JSP 2.3 & Servlet 3.1

커스텀 태그

- 김근형 -

커스텀태그(Custom tag)

- 커스텀 태그(Custom tag)
  - 표준 커스텀 태그인 JSTL에서 제공하지 않는 태그는 사용자가 직접 만들어 써야 한다.
  - 커스텀 태그는 JSTL에서 제공하지 않는 이외의 작업을 JSP의 스크립트를 사용하지 않고 태그 기반에서 간결하게 프로그래밍 하기 위해 꼭 필요하다.

커스텀태그(Custom tag)

- 커스텀 태그(Custom tag) 장점
  - 한 번 작성한 커스텀 태그는 언제든지 필요한 곳에서 재사용이 가능하다. 또한 다른 사용자에게 배포하여 재사용될 수 있다.
  - 가독성을 향상시킨다. 수백 라인의 페이지에서 가독성은 프로그램의 이해를 쉽게 해주는 중요 요소이다.
  - 커스텀 태그는 JSP의 스크립트를 사용하지 않으므로 자바 문법에 덜 의존적이다. 그래서 JSP페이지의 작성이 보다 쉽다

커스텀태그(Custom tag)

- 커스텀 태그(Custom tag) 장점
  - 한 번 작성한 커스텀 태그는 언제든지 필요한 곳에서 재사용이 가능하다. 또한 다른 사용자에게 배포하여 재사용될 수 있다.
  - 가독성을 향상시킨다. 수백 라인의 페이지에서 가독성은 프로그램의 이해를 쉽게 해주는 중요 요소이다.
  - 커스텀 태그는 JSP의 스크립트를 사용하지 않으므로 자바 문법에 덜 의존적이다. 그래서 JSP페이지의 작성이 보다 쉽다

Java 클래스 파일 기반의 커스텀 태그 작성

- 작성 방법

Java 클래스 파일 기반의 커스텀 태그 작성

- Java Class 작성
  - TagClass를 작성하기 위해서는 TagSupport를 상속 받아야 한다.
  - 16ch/customtag/WelcomeTag.java

```
public class WelcomeTag extends TagSupport{

    private static final long serialVersionUID = 1L;

    public int doStartTag() throws JspException{
        try{
            pageContext.getOut().print("Welcome to My Custom Tag");
        }catch (Exception e) {
            e.printStackTrace();
        }
        return SKIP_BODY;
    }
}
```

Java 클래스 파일 기반의 커스텀 태그 작성

- 태그 라이브러리 디스크립터 파일(.tld) 작성
  - WEB-INF/tlds/welcomeTag.tld

```
<taglib
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee web-
        jsptaglibrary_2_1.xsd"
    xmlns="http://java.sun.com/xml/ns/javaee"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     version="2.1">

  <tlib-version>1.0</tlib-version>
  <short-name>welcomeTag</short-name>

  <tag>
    <name>welcome</name>
    <tag-class>ch16.customtag.WelcomeTag</tag-class>
    <body-content>empty</body-content>
  </tag>
</taglib>
```

Java 클래스 파일 기반의 커스텀 태그 작성

- web.xml 작성
  - WEB-INF/web.xml
    - taglib-uri : JSP에서 해당 태그를 사용하기 위한 태그 라이브러리 식별자를 기술한다.
    - taglib-location : TLD 파일의 실제 경로를 기술한다.

```
<jsp-config>
  <taglib>
    <taglib-uri>/WEB-INF/tlds/welcomeTag.tld</taglib-uri>
    <taglib-location>/WEB-INF/tlds/welcomeTag.tld</taglib-location>
  </taglib>
</jsp-config>
```

Java 클래스 파일 기반의 커스텀 태그 작성

- welcomeTag.jsp 작성

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="tag" uri="/WEB-INF/tlds/welcomeTag.tld" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<h3>자바클래스 기반의 커스텀태그 작성 예제</h3>
<tag:welcome/>
</body>
</html>
```

Java 클래스 파일 기반의 커스텀 태그 작성

- javax.servlet.jsp.tagext 패키지
  - 클래스 파일 기반의 커스텀 태그를 작성하려면 javax.servlet.jsp.tagext 패키지에서 제공하는 인터페이스를 구현해서 한다.
  - 다음은 이 패키지에서 제공하는 인터페이스 리스트이다.

인터페이스	설명
BodyTag	태그의 내용 부분이 있는 커스텀 태그를 정의할 때 사용한다. JSP 1.2부터 제공
DynamicAttributes	이 인터페이스를 구현하는 동적 속성(dynamic attributes)를 받아들이는 태그를 정의하기 위해 제공한다. JSP2.0부터 제공
IterationTag	반복적인 작업을 처리하는 커스텀 태그를 정의할 때 사용한다. JSP 1.2부터 제공
JspIdConsumer	컴파일러에 의해 생성된 ID를 제공하는 태그 핸들러인 컨테이너를 감지한다. JSP 1.2부터 제공
JspTag	Tag와 SimpleTag를 위한 베이스 클래스(부모클래스)로 제공한다. JSP 2.0 부터 제공
SimpleTag	Tag, IterationTag를 하나로 묶어서 좀 더 쉽게 구현이 가능한 커스텀 태그 생성 시 사용 한다. JSP 2.0 부터 제공
Tag	단순한 커스텀 태그를 정의할 때 사용한다. JSP 1.2부터 제공
TryCatchTag	Tag, IterationTag, BodyTag를 위한 보조 인터페이스로, 리소스 관리를 위한 부분이 필요할 때 사용한다. JSP 1.2부터 제공

Java 클래스 파일 기반의 커스텀 태그 작성

- javax.servlet.jsp.tagext 패키지
  - javax.servlet.jsp.tagext 패키지에서 제공하는 인터페이스들 중 실질적으로 커스텀 태그를 정의하는 데 사용하는 것은 Tag, IterationTag, BodyTag, SimpleTag 인터페이스이다.
  - 이들 인터페이스는 커스텀 태그 작성 시 직접 사용하지 않고 이들의 하위 클래스를 상속받아 생성한다. 이들이 구현된 하위 클래스를 사용해 커스텀 태그를 정의하는 것이 더 간단하다.
    - Tag, IterationTag 인터페이스를 구현해 커스텀 태그를 정의하는 경우 : TagSupport클래스를 상속받아 정의한다.
    - BodyTag 인터페이스를 구현해 커스텀 태그를 정의하는 경우 : BodyTagSupport 클래스를 상속받아 정의한다.
    - SimpleTag 인터페이스를 구현해 커스텀 태그를 정의하는 경우 : SimpleTagSupport클래스를 상속받아 정의한다.

Java 클래스 파일 기반의 커스텀 태그 작성

- 커스텀 태그를 정의하고 처리하는 태그 핸들러 클래스
  - 태그 핸들러(Tag Handler)란 커스텀 태그를 직접적으로 담당하여 처리하고 결과물을 생성해 주는 클래스 파일을 말한다.
  - 이러한 태그 핸들러를 만든다는 것은 일반적인 자바 프로그래밍을 하는 것과 같으며 Tag, IteratorTag, BodyTag, SimpleTag 인터페이스를 통해 개발자가 작성해야 할 메소드가 미리 정의되어 있다.

Java 클래스 파일 기반의 커스텀 태그 작성

- Tag 인터페이스의 메서드는 아래와 같다(TagSupport, BodyTagSupport 클래스를 상속 받아 커스텀 태그 정의할 때 사용)

메소드	설명
Int doEndTag()	끝 태그를 만날 때 실행된다.
Int doStartTag()	시작 태그를 만날 때 실행된다
Tag getParent()	부모 태그를 구한다
Void release()	커스텀 태그를 사용하지 않을 때 실행된다
Void setPageContext(PageContext ctx)	커스텀 태그가 포함된 JSP 페이지의 컨텍스트를 전달받는다.
Void setParent(Tag t)	해당 태 ㅐ 그의 부모 태그가 존재할 때 부모 태그를 설정한다.

Java 클래스 파일 기반의 커스텀 태그 작성

- IterationTag 인터페이스는 Tag의 인터페이스를 상속받으므로 Tag의 모든 메서드 및 맴버변수를 사용할 수 있다.
- IterationTag 인터페이스의 메서드는 아래와 같다(TagSupport, BodyTagSupport 클래스를 상속받아 커스텀 태그 정의할 때 사용)

메소드	설명
Int doAfterBody()	태그의 body 내용을 처리한 뒤에 실행된다.

Java 클래스 파일 기반의 커스텀 태그 작성

- BodyTag 인터페이스는 Tag인터페이스와 IterationTag 인터페이스를 상속받으므로 Tag와 IterationTag의 모든 메서드 및 맴버변수를 사용할 수 있다.
- IterationTag 인터페이스의 메서드는 아래와 같다(BodyTagSupport 클래스를 상속받아 커스텀 태그 정의할 때 사용)

메소드	설명
Void dointBody()	Body를 수행하기 위한 준비를 한다
Void setBodyContent(BodyContent b)	Bodycontent 속성을 지정한다

Java 클래스 파일 기반의 커스텀 태그 작성

- 정의한 태그 내용의 처리 유무에 따라 TagSupport 클래스를 상속받을지 BodyTagSupport 클래스를 상속받을지를 결정해야 한다.
- 그러나 JSP2.0 이후에서는 SmipleTag 인터페이스에서 처리한다.
- SimpleTag 인터페이스의 메서드는 아래와 같다(SimpleTagSupport 클래스를 상속받아 커스텀 태그 정의할 때 사용)

메소드	설명
Void doTag()	커스텀 태그를 만나면 실행된다.
Protected JspFragment getJspBody()	이 메소드를 통해 JspFragment 객체를 얻는데, JspFragment 객체로 커스텀 태그의 body를 처리한다.
Protected JspContext getJspContext()	JspContext 객체를 얻어낸다.

Java 클래스 파일 기반의 커스텀 태그 작성

- 태그 핸들러 클래스 분류
  - 태그 핸들러 클래스는 태그 내용의 처리 여부에 따라 사용하는 것이 다르다

태그 핸들러 클래스	사용
TagSupport	태그의 내용<body>을 처리하지 않을경우
BodyTagSupport	태그의 내용을 처리하는 경우
SimpleTagSupport	태그의 내용이 있거나 없거나 모두 사용 가능

  - 내용이 있는 태그와 그렇지 않은 태그는 다음 예시와 같다.

태그 내용 처리에 따른 구분	예시
내용이 없는 태그	<tag:welcome>
내용이 있는 태그	<tag:welcome>홍길동</tag:welcome>

Java 클래스 파일 기반의 커스텀 태그 작성

- SimpleTagSupport 클래스를 사용한 내용없는 커스텀 태그 예제 작성.
  - ch16/customtag/SimpleWelcomeTag.java

```
public class SimpleWelcomeTag extends SimpleTagSupport{
    public void doTag() throws JspException {
        try {
            getJspContext().getOut()
            .print("SimpleTag: Welcome to My Custom Tag");
        }catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Java 클래스 파일 기반의 커스텀 태그 작성

- SimpleTagSupport 클래스를 사용한 내용 없는 커스텀 태그 예제 작성.
  - WEB-INF/tlds/simpleTag.tld

```
<?xml version="1.0" encoding="utf-8" ?>

<taglib
    xsi:schemaLocation=
    "http://java.sun.com/xml/ns/javaee web-
        jsptaglibrary_2_1.xsd"
    xmlns="http://java.sun.com/xml/ns/javaee"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     version="2.1">

  <tlib-version>1.0</tlib-version>
  <short-name>simpleTag</short-name>

  <tag>
    <name>simpleWelcome</name>
    <tag-class>ch16.customtag.SimpleWelcomeTag</tag-class>
    <body-content>empty</body-content>
  </tag>
```

Java 클래스 파일 기반의 커스텀 태그 작성

- SimpleTagSupport 클래스를 사용한 내용 없는 커스텀 태그 예제 작성.
  - WEB-INF/web.xml

```
<taglib>
  <taglib-uri>/WEB-INF/tlds/simpleTag.tld</taglib-uri>
  <taglib-location>/WEB-INF/tlds/simpleTag.tld</taglib-location>
</taglib>
```

Java 클래스 파일 기반 의 커스텀 태그 작성

- SimpleTagSupport 클래스를 사용한 내용 없는 커스텀 태그 예제 작성.
  - WebContent/ch16/simpleTag.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="simple" uri="/WEB-INF/tlds/simpleTag.tld" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<h3>SimpleTag를 사용한 커스텀 태그 작성- 내용없는태그</h3>
<simple:simpleWelcome/>
</body>
</html>
```

Java 클래스 파일 기반의 커스텀 태그 작성

- SimpleTagSupport 클래스를 사용한 내용 있는 커스텀 태그 예제 작성.
  - ch16/customtag/SimpleBodyTag.java

```
public class SimpleBodyTag extends SimpleTagSupport{
    public void doTag() throws JspException {
        try {
            JspFragment fragment = getJspBody();
            fragment.invoke(null);
        } catch (Exception e) {
            e.printStackTrace();;
        }
    }
}
```

Java 클래스 파일 기반의 커스텀 태그 작성

- SimpleTagSupport 클래스를 사용한 내용 있는 커스텀 태그 예제 작성.
  - ch16/customtag/SimpleTag.tld

```
<tag>
  <name>simpleBody</name>
  <tag-class>ch16.customtag.SimpleBodyTag</tag-class>
  <body-content>scriptless</body-content>
</tag>
```

Java 클래스 파일 기반의 커스텀 태그 작성

- SimpleTagSupport 클래스를 사용한 내용 있는 커스텀 태그 예제 작성.
  - WebContent/ch16/simpleBodyTag.jsp

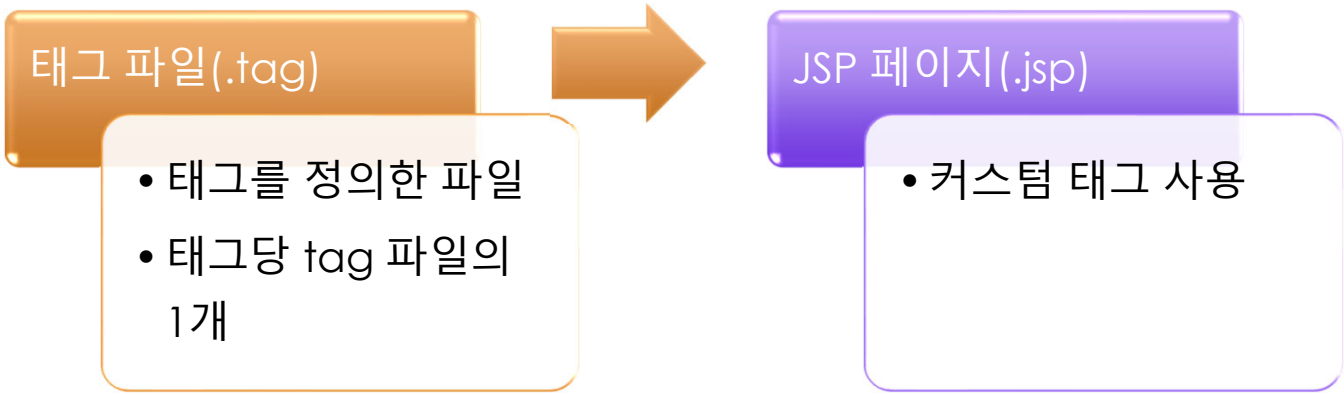
```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="simple" uri="/WEB-INF/tlds/simpleTag.tld" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<h3>SimpleTag를 사용한 커스텀 태그 작성- 내용있는태그</h3>
<simple:simpleBody>환영합니다. <b>Kingdora</b>님.</simple:simpleBody>
</body>
</html>
```

Java 클래스 파일 기반의 커스텀 태그 작성

- Java 클래스 파일 기반의 커스텀 태그 작성 단점
  - 소스코드가 공개되지는 않지만 작성 방법이 복잡하고 힘들다
  - 태그와 자바 소스와의 결합관계가 강해 재사용하기 힘들다.
  - JSP 2.0 이후 태그 파일 기반의 커스텀 태그 작성이 등장하면서 클래스 파일 기반은 서서히 빈도가 줄어듬

태그 파일 기반의 커스텀 태그 작성

- 태그 파일 기반의 커스텀 태그 작성 순서 및 필요 파일
  - 이 방법은 태그 파일(.tag)을 작성하며, 태그 라이브러리 사용하는 JSP 페이지가 필요하다.
  - 태그 파일 기반의 커스텀 태그를 작성해야 할 경우 작성순서는 아래와 같다.



태그 파일 기반의 커스텀 태그 작성

- 필요 파일 및 위치
  - 태그 파일 위치 : [웹 어플래케이션 폴더]-[WEB-INF]-[tags]
  - JSP 페이지 : [어플리케이션]-[WebContent]

태그 파일 기반의 커스텀 태그 작성

- 태그 파일 특징
  - 태그 파일 확장자는 .tag
  - <%@ tag %> 디렉티브를 사용한다.
  - 태그 파일에서만 쓸 수 있는 디렉티브를 사용해서 입력 출력 선언을 한다.
  - 태그 파일은 내부적으로 사용 될 때 태그 핸들러로 변환되어서 처리된다.
  - 태그 파일은 JSP처럼 내장 객체를 사용할 수 있다.

태그 파일 기반의 커스텀 태그 작성

- 태그 디렉티브 종류

디렉티브	설명
tag	JSP 페이지의 page 디렉티브와 동일, 태그 파일의 정보를 명시
taglib	태그 파일에서 사용할 태그 라이브러리를 명시할 때 사용,문법은 JSP 페이지와 완전히 동일
include	태그 파일에 특정한 파일을 포함시킬 때 사용, 태그 파일에 포함되는 파일은 태그 파일에 알맞은 문법으로 작성 해야 함
attribute	태그 파일이 커스텀 태그로 사용될 때 입력 받을 속성을 명시
variable	EL 변수로 사용될 변수에 대한 정보를 지정

태그 파일 기반의 커스텀 태그 작성

- tag 디렉티브
  - 태그 파일의 설정 정보를 기술하는데 사용
  - 이 디렉티브가 제공하는 속성은 아래와 같다.
    - display-name : 태그 파일을 도구에서 보여줄 때 사용될 이름을 지정, 기본값은 확장자 ".tag"를 제외한 태그 파일의 나머지 이름이다.
    - body-content : 몸체 내용의 종류를 입력, empty, tagdependent, scriptless 중 한가지를 사용, 기본값은 scriptless
    - dynamic-attributes : 동적 속성을 사용할 때, 속성의 <이름,값>을 저장하는 Map 객체를 page 범위에 속성에 저장할 때 사용할 이름을 명시, EL에서 변수이름으로 사용할 수 있다.(JSP 페이지의 pageContext 와 비슷하게 jspContext 기본 객체를 지원한다.)
    - description : 태그에 대한 설명
    - import : 사용할 자바 클래스를 지정
    - pageEncoding : 페이지 자체의 캐릭터 인코딩을 지정
    - isELIgnored : page 표현 언어를 지원할 여부를 지정
    - deferredSyntaxAllowedAsLiteral : #{ 문자가 문자열 값으로 사용되는 것을 허용 여부를 지정
    - trimDirectiveWhitespaces : 출력 결과에서 템플릿 텍스트의 공백 문자를 제거할지의 여부를 지정

태그 파일 기반의 커스텀 태그 작성

- 태그 파일을 이용한 커스텀 태그 작성 예제(내용 없는 태그)
  - WEB-INF/tags/message.tag

```
<%@ tag body-content="empty" pageEncoding="utf-8" %>

하하하... 메시지를 출력하는 태그
```

  - WebContent/ch16/messageTagUse.jsp

```
<%@ taglib prefix="tagFile" tagdir="/WEB-INF/tags" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<h3>태그 파일을 사용한 커스텀태그 작성 예제 - 내용 없는 태그</h3>
<b><tagFile:message/></b>
```

태그 파일 기반의 커스텀 태그 작성

- 태그 파일에서 사용 가능한 기본 객체
  - JSP 페이지와 마찬가지로 기본 객체를 사용해서 원하는 정보를 얻고 알맞은 작업을 수행할 수 있다
  - jspContext : pageContext 제공하는 setAttribute(), getAttribute() 메서드를 그대로 제공하며 각 속성과 관련된 작업을 처리
  - request, response, session, application, out : JSP 페이지의 기본 객체와 동일

태그 파일 기반의 커스텀 태그 작성

- 태그 파일을 사용한 커스텀 태그 작성 예제(내용 있는 태그)
  - WEB-INF/tags/tagBody.tag

```
<%@ tag body-content="scriptless" pageEncoding="utf-8" %>
<%@ tag import = "java.util.Date" %>

<% Date now = new Date();%>
오늘 날짜: <%=now%>
<hr>
<jsp:doBody/>
```

태그 파일 기반의 커스텀 태그 작성

- 태그 파일을 사용한 커스텀 태그 작성 예제(내용 있는 태그)
  - WEB-INF/tags/tagBodyUse.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="tagFile" tagdir="/WEB-INF/tags" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<h3>태그 파일을 사용한 커스텀태그 작성 예제  -  내용 있는 태그</h3>
<b><tagFile:tagBody>커스텀 태그에서 내용처리</tagFile:tagBody></b>
</body>
</html>
```

태그 파일 기반의 커스텀 태그 작성

- 태그 파일에 body가 있는 경우 처리
  - 태그 파일을 사용할 경우에도 몸체 내용을 출력, 데이터로 사용, 반복 출력 등을 할 수 있다.
  - 몸체 내용을 사용하는 방식은 EL 등을 처리 한 뒤 사용하는 방식, 데이터 자체로 사용하는 방식 2가지가 있다.
  - tag 디렉티브의 body-content 속성에 empty 대신에 'scriptless' 또는 'tagdependent' 값을 써야한다.
  - 몸체를 전달하는 방법은 몸체 내용을 직접 삽입하는 방법과 <jsp:body> 태그를 이용하는 방법이 있다.
  - 태그 파일로 구현된 태그의 몸체에는 일반 텍스트, EL, <jsp:include> 태그만 위치할 수 있다.
  - 몸체 내용에는 스크립트 요소(표현식, 스크립트릿)을 사용할 수 없다.

태그 파일 기반의 커스텀 태그 작성

- scriptless와 tagdependent 태그 사용
  - scriptless.tag

```
<%@ tag language="java" pageEncoding="UTF-8" body-content="scriptless"%>
<jsp:doBody />
```

  - tagdependent.tag

```
<%@ tag language="java" pageEncoding="UTF-8" body-content="tagdependent"%>
<jsp:doBody />
```

태그 파일 기반의 커스텀 태그 작성

- scriptless와 tagdependent 태그 사용
  - tagfileBodyContent.jsp

```
<c:set var="eltext" value="EL 텍스트" />
<table border="1" cellpadding="20">
    <tr>
        <td><tf:scriptless>
                <b>scriptless</b>
            </tf:scriptless></td>
        <td><tf:tagdependent>
                <b>tagdependent</b>
            </tf:tagdependent></td>
    </tr>
    <tr>
        <td><tf:scriptless>일반 텍스트</tf:scriptless></td>
        <td><tf:tagdependent>일반 텍스트</tf:tagdependent></td>
    </tr>
    <tr>
        <td><tf:scriptless>${ eltext }</tf:scriptless></td>
        <td><tf:tagdependent>${ eltext }</tf:tagdependent></td>
    </tr>
    <tr>
        <td><tf:scriptless>
                <jsp:body>jsp:body 태그를 이용</jsp:body>
            </tf:scriptless></td>
        <td><tf:tagdependent>
                <jsp:body>jsp:body 태그를 이용</jsp:body>
            </tf:tagdependent></td>
    </tr>
    <tr>
        <td>
            <%-- <tf:scriptless><%= out.println("스크립트") %></tf:scriptless> 에러 발생 --%>
        </td>
        <td><tf:tagdependent><%= out.println("스크립트") %></tf:tagdependent>
        </td>
    </tr>
</table>
```

태그 파일 기반의 커스텀 태그 작성

- Attribute 디렉티브(<%@ tag%>)
  - 태그 파일에서 커스텀 태그 속성을 선언하는데 사용되며 속성은 아래와 같다.

속성	설명
description	속성에 대한 설명
name	속성 이름, 태그 파일 내에서 스크립트 변수나 EL변수의 이름으로 사용, 각각의 attribute 디렉티브는 name 속성의 값이 달라야한다. tag 디렉티브(dynamic-attributes), variable 디렉티브(name-given) 속성과 값이 같으면 에러가 발생한다
required	속성의 필수 여부를 지정, 필수일 경우 true 아닌 경우 false를 값으로 지정, 기본값은 false
rtexprvalue	속성 값으로 표현식을 사용할 수 있는지의 여부를 지정, 기본값은 true
type	속성 값의 타입을 명시, int 등의 기본 데이터 타입은 명시할 수 없고 java.lang.Integer 와 같은 래퍼 타입을 사용, 기본값은 java.lang.String
fragment	<jsp:attribute> 액션 태그로 속성값을 전달할 때 true로 지정, true일 경우 rtexprvalue은 자동으로 false, type은 javax.servlet.jsp.tagext.JspFragment가 된다.

태그 파일 기반의 커스텀 태그 작성

- attribute 디렉티브를 사용한 예시
  - attributeEx.tag

```
<%@ tag body-content="scriptless" pageEncoding="utf-8" %>
<%@ attribute name="name" required="true" %>
<%@ attribute name="welcome" required="true" %>
<h2>${name}님, ${welcome}</h2>
```

태그 파일 기반의 커스텀 태그 작성

- attribute 디렉티브를 사용한 예시
  - attributeTagUse.jsp

```
<%@ page language="java" contentType="text/html;          t=UT
    pageEncoding="UTF-8"%>
<%@ taglib prefix="tagFile" tagdir="/WEB-INF/tags" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/cor
<%@ taglib  prefix="fn" uri="http://java.sun.com/jsp/jstl/f
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<h3>태그 파일을 사용한 커스텀태그 작성 예제  -  속성이 있는 태그</h3>
<c:set var="welcome" value="환영합니다." />
<form method="post" action="#">
  <input type="text" name="name">
  <input type="submit" value="Submit">
</form>

<c:if test="${fn:length(param.name) > 0}" >
  <tagFile:attribute name="${param.name}" welcome="${welcom
</c:if>
</body>
</html>
```

태그 파일 기반의 커스텀 태그 작성

- attribute 디렉티브를 사용한 예시2
  - removeHtml.tag

```
<%@ tag language="java" pageEncoding="UTF-8" body-content="scriptless"%>
<%@ attribute name="length" type="java.lang.Integer" %>
<%@ attribute name="trail" %>
<%@ attribute name="trim" %>
<jsp:doBody var="content" scope="page" />
<%
    //jspContext 기본 객체의 content 속성값을 읽음
    String content = (String)jspContext.getAttribute("content");
    if (trim != null && trim.equals("true")) {
        content = content.trim();
    }
    content = content.replaceAll(
        "<(/)?([a-zA-Z]*)(\\s[a-zA-Z]*=[^>]*)?>", "");

    if (length != null && length.intValue() > 0 &&
        content.length() > length.intValue()) {
        content = content.substring(0, length.intValue());
        if (trail != null) {
            content = content + trail;
        }
    }
%>
<%= content %>
```

태그 파일 기반의 커스텀 태그 작성

- attribute 디렉티브를 사용한 예시2
  - tagfileBodyCotentScriptless.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ page import="java.util.Date"%>
<%@ taglib prefix="tf" tagdir="/WEB-INF/tags"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <c:set var="dateEL" value="<%=new Date()%>" />
        <tf:removeHtml trim="true">
            <font size="10">현재<style>시간</style>은 ${ dateEL } 입니다.</font>
        </tf:removeHtml>
    <br>
    <br>
        <tf:removeHtml trim="true" length="5" trail=". . . 끝에 붙는 문자열">
            <u>현재 시간</u>은 <b>${ dateEL }</b> 입니다.
        </tf:removeHtml>
</body>
</html>
```

태그 파일 기반의 커스텀 태그 작성

- 변수 생성
  - 태그 파일을 사용하는 페이지에서 사용할 수 있는 EL 변수를 추가할 수 있다.
  - 태그 파일이 JSP 페이지로 데이터를 넘겨주기 위해서 변수를 사용한다.

태그 파일 기반의 커스텀 태그 작성

- variable 디렉티브 속성

속성	설명
description	속성에 대한 설명으로 기본값은 설명(description)을 기술하지 않는다(옵션)
Name-given 또는 name-from-attribute	태그 파일을 사용할 페이지에서 사용되어질 표현 언어(EL) 변수의 이름을 선언한다. name-given과 name-from-attribute 둘 중 하나는 반드시 기술해야 하며, 둘 다 기술하면 에러가 발생한다. 둘 이상의 variable 디렉티브를 가지고 있을 때 같은 name-given 속성값을 갖는 경우 에러가 발생한다. variable 디렉티브의 name-given 속성의 값, attribute 디렉티브의 name 속성의 값, tag directiv의 dynamic-attributes의 값이 같으면 에러가 발생한다(필수항목)
alias	name-from-attribute 속성을 사용하면 반드시 기술해 주어야 하는 속성으로 커스텀 태그에서 body에서 사용될 변수를 정의하는 것으로 이 변수는 같은 값을 가지는 표현 언어(EL) 변수의 이름을 명시한다(필수항목)
variable-class	변수의 타입을 명시하는 것으로 기본값은 java.lang.String(옵션)
declare	변수의 선언 유무를 나타내는 속성. 변수의 선언 유무와 관계없이 기본값은 True(옵션)
scope	변수의 범위, AT_BEGIN AT_END NESTED 중 한개 값을 가진다. 기본 값은 NESTED(옵션) - AT_BEGIN : 태그 파일의 시작 태그부터 태그파일에서 추가한 변수를 사용할 수 있다. - NESTED : 태그 파일의 시작태그와 끝 태그 사이에서 변수를 사용할 수 있다. - AT_END : 태그 파일의 끝 태그 이후부터 변수를 사용할 수 있다.

태그 파일 기반의 커스텀 태그 작성

- NESTED 예제
  - variable.tag

```
<%@ tag body-content="scriptless" pageEncoding="utf-8" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ variable name-given="x" scope="NESTED" %>

<c:set var="x" value="2"/>
<jsp:doBody/>
<c:set var="x" value="4"/>
```

태그 파일 기반의 커스텀 태그 작성

- NESTED 예제
  - variableTagUse.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="tagFile" tagdir="/WEB-INF/tags"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h3>태그 파일을 사용한 커스텀태그 작성 예제 - 속성이 있는 태그</h3>
    <c:set var="x" value="1" />
    <p>variable태그 전 : x = ${x} <%-- (x == 1) --%> </p>
    <tagFile:variable>
        <p>variable태그 안 : x = ${x} <%-- (x == 2) --%> </p>
    </tagFile:variable>
    <p>variable태그 밖 : x = ${x} <%-- (x == 1) --%> </p>
</body>
</html>
```

태그 파일 기반의 커스텀 태그 작성

- AT_BEGIN 예제
  - variable2.tag

```
<%@ tag body-content="scriptless" pageEncoding="utf-8" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ variable name-given="x" scope="AT_BEGIN" %>

<c:set var="x" value="2"/>
<jsp:doBody/>
<c:set var="x" value="4"/>
```

태그 파일 기반의 커스텀 태그 작성

- AT_BIGIN 예제
  - variable2TagUse.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="tagFile" tagdir="/WEB-INF/tags"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<h3>태그 파일을 사용한 커스텀태그 작성 예제 - 속성이 있는 태그</h3>
<c:set var="x" value="1"/>
<p>variable태그 전 : x = ${x} <%-- (x == 1) --%> </p>
<tagFile:variable2>
  <p>variable태그 안 : x =  ${x} <%-- (x == 2) --%> </p>
</tagFile:variable2>
<p>variable태그 밖 : x = ${x} <%-- (x == 4) --%> </p>
</body>
</html>
```

태그 파일 기반의 커스텀 태그 작성

- AT_END 예제
  - variable3.tag

```
<%@ tag body-content="scriptless" pageEncoding="utf-8" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ variable name-given="x" scope="AT_END" %>

<c:set var="x" value="2"/>
<jsp:doBody/>
<c:set var="x" value="4"/>
```

태그 파일 기반의 커스텀 태그 작성

- AT_END 예제
  - variable3TagUse.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="tagFile" tagdir="/WEB-INF/tags"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<h3>태그 파일을 사용한 커스텀태그 작성 예제 - 속성이 있는 태그</h3>
<c:set var="x" value="1"/>
<p>variable태그 전 : x = ${x} <%-- (x == 1) --%> </p>
<tagFile:variable3>
  <p>variable태그 안 : x =  ${x} <%-- (x == 1) --%> </p>
</tagFile:variable3>
<p>variable태그 밖 : x = ${x} <%-- (x == 4) --%> </p>
</body>
</html>
```

태그 파일 기반의 커스텀 태그 작성

- variable 디렉티브와 name-given을 이용해서 변수 추가
  - 형식 : <%@ variable name-given="EL변수명" variable-class="변수타입" scope="변수범위" />
  - name-given : 이 태그를 호출할 페이지에 추가될 변수명을 정의
  - variable-class : 추가될 변수의 타입, 기본 값은 java.lang.String

태그 파일 기반의 커스텀 태그 작성

- variable 디렉티브와 name-given을 이용해서 변수 추가
  - sum.tag

```
<%@ tag language="java" pageEncoding="UTF-8"
    trimDirectiveWhitespaces="true"%>
<%@ attribute name="begin" required="true" type="java.lang.Integer"%>
<%@ attribute name="end" required="true" type="java.lang.Integer"%>
<%@ variable name-given="sum" variable-class="java.lang.Integer"
    scope="NESTED"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<c:set var="sum" value="${ 0 }" />
반복전
<br>
<c:forEach var="num" begin="${ begin }" end="${ end }">
    반복
    <br>
    <c:set var="sum" value="${ sum + num }" />
</c:forEach>
반복끝
<br>
<jsp:doBody /><!-- 몸체 내용을 실행하기 전에 태그 파일에서 정의한 변수 sum을 태그를 호출한 페이지에 전달 -->
```

태그 파일 기반의 커스텀 태그 작성

- variable 디렉티브와 name-given을 이용해서 변수 추가
  - tagfileNameGiven.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="tf" tagdir="/WEB-INF/tags"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    태그전
    <br>
    <tf:sum end="10" begin="1"> 태그속<br> 1 ~ 10 까지 합 : ${ sum }<br>
    </tf:sum>
    태그끝
    <br>
</body>
</html>
```

태그 파일 기반의 커스텀 태그 작성

- variable 디렉티브와 name-from-attribute을 이용해서 변수 추가
  - name-given 속성을 사용하면 정해진 변수명만 사용할 수 있는데, 경우에 따라 원하는 이름을 갖는 변수를 추가할 수 있다.
  - 형식 : <%@ attribute name="속성 명" rtexprvalue="boolean" required="boolean" type="변수타입" %>
  - 형식 : <%@ variable alias="변수 명" name-from-attribute="사용할 속성 명" scope="변수범위" %>
  - alias : 태그 파일에서 변수의 값을 설정할 때 사용할 이름
  - name-from-attribute : 변수의 이름을 생성할 때 사용할 속성의 이름

태그 파일 기반의 커스텀 태그 작성

- variable 디렉티브와 name-from-attribute을 이용해서 변수 추가
  - max.tag

```
<%@ tag language="java" pageEncoding="UTF-8"
    trimDirectiveWhitespaces="true"%>
<%@ attribute name="var" required="true" rtexprvalue="false"%>
<%@ attribute name="num1" required="true" type="java.lang.Integer"%>
<%@ attribute name="num2" required="true" type="java.lang.Integer"%>
<%@ variable alias="maxNum" name-from-attribute="var"
    variable-class="java.lang.Integer" scope="AT_END"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
choose 전
<br>
<c:choose>
    <c:when test="${ num1 > num2 }">
        <c:set var="maxNum" value="${ num1 }" />
    </c:when>
    <c:otherwise>
        <c:set var="maxNum" value="${ num2 }" />
    </c:otherwise>
</c:choose>
choose 후
<br>
```

태그 파일 기반의 커스텀 태그 작성

- variable 디렉티브와 name-from-attribute을 이용해서 변수 추가
  - tagfileNameFromAttribute.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="tf" tagdir="/WEB-INF/tags"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    1번째 태그 전
    <br>
    <tf:max var="max1" num2="550" num1="300"> 1번째 태그<br></tf:max>
    1번째 태그 후
    <br> 최대값 : ${ max1 }

    <br> 2번째 태그 전
    <br>
    <tf:max var="max2" num1="550" num2="300" />
    2번째 태그 후
    <br> 최대값 : ${ max2 }
    <br>
</body>
</html>
```