는 코드를 분리하여 쉽게 지역화될 수 있게 만들었다는 것을 의미한다.

5.1.2.지역 (Locale)

- 관습과 문화, 언어를 공유하는 영역을 의미함
- 지역화(L10N, Localization)는 제대로 국제화된 어플리케이션을 특정 지역에 맞게 만드는 일련의 과정

국제화 지원 시 위의 특징들 중 일부만 골라서 구현하는 것은 별 의미가 없다. 모두 지원하든지 지원하지 않든지 둘중의 하나이다. cf.) Anyframe 에서는 모든 국제화 문제를 고려할 수는 없고 Core Class 인 java.util.Locale, java.util.ResourceBundle 위주로 논의되었다. Anyframe 에서 제공하는 리소스들은 텍스트와 이미지에 초점이 맞추어져 있다. MessageResource Bundle은 PropertyResourceBundle Class 의 규약에 따라 만들어야 한다. 확장자 .properties 의 텍스트 파일로, 메시지들은 key=value와 같은 형식으로 작성해야 함은 이미 말한 바 있다. 또한 클래스처럼 찾아오기 때문에 클래스패스 상에 작성해야 한다. MessageResource Bundle 이 여러 개 있을 경우 Struts는 번들 중 하나에서 메시지를 찾을 때 기본 이름과 지역을 사용하여 이름이 가장 가깝게 일치하는 번들을 찾는다. 찾는데 실패하면 기본 번들을 사용한다. 다음은 메시지 정의의 예이다.

```
# error message
common.msg.authorization.error=You can not access this page.
..
# text resource
common.ui.title=eMarketPlace
..
```

다음은 ActionMessage 생성 시 메시지 키를 설정하는 예이다.

```
erros.add(ActionErrors.GLOBAL_ERROR, new ActionMessage("common.msg.authorization.error"));
```

다음은 Bean 태그 라이브러리의 MessageTag 를 사용하여 해당 메시지 키에 대한 실제 텍스트 value 를 표시하는 예이다.

```
<head>
```

```
<title><bean:message key="common.ui.title"/></title>
</head>
```

5.2. Internationalization Sample

다음은 사용자의 Locale정보에 의해 JSP페이지에 Message가 다른 언어로 보여지는 예이다.

5.2.1. Sample

- JSP

JSP화면에서 해당 언어를 클릭하면 MessageResource Bundle에 등록되어 있는 msg.internationalization키에 대한 값이 "Internationalization"과 "국제화"로 바뀌는 것을 보여주는 internationalization.jsp 이다.

```
<%@ page contentType="text/html; charset=utf-8" %>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<html>
<head>
<title>Internationalization Sample</title>
</head>
<body>
<strong>Change Language | 언어 변경 </strong><br/>
<html:link action="/internationalizationSample?language=en">English | 영어</
html:link><br/>
<html:link action="/internationalizationSample?language=ko">Korean | 한국어</
html:link><br/>
<bean:message key="msg.internationalization"/>
</body>
</html>
```

- Action

아래는 위의 JSP화면에서 해당 언어를 클릭했을 때 Action에서 Session에 사용자 Locale정보를 설정하는 예를 보여주는 InternationalizationAction.java 의 일부이다.

```
public ActionForward execute(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response)
    throws Exception {

    HttpSession session = request.getSession();
    Locale locale = getLocale(request);

    String language = null;
    String country = null;

    try {
        language = (String)
            PropertyUtils.getSimpleProperty(form, "language");
        country = (String)
            PropertyUtils.getSimpleProperty(form, "country");
    } catch (Exception e) {
        e.printStackTrace();
    }

    if ((language != null && language.length() > 0) &&
        (country != null && country.length() > 0)) {
        locale = new java.util.Locale(language, country);
    }
}
```

```
    } else if (language != null && language.length() > 0) {
        locale = new java.util.Locale(language, "");
    }
    session.setAttribute(Globals.LOCALE_KEY, locale);

    return mapping.findForward("success");
}
```

위 InternationalizationAction.java는 Struts에서 제공하고 있는 struts-examples-1.3.10의 LocaleAction을 수정한 것으로 화면에서 parameter값으로 보내온 language 값을 Globals.LOCALE_KEY 이름으로 Session에 저장한다.

6. Validator

사용자 입력 값을 검증하는 여러가지 방법 중에서 일반적으로 많이 사용하고 있는 방법이 JSP페이지 내에서 javascript 함수를 이용해 사용자 입력 값을 검증하는 방법이다. 하지만 이런 방법으로는 완전한 검증을 할 수 없기 때문에 Sever측에서 다시 한번 검증하는 것이 필요하다. Validator는 Struts 1.1부터 배포된 프레임워크로 ActionForm에 대한 유효성 검사를 편리하게 도와주고 있다. 본 문서에서는 Validator를 사용하기 위한 플러그인 등록 및 Struts에서 제공하고 있는 validator-rule에 대해서 소개하기로 한다.

6.1. Plug-in 등록

6.1.1. struts-config.xml에 plug-in 등록

Validator를 사용하기 위해서는 struts-config.xml에 org.apache.struts.validator.ValidatorPlugIn을 등록해야 한다. ValidatorPlugIn은 property로 pathname을 가지며 pathname의 값으로는 Struts에서 기본으로 제공하고 있는 검증 규칙이 정의되어 있는 validator-rules.xml과 사용자가 검증한 ActionForm에 대한 검증 규칙을 정의한 xml파일을 세팅한다.

6.1.2. Samples

다음은 Validator Plug-in을 설정한 struts-config-validator.xml 의 일부분이다.

```
<plug-in className="org.apache.struts.validator.ValidatorPlugIn">
  <set-property property="pathnames"
    value="/org/apache/struts/validator/validator-rules-compressed.xml,
    /WEB-INF/validator/validation-sample.xml" />
</plug-in>
```

Struts에서는 일반적으로 많이 사용하고 있는 검증 규칙에 대한 정의가 포함된 validator-rules.xml과 validator-rules-compressed.xml을 struts-core-1.3.x.jar파일에 같이 배포하고 있다. validation-sample.xml은 검증하고자 하는 ActionForm에 대한 formset 설정이 포함되어 있다.

6.2. Validator Rules

6.2.1. Struts Validator Rules 기본 기능

Struts에서 기본적으로 많은 검증 규칙을 제공하고 있기 때문에 사용자가 검증규칙을 새로 작성하는 번거로운 작업이 많이 줄어들었다. 다음은 validation rule 설정의 예이다.

```
<validator name="minlength"
  classname="org.apache.struts.validator.FieldChecks"
  method="validateMinLength"
  methodParams="java.lang.Object,
    org.apache.commons.validator.ValidatorAction,
    org.apache.commons.validator.Field,
    org.apache.struts.action.ActionErrors,
    javax.servlet.http.HttpServletRequest"
  depends="required"
  msg="errors.minlength">
  <javascript>[function validateMinLength(form) {
  // javascript를 사용하고 자 할 경우 여기에 작성하면 된다.
  }]
```

```
</javascript>
</validator>
```

name 은 unique 해야 하며 다른 rule에서 참조할 때도 name을 사용하게 된다. classname과 method 는 실제 validation 로직이 들어있는 클래스와 메소드의 이름이다. depends 는 validator rule 간의 우선 순위이다. 위의 예는 minlength를 체크하기 전에 required 라는 validation rule을 먼저 수행한다는 뜻이다. depends="required,integer"라고 되어있다면 null 인지 체크하고, Integer 인지 체크하고 그 다음에 minlength를 체크하겠다는 뜻이다. msg 는 validation error 가 발생했을 때 뿌려줄 메시지를 resource bundle 에서 가져올 때 사용할 key 값이다.

기본적으로 다음의 값들을 사용한다.

```
# Struts Validator Error Messages
errors.required={0} is required.
errors.minlength={0} can not be less than {1} characters.
errors.maxlength={0} can not be greater than {1} characters.
errors.invalid={0} is invalid.
errors.byte={0} must be a byte.
errors.short={0} must be a short.
errors.integer={0} must be an integer.
errors.long={0} must be a long.
errors.float={0} must be a float.
errors.double={0} must be a double.
errors.date={0} is not a date.
errors.range={0} is not in the range {1} through {2}.
errors.creditcard={0} is an invalid credit card number.
errors.email={0} is an invalid e-mail address.
```

다음은 validator-rules.xml에 포함되어 있는 검증 규칙에 대한 설명이다.

Name	Description	<var-name>
required	입력 값이 반드시 존재해야 한다.	-
validwhen	다른 필드의 값을 비교한다.	test
minlength	입력 값의 최소 글자수를 제한한다.	minlength
maxlength	입력 값의 최대 글자수를 제한한다.	maxlength
mask	설정된 regular expression을 만족 해야한다.	mask
date	Date형태로 변환 가능해야 한다.	datePatternStrice 또는 datePattern
creditCard	신용카드 번호 규칙에 만족해야 한다.	-
email	e-mail 규칙에 만족해야 한다.	-

6.3.ActionForm

6.3.1.ValidatorForm의 상속

Validator를 이용할 때는 ActionForm이 아닌 org.apache.struts.validator.ValidatorForm을 상속 받아야 한다. ValidatorForm은 유효성 검증에 필요한 validate() 메소드를 포함하고 있으며 검증 실패 시 ActionErrors에 error message를 셋팅해서 리턴한다.

6.3.2.Samples

다음은 ValidatorForm을 상속받은 ValidatorSampleForm.java 의 일부분이다.

```

public class ValidatorSampleForm extends ValidatorForm{

    private String userName;
    private String userId;
    private String password;
    private String phoneNumber;
    private String email;
    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }
    //종락
}
}

```

6.4.formset 설정

6.4.1.formset 설정 방법

validator-rules.xml에 등록된 검증 규칙과 ActionForm과의 매핑을 하기 위해서는 formset을 정의해야 한다. 다음은 <form>의 attribute들이다.

Attribute	Description
property	ActionForm subclass의 property에 해당한다.
depends	하나 이상의 Validation rule들을 지정한다
page	Multi-page ActionForm에서 페이지 번호를 지정할 때 사용
indexedListProperty	ActionForm 에서 Collection을 return하는 property name

6.4.2.Sample

다음은 ValidatorSampleForm에 Struts에서 제공한 검증규칙을 적용하기 위해 설정한 validation-sample.xml 의 일부이다.

```

<form name="validatorSampleForm">
  <field property="userName" depends="required,minlength,maxlength">
    <arg key="validator.sample.userName" position="0"/>
    <arg name="minlength" key="{var:minlength}" resource="false" position="1"/>
    <arg name="maxlength" key="{var:maxlength}" resource="false" position="1"/>
    <var>
      <var-name>minlength</var-name>
      <var-value>4</var-value>
    </var>
    <var>
      <var-name>maxlength</var-name>
      <var-value>10</var-value>
    </var>
  </field>
  <!-- 종락 -->
  <field property="email" depends="email">
    <arg key="validator.sample.email" position="0"/>
  </field>
</form>

```

<form>의 name attribute에 struts-config.xml에 등록된 ActionForm의 form name을 등록하고 <form>의 하위의 <field>에 ActionForm의 유효성 검사할 attribute에 대한 설정을 등록한다. 위의 Sample에서는 userName은 필수 입력값에, 4자리 이상 10자리 이하로 설정을 했고 email은 e-mail 주소 검증 규칙을 적용했다. 유효성 검증에 실패하면 arg 값을 에러 메시지의 arguments로 설정하여 돌려주게 된다. 기본으로 arg0-arg3 요소는 message resources에서 해당 key 값으로 찾게되고. resource 속성이 false로 설정되면 message resources에서 찾지 않고 해당 값을 바로 돌려준다. validation.xml은 크게 global 과 formset 으로 나누는데 global은 file 내에서 상수로 사용될 값들을 정의한다. formset은 다시 constant 와 form으로 나누는데 constant는 global과 같으며 form은 validation을 적용할 ActionForm 에 매핑된다. 참고로 formset 에는 language와 country라는 attribute 가 있어서 다국어 지원이 가능하다.

6.5.Action 매핑 설정

6.5.1.struts-config.xml의 Action 매핑 설정

RequestProcessor가 들어온 Request 정보를 이용해 ActionForm을 생성할 때 유효성 검증을 실행하기 위해서는 struts-config.xml의 <action>에 **validate="true"**를 설정해야 한다.

6.5.2.Sample

다음은 struts-config.xml의 <action>에 Validator를 적용하기 위한 설정을 보여주는 struts-config-validator.xml 의 일부이다.

```
<action
  path="/validatorSample"
  type="org.anyframe.sample.struts.basic.ValidatorSampleAction"
  validate="true"
  cancelable="true"
  scope="request"
  input="/basic/validatorSample.jsp"
  name="validatorSampleForm">
  <forward name="success" path="/basic/validatorSuccess.jsp"></forward>
</action>
```

7.Exception Handling

[Action클래스의 execute() 메소드에서 Exception이 전달되었을 때 실행하는 ExceptionHandler를 설정할 수 있다. struts-config.xml에 개별 Action에 대한 ExceptionHandler를 설정할 수도 있고(action level), <global-exceptions>을 이용해 application의 모든 Action에서 발생하는 Exception에 대한 ExceptionHandler를 설정(global level)할 수도 있다. 양쪽 모두에 ExceptionHandler가 정의되었을 때 action level에 선언된 것이 더 우선순위가 높다.

7.1.Global Level Exception Handling

7.1.1.Global Level Exception Handling의 특징

- 에러나 Exception 처리를 정의하는 선언적인 방법
- 모든 action에서 사용할 수 있는 global level Exception 처리를 정의
- 하위로 여러 개의 <exception>을 가질 수 있음

다음은 <exceptions>의 attribute들이다.

Name	Description
bundle	Exception의 handler 클래스가 사용하는 message resources bundle에 대한 servlet context attribute의 name 값이다. 디폴트 값 : org.apache.struts.Globals.MESSAGES_KEY의 값
className	Exception들의 configuration 정보를 담고 있을 객체이다. 반드시 org.apache.struts.config.ExceptionConfig 또는 이를 상속 받은 클래스여야 한다. 디폴트 값 : org.apache.struts.config.ExceptionConfig
handler	이 exception이 발생할 때, 이 exception을 처리하는 클래스를 나타낸다. 즉, handler 클래스는, 어떤 exception이 발생하면, 적절한 error message('key' attribute)와 함께 적절한 페이지('path' attribute)로 forward 해주는 클래스이다. 반드시 org.apache.struts.action.ExceptionHandler 또는 이를 상속 받은 클래스여야 한다. 디폴트 값 : org.apache.struts.action.ExceptionHandler
key	이 exception이 발생할 때, message resource bundle에서 찾아낼 error message의 key 값이다. [required]
path	이 exception이 발생할 때, forward할 resource(*.do, *.jsp 등)의 상대(module-relative)경로를 나타낸다.
scope	ActionError 객체에 접근할 context의 scope를 나타낸다. request 또는 session. 디폴트 값 : request
type	Exception Handling을 수행할 exception의 type을 나타낸다. [required]

7.1.2.Samples

다음은 struts-config.xml 파일에서 global-exceptions 설정에 대한 예제이다.

```
<global-exceptions>
  <exception key="global.exception.message"
    path="/basic/globalException.jsp"
    type="java.lang.Exception"
    handler="org.apache.struts.action.ExceptionHandler" />
</global-exceptions>
```

Action 클래스의 execute() 메소드에서 Exception이 발생하면 ExceptionHandler에서 exception message key값(global.exception.message)을 이용해 Resource Bundle에서 Exception Message를 세팅한 후 path에 설정된 /basic/globalException.jsp로 forwarding한다.

7.2.Action Level Exception Handling

7.2.1.Action Level Exception Handling의 특징

- 개별 Action에 대한 Exception Handling이 가능
- Action Level Exception이 정의되어 있지 않을 경우 Global Level Exception 적용
- <action>하위에 <exception>으로 정의

다음은 <exception>의 attribute들이다.

Name	Description
bundle	Exception의 handler 클래스가 사용하는 message resources bundle에 대한 servlet context attribute의 name 값이다. 디폴트 값 : org.apache.struts.Globals.MESSAGES_KEY의 값
className	Exception들의 configuration 정보를 담고 있을 객체이다. 반드시 org.apache.struts.config.ExceptionConfig 또는 이를 상속 받은 클래스여야 한다. 디폴트 값 : org.apache.struts.config.ExceptionConfig
handler	이 exception이 발생할 때, 이 exception을 처리하는 클래스를 나타낸다. 즉, handler 클래스는, 어떤 exception이 발생하면, 적절한 error message('key' attribute)와 함께 적절한 페이지('path' attribute)로 forward 해주는 클래스이다. 반드시 org.apache.struts.action.ExceptionHandler 또는 이를 상속 받은 클래스여야 한다. 디폴트 값 : org.apache.struts.action.ExceptionHandler
key	이 exception이 발생할 때, message resource bundle에서 찾아 낼 error message의 key 값이다. [required]
scope	ActionError 객체에 접근할 context의 scope를 나타낸다. request 또는 session. 디폴트 값 : request
type	Exception Handling을 수행할 exception의 type을 나타낸다. [required]

7.2.2.Samples

다음은 struts-config-login.xml 파일에서 <action> 하위의 <exception> 설정에 대한 예제이다.

```
<action
  path="/login"
  type="org.anyframe.sample.struts.action.LoginAction"
  name="userForm"
  scope="request"
  input="/basic/login.jsp">
  <exception key="error.password.mismatch" path="/basic/login.jsp"
    type="javax.security.auth.login.FailedLoginException" />
  <forward name="success" path="/basic/main.jsp" />
</action>
```

Global Level Exception 설정과 달리 모든 Action에서 발생하는 Exception이 아닌 LoginAction의 execute() 메소드에서 발생하는 Exception에 대한 처리를 담당한다. FailedLoginException이 발생했을 경우 /basic/login.jsp로 forwarding되고, 이 외 다른 Exception이 발생할 경우 Global Level Exception에 설정된 error page로 forwarding된다.

III.Struts Extensions

Struts Extensions은 Apache Struts Framework를 실제 프로젝트에서 보다 편리하게 사용하기 위해 추가적인 기능들을 제공한다. Role기반의 인증 및 관리 기능 등이 추가 된 Struts의 TilesRequestProcessor를 확장한 DefaultRequestProcessor, 선언적인 Synchronized Token 처리 기능 등이 추가된 AbstractActionSupport 외에 다양한 Controller클래스와 Exception 처리를 위한 DefaultExceptionHandler 등을 제공하고 있다. 그 밖에 Ria Solution인 MiPlatform을 이용해 화면 개발 시 공통적인 로직을 담당하고 있는 AnyframeMiPAction클래스가 포함되어 있다.

8.Controller

Struts의 Controller는 크게 ActionServlet, RequestProcessor, Action 클래스로 구성된다. Anyframe 에서는 Struts Controller의 기본 기능과 Business Service연계, 로깅, 인증 및 권한 처리, Exception 처리 등을 위한 확장 기능을 제공하고 있다. 각각에 대한 내용은 아래와 같다.

8.1.DefaultActionServlet

<servlet-class>는 org.apache.struts.action.ActionServlet을 extends한 org.anyframe.struts.action.DefaultActionServlet 으로 정의한다. <init-param>은 ActionServlet이 제공하는 기본 기능과 Character Encoding을 설정 할 수 있다. Character Encoding 속성은 DefaultActionServlet 에서 확장한 것으로 정의된 Message Resource Bundle을 읽어들이기 때, 적용할 Character Encoding을 설정하기 위한 것이다. 다음은 DefaultActionServlet을 <servlet>으로 설정한 web.xml 의 일부이다.

```
<servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>
    org.anyframe.struts.action.DefaultActionServlet
  </servlet-class>
  <init-param>
    <param-name>config</param-name>
    <param-value>
      /config/struts/struts-config.xml,
      /config/struts/struts-config-login.xml,
      /config/struts/struts-config-dispatch.xml,
      /config/struts/struts-config-token.xml,
      /config/struts/struts-config-exception.xml,
      /config/struts/struts-config-authorization.xml
    </param-value>
  </init-param>
  <init-param>
    <param-name>character-encoding</param-name>
    <param-value>utf-8</param-value>
  </init-param>
  <init-param>
    <param-name>debug</param-name>
    <param-value>3</param-value>
  </init-param>
  <init-param>
    <param-name>detail</param-name>
    <param-value>3</param-value>
  </init-param>
  <init-param>
    <param-name>convertNull</param-name>
    <param-value>true</param-value>
  </init-param>
  <load-on-startup>0</load-on-startup>
</servlet>
```

8.2.DefaultRequestProcessor

org.anyframe.struts.tiles.DefaultRequestProcessor는 Struts의 TilesRequestProcessor를 extends하고 있다. 따라서, DefaultRequestProcessor를 Struts 속성 정의 파일에 controller로써 설정한 경우에는 반드시 plug-in으로 TilesPlugin을 등록해 주어야 한다. TilesPlugin을 plug-in으로 등록하는 방법은 Struts Tiles의 Tiles 설치를 참고한다. 아래는 struts-config-common.xml 의 일부로, DefaultRequestProcessor를 <controller> 내에 정의하고 있다.

```
<controller contentType="text/html;charset=utf-8" locale="true"
    processorClass="org.anyframe.struts.tiles.DefaultRequestProcessor"/>
```

8.2.1.DefaultRequestProcessor 기능

- role 기반의 인증 및 관리 기능
- web.xml에서 설정한 character-encoding 값 적용
- Locale 정보를 Session에 org.apache.struts.action.LOCALE이란 key 값으로 저장

아래는 인증 및 권한 처리를 수행하고 있는 DefaultRequestProcessor의 processRoles 메소드 구현 로직의 일부이다.

```
protected boolean processRoles(HttpServletRequest request,
    HttpServletResponse response, ActionMapping mapping)
    throws IOException, ServletException {
    // [public] Is this action protected by role requirements?
    String roles[] = mapping.getRoleNames();
    if ((roles == null) || (roles.length < 1)) {
        return (true);
    }

    Subject _subject = null;

    HttpSession session = request.getSession();
    _subject = (Subject) session.getAttribute("subject");

    if (_subject == null) {
        log.debug("#AuthenticationException is encountered");

        ExceptionConfig config = mapping.findException(AuthenticationException.class);

        if(config == null ){
            mapping.findException(Exception.class);
        }

        AuthenticationException ae = new AuthenticationException(config.getKey(), request
            .getRequestURI());
        // 종략 ...
    }
}
```

위의 소스 코드에서 보듯이 Struts 속성 정의 파일의 특정 action 매핑 정보에 role이 부여되었을 때 Session에 저장된 Subject에서 사용자 인증 정보를 확인한 후, 인증되지 않았을 경우 AuthenticationException을 throw한다. Exception 메시지의 key는 Struts 속성 정의 파일 <exception> 내에 정의된 AuthenticationException에 대한 key와 동일하며, 정의되지 않았을 경우엔 java.lang.Exception에 대한 key와 동일하다. 아래는 struts-config-exception.xml 파일의 일부로 AuthenticationException을 Global Level Exception으로 등록한 예이다.

```
<global-exceptions>
    <exception key="common.msg.authentication.error"
        path="/WEB-INF/jsp/struts/common/error.jsp"
        type="org.anyframe.struts.util.AuthenticationException"
        handler="org.anyframe.struts.action.DefaultExceptionHandler" />
</global-exceptions>
```

메시지리소스로 등록하는 메시지 properties 파일에는 common.msg.authentication.error 의 메시지 키에 대하여

```
common.msg.authentication.error=Authentication Fail
```

```
- You are not logon or Session expired. Please try re-logon. - {0}.
common.msg.authorization.error=You can not access this page. - {0}.
..
```

과 같이 메시지 파일이 등록되어 있음을 가정한다.

8.3.AbstractActionSupport

org.anyframe.struts.action.AbstractActionSupport 클래스는 다음과 같은 주요 기능을 제공한다.

- Spring 기반의 Anyframe 서비스와의 손쉬운 연동 지원
- 선언적인 Synchronized Token 처리
- 공통 Exception 처리

따라서, 각 Action 클래스는 AbstractActionSupport를 상속받아 구현하되, process 메소드를 오버라이드 하여 비즈니스 레이어와 연계하여 클라이언트의 요청을 처리하는 로직을 담도록 한다. 위와 같은 주요 기능을 제공하는 AbstractActionSupport의 execute 메소드는 다음과 같이 구현되어 있다.

```
public ActionForward execute(ActionMapping mapping, ActionForm form,
                            HttpServletRequest request, HttpServletResponse response)
                            throws Exception {

    ActionForward forward = null;

    try {
        preProcess(mapping, form, request, response);

        getLogger().debug(this.getClass().getName() + ".process() Started!");
        forward = process(mapping, form, request, response);
        getLogger().debug(this.getClass().getName() + ".process() Ended!");
        forward = postProcess(mapping, form, request, response, forward);
    } catch (InvalidTokenException tokenException) {
        forward = processInvalidTokenException(mapping, form, request,
            response, tokenException);
    } catch (RuntimeException uncheckedException) {
        forward = processUncheckedException(mapping, form, request,
            response, uncheckedException);
    } catch (Exception checkedException) {
        getLogger().debug("\n Action Support Exception catch!!");
        forward = processCheckedException(mapping, form, request, response,
            checkedException);
    } finally {
        forward = processFinally(mapping, form, request, response, forward);
    }

    return forward;
}
```

다음 목록에 제시된 메소드들은 AbstractActionSupport 클래스 내에 구현된 메소드들로서, execute 메소드의 로직을 수행하기 위해 적절한 순서에 따라 호출된다.

- preProcess : AbstractActionSupport 클래스를 상속받은 Action 클래스의 process 메소드 수행 전에 호출되는 메소드로서 Action 매핑 정보(validateToken, resetToken)에 기반하여 Token의 유효성을 체크한다. 해당 Action을 수행하기 위한 precondition이 필요할 경우 이 메소드를 오버라이드하면 된다.
- process : abstract 메소드이다. 따라서, AbstractActionSupport를 상속받은 하위 Action 클래스에서 반드시 구현해야 하며, process 메소드 내에는 비즈니스 레이어와 연계하여 클라이언트의 요청을 처리하는 로직을 담는다.

- `postProcess` : `AbstractActionSupport` 클래스를 상속받은 `Action` 클래스의 `process` 메소드 수행 후 호출되는 메소드로서 `Action` 매핑 정보(`saveToken`)에 기반하여 `Token`을 생성한다. 해당 `Action`을 수행하기 위한 `postCondition`이 필요할 경우 이 메소드를 오버라이드하면 된다.
- `processInvalidTokenException` : `Synchronized Token` 사용시 `Token`이 유효하지 않을 경우에 대한 처리 로직을 담고 있다. "요청이 올바르지 않습니다."라는 메시지를 담은 `ActionMessage`를 생성하고, `InvalidTokenException`을 throw한다.
- `processUncheckedException` : `preProcess()`, `process()`, `postProcess()` 수행시 `RuntimeException`이 발생한 경우, 해당 `Exception`을 throw한다. `UncheckedException` 발생시 별도 처리 로직이 필요한 경우 이 메소드를 오버라이드하면 된다.
- `processCheckedException` : `preProcess()`, `process()`, `postProcess()` 수행시 `Exception`이 발생한 경우, 해당 `Exception`을 throw한다. `CheckedException` 발생시 별도 처리 로직이 필요한 경우 이 메소드를 오버라이드하면 된다.
- `processFinally` : `AbstractActionSupport` 클래스 `execute` 메소드의 `finally` 구문에서 호출되는 메소드이다. `finally` 구문에서 별도 처리 로직이 필요한 경우 이 메소드를 오버라이드하면 된다.

위에서 제시한 `AbstractActionSupport`의 기본 제공 기능 이외에 각 `Action` 클래스에서 처리해야 할 공통 기능이 필요할 경우, `AbstractActionSupport`를 상속받은 클래스를 생성하고, 해당 클래스에서 필요한 기능을 추가하도록 한다. 그리고 각 `Action` 클래스는 `AbstractActionSupport`를 상속받은 클래스를 상속받아 구현하도록 한다.

8.3.1.Action Sample

다음은 `AbstractActionSupport`를 상속받아 구현한 `LoginAction.java` 이다.

```
public class LoginAction extends AbstractActionSupport {

    public Log getLogger() throws Exception {
        return LoggerFactory.getLog(this.getClass().getName());
    }

    public ActionForward process(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response)
        throws Exception {
        AuthenticationService authenticationService =
            (AuthenticationService) getService("authenticationService");

        UserForm userForm = (UserForm) form;
        UserVO userVO = new UserVO();
        BeanUtils.copyProperties(userVO, userForm);

        Subject subject = authenticationService.authenticate(userVO);

        HttpSession session = request.getSession();

        session.setAttribute("subject", subject);
        return (mapping.findForward("success"));
    }
}
```

위의 소스코드에서는 `LoginAction` 클래스에서 개별 `Logger`를 사용하기 위해 `AbstractActionSupport`의 `getLogger` 메소드를 오버라이드하고 있다.

8.4.DefaultDispatchActionSupport

org.anyframe.web.struts.action.DefaultDispatchActionSupport은 앞서 언급한 AbstractActionSupport를 상속받아 구현한 클래스로써 Struts에서 기본으로 제공하는 DispatchAction과 동일한 기능을 제공한다.

8.4.1.Action Sample

다음은 DefaultDispatchActionSupport를 상속받아 구현한 ProductAction.java 이다.

```
public class ProductAction extends DefaultDispatchActionSupport {

    public ActionForward get(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response)
        throws Exception {

        // TODO : 단건 조회 기능 관련 토직

        return mapping.findForward("success_get");
    }

    public ActionForward list(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response)
        throws Exception {

        // TODO : 리스트 조회 기능 관련 토직

        return mapping.findForward("success_list");
    }
}
```

DefaultDispatchActionSupport를 상속한 Action 클래스는 DispatchAction이므로, AbstractActionSupport를 상속한 Action 클래스 작성과 다르게 한 Action 내에 여러 메소드 정의가 가능하다.

8.5.DefaultForwardAction

org.anyframe.web.struts.common.action.DefaultForwardAction은 Struts에서 기본으로 제공하는 ForwardAction과 동일한 기능을 수행한다. 단, 차이점은 DefaultForwardAction은 AbstractActionSupport를 상속받았기 때문에 공통 Exception 처리, Synchronized Toke 처리 등이 가능하다는 장점이 있다. 아래는 DefaultForwardAction을 이용해 Action 매핑을 정의한 struts-config-dispatch.xml 파일의 일부이다. 요청 URL이 dispatchActionView.do일 경우 DefaultForwardAction을 통해 /extensions/dispatch.jsp 페이지로 이동할 것이다.

```
<action path="/dispatchActionView"
    type="org.anyframe.struts.action.DefaultForwardAction" parameter="/extensions/
dispatch.jsp" />
```

8.6.AnyframeMiPAction

프리젠테이션 레이어 구성시 X-Internet 제품인 MiPlatform을 이용하는 경우 MiPlatform에서 다루는 고유한 형태의 데이터를 쉽게 처리할 수 있도록 하기 위해 org.anyframe.struts.action.mip.AnyframeMiPAction 클래스를 제공한다. 개발자는 AnyframeMiPAction을 상속받아 개별 Action을 개발하되, process 메소드를 구현해주도록 한다.

process 메소드는 MiPlatform과 관련하여 다음과 같은 입력 인자를 가진다.

Type	Paramter Name	Description
VariableList	inVI	Client에서 GET 방식으로 전송한 parameter들 포함
VariableList	outVI	Client로 전송하는 VariableList
DatasetList	inDI	Client에서 POST 방식으로 전송한 Dataset XML 포함
DatasetList	outDI	Client로 전송하는 Dataset XML 설정

8.6.1. Sample Action

다음은 AnyframeMiPAction을 상속받아 구현한 클래스의 일부로, process 메소드 내부에서 비즈니스 서비스를 실행하고, 그 결과값을 반환하고 있다.

```
public void process(ActionMapping mapping, PlatformRequest request,
    variableList inVI, DatasetList inDI, variableList outVI,
    DatasetList outDI) throws Exception {
    this.inVI = inVI;
    this.inDI = inDI;
    this.outVI = outVI;
    this.outDI = outDI;

    MiPUserService userService = (MiPUserService)getService("userService");
    Dataset ds = userService.getUserList(inVI);
    outDI.addDataset("ds_access", ds);

    // 종략 ...
}
```

9.View

Anyframe 에서는 개발자들이 보다 view 개발을 쉽게 할 수 있도록 Custom tag library를 제공한다. 이런 Custom tag library에는 Spring의 message 태그를 utf-8/euc-kr의 인코딩된 한글 메시지를 위해 확장한 Anyframe message 태그와 페이지 네비게이션을 JSP단의 java 코드 없이 태그로 개발할 수 있는 Page Navigator 태그가 있다.

9.1.Tag library

Anyframe 에서는 개발자들이 자바 코드를 사용하지 않고 보다 쉽게 JSP 페이지를 구현할 수 있도록 다음과 같은 Anyframe Tag Library를 제공한다.

9.1.1.Page Navigator Tag

Anyframe 에서는 Page 처리에 대한 구현이 편리하도록 page 관련 Tag Library인 Page Navigator Tag를 제공한다. 이 태그를 사용하면 리스트 화면을 출력할 때 Tag Library를 사용하여 간단히 Page Navigator를 출력해줄 수 있다. 이 태그를 사용하기 위해 JSP의 상단에 다음과 같이 anyframe-pagenavigator.tld 파일을 taglib으로 지정해 준다.

```
<%@ taglib uri="http://www.anyframejava.org/tags" prefix="anyframe" %>
```

prefix를 'anyframe'으로 정의할 경우 아래와 같이 태그를 사용할 수 있다.

```
<anyframe:pagenavigator linkUrl="javascript:fncGetUserList(2);"
    pages="<%=resultPage%>" formName="listForm"
    firstImg="sample/images/ct_btn_pre.jpg"
    prevImg="sample/images/ct_btn_pre01.jpg"
    lastImg="sample/images/ct_btn_next.jpg"
    nextImg="sample/images/ct_btn_next01.jpg" />
```

anyframe을 prefix로 하는 태그로 tag name은 pagenavigator이다 . 이 때 pages라는 attribute는 반드시 org.anyframe.pagination.Page 타입의 객체를 설정해줘야 함에 유의하도록 한다.

10.Preventing Double Form Submission

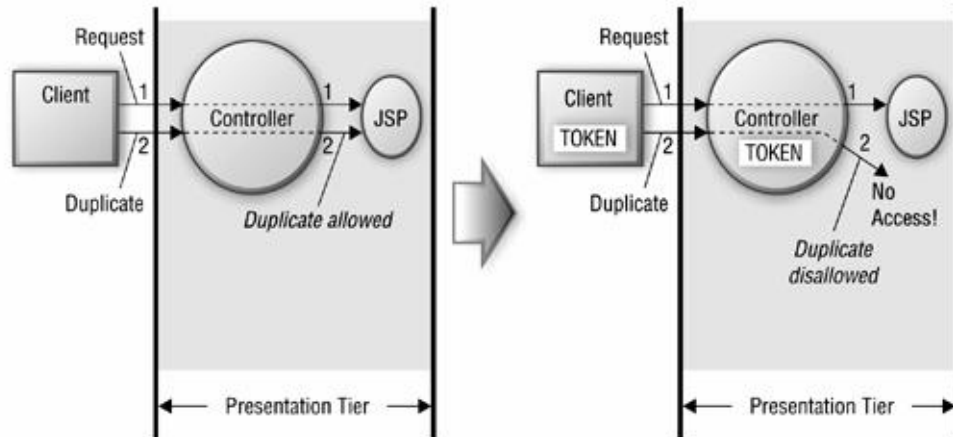
Anyframe의 Struts Extensions에서는 Synchronized Token이라는 진보된 방법을 통해 선언적으로 중복 Form submit으로 인한 오동작을 방지할 수 있는 기능을 제공한다.

10.1.Double Submit의 개념

중복 Form Submit은 다음과 같은 경우에 발생할 수 있다.

- Browser의 Refresh Button을 사용한 경우
- Browser의 Back Button을 사용하여 이전 Page로 이동 후 다시 Form Submit 하는 경우
- Browser History을 사용하여 이전 Page로 이동 후 다시 Form Submit 하는 경우
- 서버에 영향을 주도록 고의적으로 악의적인 Form Submit을 하는 경우
- submit Button을 한번 이상 Click 하는 경우

Browser Refresh Button을 사용한 경우에 중복 submit이 발생하는 이유는 Form Submit 이후에 Browse 주소창에 보이는 URL에 있다. 예를 들어, "<form name='test' action='/submitForm.do'>" 을 통해 Form Submit이 발생하였다라고 가정하자. 이 Form이 전송된 이후에 주소창에 '/submitForm.do'이 남아있고, 이 상황에서 Refresh Button을 누르면 동일한 URL이 재전송 되는 것이다. 이러한 상황을 막는 가장 손쉬운 방법은 Form Submit 후에 HTTP Redirect 기능을 사용하는 것이다. 만일 test Form 전송 후에 보여 주는 Page가 success.jsp 라고 하면, HTTP Redirect를 사용할 경우 Browse 주소창에 success.jsp가 보일 것이다. 이 경우에는 Refresh Button을 눌러도 success.jsp가 다시 로드된다. <forward name="success" path="/Success.jsp" redirect="true" /> 이렇게 함으로써 우연히 Browser Refresh Button을 눌렀을 때의 동작 오류를 방지할 수 있다. 하지만 Browser Back Button 등을 사용한 Form Resubmit을 근원적으로 막지는 못하게 된다. 따라서, Anyframe 에서는 DefaultActionMapping과 AbstractActionSupport를 제공함으로써, 선언적인 기법으로 Synchronized Token을 처리하여 중복 Form Submit을 방지할 수 있게 한다. 다음은 Synchronized Token을 이용한 중복 Form Submit 방지 개념도이다.



10.2.일반적인 Token 처리

일반적으로 Action 클래스에서는 다음과 같은 로직을 통해 중복 Form Submit 여부를 체크할 수 있다.

- Action Class

```
boolean valid = isValid(request, true);  
if (valid) {
```

```
// TODO: submit 할 때 수행할 토직을 넣을 것
System.out.println("status: performed");
} else {
// TODO: init / reload 할 때 수행할 토직을 넣을 것
System.out.println("status: initialized or reloaded");
}
saveToken(request);
```

- JSP

```
<input type="hidden" name="org.apache.struts.taglib.html.TOKEN"
value="<%= session.getAttribute(org.apache.struts.Globals.TRANSACTION_TOKEN_KEY) %>">
```

UI(JSP, HTML)에서는 "org.apache.struts.taglib.html.TOKEN"을 Key로 하는 Hidden Field를 통해 할당된 Token 값을 서버로 전송하고, 해당 Action 클래스에서는 isValid 메소드 호출을 통해 이 Token 값과 Session에 저장된 Token 값을 비교함으로써, Token의 유효성을 검사한다.

10.3.선언적인 Token 처리

중복 Form Submit 방지 처리가 필요한 모든 Action과 JSP에 Token 처리를 위한 로직이 중복 구현되지 않도록 하기 위해, Anyframe 에서는 AbstractActionSupport 클래스와 DefaultActionMapping 클래스를 이용하여 선언적으로 Synchronized Token을 처리할 수 있는 기능을 제공한다. 선언적인 Synchronized Token 처리를 위해서는 Struts 속성 정의의 파일 내에 Action 매핑시 다음과 같은 속성을 추가 정의해 주어야 한다.

- **saveToken** : 해당 Action 수행 후, Client에서 전달한 Token을 Session에 저장할지에 대한 설정
- **validateToken** : 해당 Action 수행 전, Token의 유효성을 체크할지에 대한 설정
- **resetToken** : Token의 유효성을 체크한 후에 Session에 저장된 Token을 reset할지에 대한 설정

아래는 struts-config-token.xml 파일의 일부로, 선언적인 기법으로 중복 Form Submit을 방지하는 예제이다.

```
<action-mappings type="org.anyframe.struts.action.DefaultActionMapping">
  <action path="/synchronizedTokenView"
    type="org.anyframe.struts.action.DefaultForwardAction"
    parameter="/extensions/token.jsp">
    <set-property property="saveToken" value="true" />
  </action>
  <action path="/doubleSubmit"
    type="org.anyframe.sample.struts.action.SampleTokenAction"
    name="submitForm" scope="request">
    <set-property property="validateToken" value="true" />
    <set-property property="resetToken" value="true" />
    <forward name="success" path="/extensions/tokenSuccess.jsp" />
  </action>
</action-mappings>
```

각 Action 매핑시 정의한 위의 세가지 속성이 AbstractActionSupport 클래스를 통해 어떻게 처리되는지에 대해서는 다음 예를 통해 상세히 살펴보자.

10.3.1.Samples

1. 중복 Form Submit 방지가 필요한 UI의 경우, 해당 UI를 로드시키기 위한 Action 실행을 통해 Token을 Session에 저장해야 한다. 따라서 해당 Action 매핑 정의시 saveToken 속성값을 true로 정의해 주도록 한다.

```
<set-property property="saveToken" value="true"/>
```

2. 해당 Action 클래스의 process 메소드 수행후, AbstractActionSupport의 postProcess 메소드에서는 Action 매핑시 정의한 saveToken이 true일 때 상위 클래스에서 제공하는 saveToken 메소드를 호출 함으로써, 이 Token을 Session에 저장한다.

```
public ActionForward postProcess(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response,
    ActionForward forward) throws Exception {
    boolean saveToken = false;
    if (mapping instanceof DefaultActionMapping) {
        DefaultActionMapping t_mapping = (DefaultActionMapping) mapping;
        saveToken = t_mapping.isSaveToken();
    }
    if (saveToken) {
        saveToken(request);
    }
    return forward;
}
```

3. 해당 UI에서 Form Submit시 이를 처리하기 위한 Action 클래스의 매핑 정보에 validateToken, resetToken 속성값을 true로 정의해 주도록 한다.

```
<set-property property="validateToken" value="true"/>
<set-property property="resetToken" value="true"/>
```

4. 해당 Action 클래스의 process 메소드 수행 전에, AbstractActionSupport의 preProcess 메소드에서는 Action 매핑시 정의한 validateToken, resetToken 값에 따라 Session에 있는 Token이 유효한지 체크 하고, 유효하다면 Session에 있는 Token을 지우게 된다. 이렇게 함으로써 중복 Form Submit만 발생하는 경우 Session에 있는 Token은 이미 지워졌으므로, Token 유효성 체크시 오류가 발생하게 되는 것이다.

```
public void preProcess(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response)
    throws Exception {

    boolean validateToken = false;
    boolean resetToken = false;

    if (mapping instanceof DefaultActionMapping) {
        DefaultActionMapping t_mapping = (DefaultActionMapping) mapping;
        validateToken = t_mapping.isValidToken();
        resetToken = t_mapping.isResetToken();
    }

    if (validateToken) {
        if (!isTokenValid(request, resetToken)) {
            throw new InvalidTokenException("common.msg.invalidtoken.error");
        }
    }
}
```

10.3.2.참고 사항

- <html:form> 사용

<html:form>를 사용하여 Form을 생성하는 경우, 별도 정의없이 Token 할당을 위한 Hidden 필드가 추가된다. 만일 <html:form>를 사용하지 않는 경우에는 다음과 같이 Hidden Field를 추가 정의해 주어야 한다.

```
<input type="hidden" name="org.apache.struts.taglib.html.TOKEN"
```

```
value="<%= session.getAttribute(org.apache.struts.Globals.TRANSACTION_TOKEN_KEY) %>">
```

- DefaultForwardAction 사용

별도 Action 수행없이 단순 페이지 이동만이 필요한 경우 Struts에서 기본으로 제공하는 ForwardAction 을 사용하게 된다. 그러나 이런 경우에도 입력 화면으로 이동하기 전에 Synchronized Token 처리를 위한 설정이 필요하다면, Anyframe 에서 제공하는 DefaultForwardAction을 사용토록 한다.

```
<action path="/synchronizedTokenView"  
  type="org.anyframe.struts.action.DefaultForwardAction"  
  parameter="/extensions/token.jsp">  
  <set-property property="saveToken" value="true" />  
</action>
```

11.Exception Handling

Anyframe에서 제공하는 BaseException 유형의 Exception이 throw되었을 때, 이를 처리하는 ExceptionHandler에 대해 알아보기로 하자.

- DefaultExceptionHandler

DefaultExceptionHandler는 Struts의 ExceptionHandler를 확장한 클래스로써, Anyframe에서 제공하는 BaseException이 catch 되었을때, BaseException 내에 정의된 메시지 key와 대체 문자열을 ActionMessage 객체에 저장하여 Forward하는 로직으로 구성되어 있다. 이 외, Exception이 catch 되었을 때는 Struts의 기본 ExceptionHandler에서와 같이 <exception>에 정의된 key 값을 이용하게 된다.

- DefaultBaseExceptionHandler

Anyframe에서 제공하는 BaseException을 상속받아 구현한 비즈니스 Exception은 생성 시점에 전달된 메시지 Key를 이용하여, Message Resource Bundle 내에 정의된 유형별 메시지(기본,해결책,원인)가 해당 Exception 객체에 담기게 된다. DefaultBaseExceptionHandler는 catch한 BaseException으로부터 유형별 메시지를 모두 처리할 수 있게 구성되어 있으므로 DefaultBaseExceptionHandler를 사용하는 것이 좋다. Tag Library를 통해 메시지 Key로써 여러 메시지를 출력하는 형태가 아닌 유형별 메시지 자체를 추출하고 있다. 또한, 비즈니스 Exception 외에 프리젠테이션 레이어에서 발생한 Exception에 대해서도 여러 메시지 처리가 가능토록 기본 Resource Bundle명인 "org.anyframe.struts.CommonResource"를 참조한다. 일반적으로, ExceptionHandler는 Anyframe에서 제공하는 DefaultBaseExceptionHandler를 상속받아 구현하면 된다.

다음에서는 선언적인 Exception Handling 기법과 DefaultBaseExceptionHandler 확장 방법에 대해 알아보기로 한다.

11.1.선언적인 Exception Handling

Anyframe 에서는 Action 클래스에서 직접 try-catch 문으로 Exception을 처리하지 않고, 속성 정의를 통해 선언적으로 exception을 처리할 수 있다.

11.1.1.Samples

다음은 struts-config-exception.xml 의 일부로 선언적인 exception 처리의 예이다.

```
<action path="/exceptionHandling"
  type="org.anyframe.struts.sample.action.ExceptionHandlingAction"
  scope="request">
  <exception type="org.anyframe.exception.BaseException"
    key="common.msg.action.error"
    path="/extensions/error.jsp"
    handler="org.anyframe.struts.action.DefaultBaseExceptionHandler"/>
</action>
```

위 설정을 통해 ExceptionHandlingAction 수행시 Exception이 발생한 경우, 발생한 Exception이 BaseException 유형이면 DefaultBaseExceptionHandler를 통해 해당 Exception이 처리된다. 아래와 같이 <global-exceptions> 내에 Exception에 대한 처리를 공통 정의할 수도 있다. 만일, <action>과 <global-exceptions> 내에 동일한 Exception이 정의되어 있는 경우 <action>에 정의된 Exception 처리가 우선 적용된다.

```
<global-exceptions>
  <exception path="/extensions/error.jsp"
    key="common.msg.authentication.error"
    type="org.anyframe.struts.util.AuthenticationException"
    handler="org.anyframe.struts.sample.common.SampleExceptionHandler" />
```

```

<exception path="/extensions/error.jsp"
  key="common.msg.authorization.error"
  type="org.anyframe.struts.util.AuthorizationException"
  handler="org.anyframe.struts.sample.common.SampleExceptionHandler" />
<exception path="/extensions/error.jsp"
  key="common.msg.base.error"
  type="org.anyframe.struts.util.BaseException"
  handler="org.anyframe.struts.sample.common.SampleExceptionHandler" />
</global-exceptions>

```

11.2.DefaultBaseExceptionHandler 확장

다음은 DefaultBaseExceptionHandler를 확장하여 해당 프로젝트의 Exception 처리 방식을 재정의한 SampleExceptionHandler.java 이다.

```

public class SampleExceptionHandler extends DefaultBaseExceptionHandler {

    public SampleExceptionHandler() {
        this.defaultBundle
            = "org.anyframe.sample.struts.web.common.SampleResources";
    }

    public ActionForward execute(Exception exception, ExceptionConfig config,
        ActionMapping mapping, ActionForm form, HttpServletRequest request,
        HttpServletResponse response) throws ServletException {

        ActionForward forward = mapping.getInputForward();

        if (exception instanceof AuthenticationException) {
            String loginPageURI = "/loginView.do";
            forward.setPath(loginPageURI);
            request.setAttribute("authenticateFail", "true");
        } else if (exception instanceof AuthorizationException) {
            String homePageURI = "/authorizationView.do";
            forward.setPath(homePageURI);
            request.setAttribute("authFail", "true");
        } else {
            String forwardPath = forward.getPath();
            if (forwardPath == null || forwardPath.equals("")) {
                forwardPath = "/loginView.do";
                request.setAttribute("authFail", "true");
            }
            String url = forwardPath + "?";

            Enumeration enumrequest = request.getParameterNames();
            while (enumrequest.hasMoreElements()) {
                String parameterName = (String) enumrequest.nextElement();
                String parameterValue = request.getParameter(parameterName);
                url += parameterName + "=" + parameterValue + "&";
            }
            forward.setPath(url);
        }
        request.getSession().setAttribute("afterErrorPage", forward);
        return super.execute(exception, config, mapping, form, request,
            response);
    }
}

```

예러 메시지 처리를 위한 기본 Message Resource Bundle을 org.anyframe.sample.struts.web.common.SampleResources로 재정의하였고 이전 요청 정보를 get

방식으로 url에 추가하여 afterErrorPage로 포워딩하는 로직이 추가되어 있다. 또한, Exception 유형에 따른 처리는 super.execute 메소드를 그대로 사용하고 있음을 알 수 있다. 다음은 해당 SampleExceptionHandler를 통해 전달된 에러 메시지를 처리하는 error.jsp의 일부이다.

```

<%
    ... 중략 ...
    String[] messages
        = (String[])request.getAttribute(Globals.MESSAGE_KEY);

    String userMessage = messages[0];
    String solution = "";
    String reason = "";

    if(messages.length==2) {
        solution = messages[1];
    }

    if(messages.length==3) {
        solution = messages[1];
        reason = messages[2];
    }
%>
    ... 중략 ...
    <%= userMessage %><p/>
    <% if(!solution.equals("")) { %>
        <strong>* SOLUTION</strong><br/>
        <%= solution %>
    <% } %>
    <% if(!reason.equals("")) { %>
        <strong>* REASON</strong><br/>
        <%= reason %>
    <% } %>
    ... 중략 ...
    <td background="<html:rewrite page='/sample/images/ct_btnbg02.jpg' />"
        class="ct_btn01" style="padding-top:3px;">
        <a href="javascript:fncGoAfterErrorPage();">확인</a>
    </td>
    ... 중략 ...

```

12. Authentication and Authorization

요청을 보낸 클라이언트가 등록된 사용자인지 체크하여 로그인하게 해주는 Authentication과 사용자와 어플리케이션 내의 자원 간의 관계 정보를 기반으로 접근 권한을 관리하는 Authorization은 어플리케이션 개발 시 항상 고려되어야 하는 부분 중의 하나이며 두 부분이 밀접히 관련되어 있다. 인증과 권한 기능을 어플리케이션에 추가하려면 몇 가지 고려해야 할 것이 있다.

- 사용자 정보는 어떤 방법(DB, LDAP, FILE, NT ...)을 이용해 관리할 것인지 ...
- 인증된 사용자 정보를 어떻게 (Session, Cookie...) 저장할 것인지 ...
- 사용자에게 권한을 부여하는 방법과 접근을 통제할 자원은 어떤 것인지 ...

JAAS 기반의 인증 방식을 적용하여 Login Context를 추상화함으로써 Login Module 개발 시 어플리케이션 코드의 수정 없이 타 시스템을 통한 인증을 수행할 수 있도록 처리하는 것이 가장 바람직하며, 프로젝트별 인증 처리 요건(ex. 외부 인증 솔루션 적용)에 맞게 작성해야 한다. 인증과 권한 관리 기능은 모든 어플리케이션에서 그대로 재사용할 수 있는 것은 아니다. 각 어플리케이션마다 정책이 다를 수 있으므로 커스터마이징이 필요하기 때문이다. 여기에서는 Anyframe 의 Struts 기반 개발시 인증과 권한 관리 방법에 대해 살펴보기로 한다.

12.1. Authentication

여기서는 일반적으로 많이 쓰이는 인증 방법인 DB에 저장된 사용자 정보를 기반으로, 인증을 수행하는 예에 대해 알아본다.

1. Anyframe 기반에서 인증 수행을 위한 서비스 개발. 이때, 해당 서비스에서는 인증된 사용자 정보를 담은 `javax.security.auth.Subject` 객체 전달
2. 로그인 수행을 위한 Action 코드에서 User ID, Password를 기반으로 해당 서비스를 호출하여 유효성 검증
3. 인증된 사용자 정보를 담고 있는 Subject 객체 내에 사용자가 속한 그룹(Role)의 정보를 `TypedPrincipal` 형태로 저장
4. Subject 객체를 'subject'라는 key값으로 Session에 저장
5. 이 후 Session에 저장된 Subject 객체를 이용하여 권한 관리 수행

12.1.1. Samples

다음은 DB 기반의 사용자 인증 처리를 수행하는 서비스 `DBAuthenticationServiceImpl.java` 의 일부분이다.

```
public class DBAuthenticationServiceImpl implements AuthenticationService {  
  
    // 종략 ...  
  
    public Subject authenticate(org.anyframe.sample.struts.services.UserVO userVO)  
        throws Exception {  
  
        Subject subject = null;  
        ResultSet rsu = null;  
        PreparedStatement pstmt = null;  
        Connection conn = null;  
  
        String userId = userVO.getUserId();  
        String password = userVO.getPassword();
```

```

try {
    conn = dataSource.getConnection();

    pstmt = conn.prepareStatement(sqlQuery);

    pstmt.setString(1, userId);
    pstmt.setString(2, password);

    // 입력된 사용자 정보를 기반으로 등록된 사용자 정보 검색
    rsu = pstmt.executeQuery();

    if (rsu.next()) {

        userId = rsu.getString(1);
        String userName = rsu.getString(2);
        rsu.getString(3);
        String grade = rsu.getString(5);

        Set principals = new HashSet();
        Set credentials = new HashSet();

        principals
            .add(new TypedPrincipal(userName, TypedPrincipal.USER));

        StringTokenizer tokens = new StringTokenizer(grade, ",");
        while (tokens.hasMoreTokens()) {
            principals.add(new TypedPrincipal(tokens.nextToken(),
                TypedPrincipal.GROUP));
        }

        // 사용자 정보를 Subject 객체에 저장
        subject = new Subject(false, principals, credentials,
            credentials);
    } else {
        throw new FailedLoginException();
    }
} catch (Exception e) {
    // 종락 ...
} finally {
    // 종락 ...
}
return subject;
}
}

```

다음은 앞서 구현한 DBAuthenticationServiceImpl 클래스의 속성 정의 파일인 context-authentication.xml의 일부이다. USER_ID와 PASSWORD 정보를 이용하여 사용자 정보와 그룹(Role) 정보를 조회하는 쿼리문을 속성값으로 정의하고 있음을 알 수 있다. 해당 서비스는 이 쿼리문을 이용하게 될 것이다.

```

<bean id="authenticationService"
    class="org.anyframe.sample.struts.services.impl.DBAuthenticationServiceImpl">
    <property name="dataSource" ref="dataSource" />
    <property name="sqlQuery"
        value="SELECT u.USER_ID,u.USER_NAME,u.PASSWORD,u.ENABLED,a.AUTHORITY
        FROM USERS u, AUTHORITIES a
        WHERE u.USER_ID=? and u.PASSWORD=? and a.USER_ID = u.USER_ID" />
</bean>

```

다음은 LoginAction.java 로, 앞서 정의한 서비스를 이용하여 사용자의 유효성을 체크하고, 유효한 사용자 정보를 Session에 저장하는 역할을 수행하고 있다.

```
public ActionForward process(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response)
    throws Exception {

    AuthenticationService authenticationService =
        (AuthenticationService) getService("authenticationService");

    UserForm userForm = (UserForm) form;
    UserVO userVO = new UserVO();
    BeanUtils.copyProperties(userVO, userForm);

    Subject subject = authenticationService.authenticate(userVO);

    HttpSession session = request.getSession();

    session.setAttribute("subject", subject);

    return (mapping.findForward("success"));
}
```

12.2.Authorization

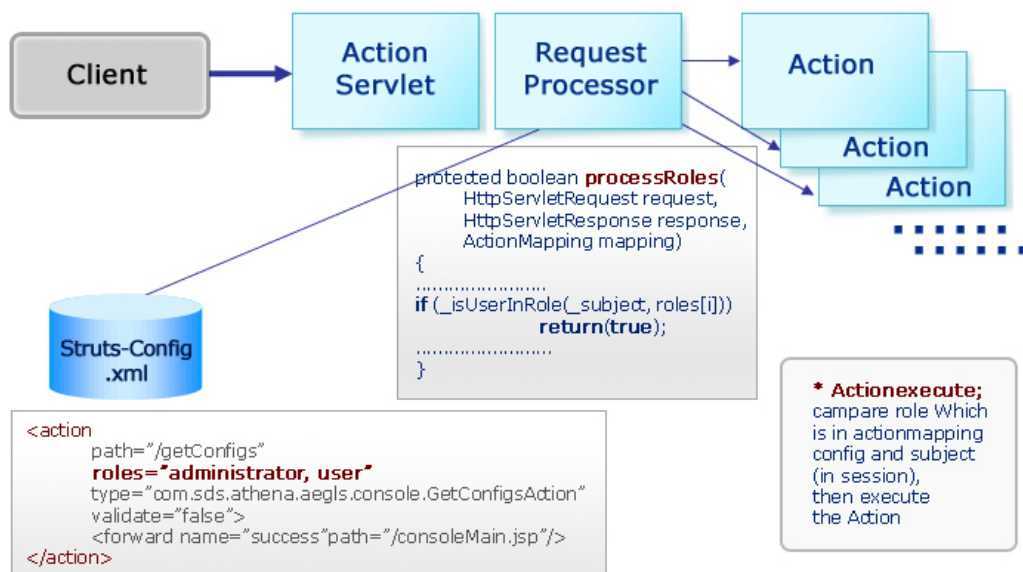
Action 단위로 접근 권한 제어가 가능하다.

12.2.1. 접근 권한 제어 프로세스

다음 그림에서 보여지는 사용자 정보를 기반으로 어떻게 특정 URL에 대한 접근 제어가 이루어지는지 살펴보고 하자.

USER_ID	USER_NAME	PASSWORD	GROUP
tester	관리자	test123	admin

1. Struts 속성 정의 파일 내에 Action 매핑시 roles 속성값 부여
2. Anyframe 에서 확장한 DefaultRequestProcessor의 processRoles 메소드에서 Session에 저장되어 있는 Subject 객체의 값과 roles 정보 비교 후 해당 Action 수행 여부 결정



12.2.2.Samples

다음은 특정 URL에 접근 가능한 사용자 그룹(Role)을 지정하고 있는 Action 매핑 정보이다.

```
<action path="/authorization"  
  type="org.anyframe.struts.action.DefaultForwardAction"  
  roles="admin"  
  scope="request"  
  parameter="/extensions/accessSuccess.jsp">  
</action>
```

위의 예에서 tester라는 USER_ID를 가진 사용자는 admin이란 GROUP에 속해 있으므로 authorization.do라는 요청을 수행할 수 있게 된다. roles 속성값이 부여되지 않은 Action의 경우에는 모든 사용자가 접근할 수 있음을 의미한다.

13.Spring Integration

Anyframe의 Struts 기반에서 웹 어플리케이션을 개발할 때 일반적으로 MVC의 Model 영역에 해당하는 비즈니스 객체는 Spring Framework 기반의 Bean 형태로 개발될 것이다. 따라서, 프리젠테이션 로직을 수행하는 Action 클래스에서 비즈니스 로직을 수행하는 Spring Framework 기반의 서비스에 접근하기 위한 방법에 대해서 살펴해보도록 하자.

13.1.Configuration

Spring Framework과 연계를 위해 다음과 같은 속성 정의가 필요하다.

- web.xml에 org.springframework.web.context.ContextLoaderListener를 listener로 등록
- web.xml에 context-param 요소로 contextConfigLocation 등록

13.1.1.ContextLoaderListener, ContextConfigLocation 정의

Servlet 2.3 이상에서는 웹 컨텍스트(하나의 웹 어플리케이션) 라이프 사이클 관련 이벤트인 ServletContextEvent와 Session의 라이프 사이클 관련 이벤트인 HttpSessionEvent가 추가되었다. 따라서 web.xml에 이러한 웹 어플리케이션의 이벤트에 응답하는 Context Event Listener를 등록해 줌으로써 해당 Listener 구현 클래스에서 웹 컨텍스트 초기나 종료 시점에 무언가 유용한 작업(ex. 어플리케이션의 초기 속성 로드, 서비스 컨테이너 기동 등..)을 수행할 수 있도록 해야 한다. org.springframework.web.context.ContextLoaderListener 클래스는 ServletContextListener 인터페이스를 구현하고 있으며, 어플리케이션이 Servlet 컨테이너에 의해 처음으로 로드되는 시점에 발생하는 startup 이벤트와 어플리케이션이 종료되는 시점에 발생하는 shutdown 이벤트를 처리할 수 있도록 다음의 두 메소드를 포함한다.

- contextInitialized : root WebApplicationContext를 초기화하고 contextConfigLocation에 정의된 Bean 속성 정의 XML 파일을 기반으로 관련된 서비스 인스턴스를 생성한다.
- contextDestroyed : 관련 자원을 release하고 root WebApplicationContext를 close 한다.

다음은 Spring Framework 연계를 위한 web.xml (ContextLoaderListener, contextConfigLocation) 의 일부이다.

```
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>
    /config/spring/context-*.xml
  </param-value>
</context-param>
<listener>
  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
```

13.2.Action

다음은 DefaultDispatchActionSupport를 상속받아 상품조회 기능에 대한 프리젠테이션 로직을 처리하는 ProductAction클래스의 일부이다. Super Class(AbstractActionSupport)에서 제공하는 getService 메소드를 호출하여 접근해야 할 비즈니스 서비스를 얻어내고 있음을 알 수 있다.

```
public class ProductAction extends DefaultDispatchActionSupport {
```

```
// 종료 ...
public ActionForward list(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response) throws Exception {
    ProductService productService =
        (ProductService) getService("productService");

    ProductSearchVO searchVO = new ProductSearchVO();

    ProductForm productForm = (ProductForm) form;
    BeanUtils.copyProperties(searchVO, productForm);

    // 종료 ...

    Page resultPage = productService.getPagingList(searchVO);

    request.setAttribute("search", searchVO);
    request.setAttribute("productList", resultPage.getList());
    request.setAttribute("size", resultPage.getTotalCount());
    request.setAttribute("pagesize", resultPage.getPagesize());
    request.setAttribute("pageunit", resultPage.getPageunit());

    return mapping.findForward("success_list");
}
}
```

Spring Framework 기반의 서비스 개발시 Bean 속성 정의에 관련된 자세한 사항은 Spring IOC [http://dev.anyframejava.org/docs/anyframe/plugin/essential/core/1.0.3/reference/htmlsingle/core.html#core_spring_ioc]를 참고한다.