

## Extending Commons-BeanUtils

2009-09-10 19:03:23

In some cases it's useful to concat the values of BeanProperties when using the BeanUtils from Apache Commons. I just extended some few functionality to achieve this within the property-identifier itself.

With the help of a simple and weak parser one can use property descriptions like `propertyA+' '+propertyB`. For example with some Person bean one can write

```
ExtendedBeanUtil.getProperty(personBean, "lastname+', '+firstname");
```

The parser supports text within single quotes ' as well as + for concatenation. Also within a text can be used to escape values, like \ to escape a single quote (the first backslash is for the Java parser to escape the second backslash).

Of course such extended property descriptions can only be used for display issues, not for setting values of beans.

```
package de.keyboardsamurais.narcanti.commons.beans; import
org.apache.commons.beanutils.BeanUtils; /** * Supports concatenation and string literals within property
descriptions. * * @author M. Serhat Cinar */ public class ExtendedBeanUtil { public static class Token{
private boolean aProperty; private String value; public Token(boolean aProperty, String value) {
this.aProperty = aProperty; this.value = value; } public boolean isAProperty() { return
aProperty; } public void setAProperty(boolean property) { aProperty = property; } public
String getValue() { return value; } public void setValue(String value) { this.value = value; }
} public static class TokenParser{ private static enum State{Property, Concat, Text, Escape};
private ArrayList<Token> tokens = new ArrayList<Token>(); public TokenParser(String content){
State state = State.Property; final StringBuilder value = new StringBuilder(); for (final char
c:content.toCharArray()){ // System.out.println("&quot;c: &quot;&quot;"+c+"&quot;&quot; cur. state:
&quot;+state); switch (state){ case Property: // end of a property, change to
concatenation if (c=='+'){ // only if there is a property name at all if
(value.length()>0){ tokens.add(new Token(true, value.toString())); value.delete(0,
value.length()); } state = State.Concat; } // skip whitespaces within
property names else if (!Character.isWhitespace(c)){ value.append(c); }
break; case Concat: // skip double concatenations (++) and whitespaces if (c!='+' &&
!Character.isWhitespace(c)){ // next comes a text if (c=='"){ state = State.Text;
} // as this is not a text, it must be a property else{ value.append(c);
state = State.Property; } } break; case Text: // escape some
character if (c=='\'){ state = State.Escape; } // end of a string else if
(c==''){ // only if it's not empty if (value.length()>0){ tokens.add(new
Token(false, value.toString())); value.delete(0, value.length()); } state =
State.Property; } // append anything else (incl. whitespaces) else{
value.append(c); } break; case Escape: // no matter which character, just
append as it has been escaped value.append(c); state = State.Text; break; }
} // fini if (value.length()>0){ switch (state){ case Property: tokens.add(new
Token(true, value.toString())); break; case Text: tokens.add(new Token(false,
value.toString())); break; } } } public List<Token> getTokens(){ return tokens;
} } private ExtendedBeanUtil(){ // this method does the work of building a result string from the
property public static String getProperty(Object targetBean, String property){ final TokenParser parser
```

## Narcanti

Extending Commons-BeanUtils

```
= new TokenParser(property);    final StringBuilder result = new StringBuilder();    for (final Token  
token:parser.getTokens()){    if (!token.isAProperty()){    result.append(token.getValue());    }  
else{    result.append(BeansUtil.getProperty(targetBean, token.getValue()));    }    return  
result.toString(); } }
```