escapeHtml And unescapeHtml

- description : escapeHtml And unescapeHtml
- author : 오션
- email : shlim@repia.com
- lastupdate : 2022-04-12 The

R-Utils > Crypto

escapeHtml

1. Encryption - ***escapeHtml*** 선택 - Input Text 입력란에 입력 후,

```
<p>MyName<p>
```

2. Encrypt Text 버튼을 클릭 시 Input Text 입력란에 입력되어 있던 상기 코드가 아래와 같이 변경되는 이유는

```
&lt;p&gt;MyName&lt;p&gt;
```

2.1 - 이때 Encryption - Input Text 입력란은 아래와 같이 코딩되어 있었고,

```
<input id="encryptInptText" class="encryptInptFld" name="encryptInptFld"
type="text"
 placeholder="Please input ID or Password" value="<c:out
value="${inputParam.encryptInptFld}" />"/>
```

2.2 - jsp에서 JSTL로 아래와 같이 처리하면,

```
<c:out value="값" />
```

2.3 - 상기 코드에서 **escapeXml="true"**로 default 설정되기 때문에 캐릭터 **<**, **>**, **&**, **'** , **"**들을 각각 캐릭터 엔티티 **<**, **>**, **&**, **'**, **"**로 변경하여 jsp에 표시되기 때문입니다.

2.4 - Ref Site
jstl 사용할 때 tag가 jsp 화면에 그대로 출력될 경우, escapeXml

JSTL core Tag out

2.5 - 아래와 같이 **escapeXml="false"**를 입력하면,

```
<input id="encryptInptText" class="encryptInptFld" name="encryptInptFld"
type="text"
 placeholder="Please input ID or Password" value="<c:out
value="${inputParam.encryptInptFld}" escapeXml="false" />"/>
```

2.6 - Encrypt Text 버튼을 클릭하여도 jsp의 Input Text 입력란은 아래와 같이 그대로 유지됩니다.

```
<p>MyName<p>
```

2.7 - 하지만 내부에서 Encrypted Result에 전달되는 값은 아래와 같으며,

```
&lt;p&gt;MyName&lt;p&gt;
```

2.8 - 상기의 값을 escapeHtml 처리를 하여 최종적으로 Encrypted Result 입력란에 표시되는 값은 아래와 같습니다.

```
&amp;lt;p&amp;gt;MyName&amp;lt;p&amp;gt;
```

2.9 - 정리 : 입력값, JSTL 처리값, escapeHtml 처리값은 아래와 같습니다.

2.9.1 - 입력값, jsp에 표시되는 값

```
<p>MyName<p>
```

2.9.2 - JSTL 처리값(내부에서만 존재)

```
&lt;p&gt;MyName&lt;p&gt;      // <P></p> 태그에서 "<" ⇒ "&lt;", ">" ⇒ "&gt;"로
변경되어 표시됩니다.
```

2.9.3 - escapeHtml 처리 완료 후 표시되는 값

```
&amp;lt;p&amp;gt;MyName&amp;lt;p&amp;gt;      // "&lt;" ⇒ "&amp;lt;", "&gt;"
⇒ "&amp;gt;"로 변경되어 표시됩니다.
```

unescapeHtml

3. Decryption - ***unescapeHtml*** 선택 - Input Text 입력란에 아래와 같이 escapeHtml 처리값을 입력 후,

```
&amp;lt;p&amp;gt;MyName&amp;lt;p&amp;gt;
```

3.1 - Decrypt Text 버튼 클릭 시 Input Text 입력란에 입력되어 있던 상기 코드가 아래와 같이 변합니다.

```
&amp;amp;lt;p&amp;amp;gt;MyName&amp;amp;lt;p&amp;amp;gt;
```

3.2 - 이때 Decryption - Input Text 입력란은 아래와 같이 코딩되어 있었습니다.

```
<input id="decryptInptText" class="decryptInptFld" name="decryptInptFld"
type="text"
placeholder="Please input encrypted ID or Password" value="<c:out
value="${inputParam.decryptInptFld}" />"/>
```

3.3 - 상기의 escapeHtml의 경우와 마찬가지로, 아래와 같이 **escapeXml="false"**를 입력하면,

```
<input id="decryptInptText" class="decryptInptFld" name="decryptInptFld"
type="text"
placeholder="Please input encrypted ID or Password" value="<c:out
value="${inputParam.decryptInptFld}" escapeXml="false"/>"/>
```

3.4 - Decrypt Text 버튼을 클릭하여도 jsp의 Input Text 입력란은 아래와 같이 그대로 유지됩니다.

```
&amp;lt;p&amp;gt;MyName&amp;lt;p&amp;gt;
```

3.5 - 하지만 내부에서 Decrypted Result에 전달되는 값은 아래와 같으며,

```
&amp;amp;lt;p&amp;amp;gt;MyName&amp;amp;lt;p&amp;amp;gt;
```

3.6 - 상기의 값을 내부에서 unescapeHtml 전처리 과정을 거쳐 최종적으로 Decrypted Result 입력란에 표시되는 값은 아래와 같습니다.

```
<p>MyName<p>
```

3.7 - 정리 : 입력값, JSTL 처리값, unescapeHtml 처리값은 아래와 같습니다.

3.7.1 - 입력값, jsp에 표시되는 값

```
&amp;lt;p&amp;gt;MyName&amp;lt;p&amp;gt;
```

3.7.2 - JSTL 처리값(내부에서만 존재)

```
&amp;amp;lt;p&amp;amp;gt;MyName&amp;amp;lt;p&amp;amp;gt;
```

3.7.3 - UnescapeHtml 처리 완료 후 표시되는 값

```
<p>MyName<p>
```

3.8 - unescapeHtml 전처리 과정 실수

3.9.1 - 지저분한 코드(4 lines) & 많은 변수

```
String unescapeHtml1stRslt =
resultData.setDecryptRsltFld(StringEscapeUtils.unescapeHtml(cryptoVO.getDecr
yptInptFld().replaceAll("&amp;", "&")));
String unescapeHtml2ndRslt =
(StringEscapeUtils.unescapeHtml(unescapeHtml1stRslt.replaceAll("&amp;",
"&")));
String unescapeHtml3ndRslt =
(StringEscapeUtils.unescapeHtml(unescapeHtml2ndRslt.replaceAll("&apos;",
"'")));
String unescapeHtml4thRslt =
(StringEscapeUtils.unescapeHtml(unescapeHtml3ndRslt.replaceAll("&lt;",
"<").replaceAll("&gt;", ">")));
```

3.9.2 - 지저분한 코드(4 lines)

```
String unescapedHtmlRslt =
resultData.setDecryptRsltFld(StringEscapeUtils.unescapeHtml(cryptoVO.getDecr
yptInptFld().replaceAll("&amp;", "&")));
unescapedHtmlRslt =
(StringEscapeUtils.unescapeHtml(unescapedHtmlRslt.replaceAll("&amp;",
"&")));
unescapedHtmlRslt =
(StringEscapeUtils.unescapeHtml(unescapedHtmlRslt.replaceAll("&apos;",
"'")));
unescapedHtmlRslt =
(StringEscapeUtils.unescapeHtml(unescapedHtmlRslt.replaceAll("&lt;",
"<").replaceAll("&gt;", ">")));
```

3.9.3 - 지저분한 코드(2 lines)

```
String unescapedHtmlRslt =
resultData.setDecryptRsltFld(StringEscapeUtils.unescapeHtml(cryptoVO.getDecr
yptInptFld().replaceAll("&amp;", "&")));
unescapedHtmlRslt =
(StringEscapeUtils.unescapeHtml(unescapedHtmlRslt.replaceAll("&amp;",
"&").replaceAll("&apos;", "'").replaceAll("&lt;", "<").replaceAll("&gt;",
">")));
```

3.9.4 - 최종(1 line)

```
String unescapedHtmlRslt =
resultData.setDecryptRsltFld(StringEscapeUtils.unescapeHtml(cryptoVO.getDecr
yptInptFld().replaceAll("&amp;", "&").replaceAll("&amp;",
"&").replaceAll("&apos;", "'").replaceAll("&lt;", "<").replaceAll("&gt;",
">")));
```

HtmlTagFilterRequestWrapper.class 확인 및 학습 필요

삽질 금지…

오션,, escapeHtml,, unescapeHtml