How to modify an attribute on a HTML tag

There are three different approaches to adding or updating a Components HTML attributes.

The first approach is the simplest approach which overrides the Components "onComponentTag" method. In your WebPage:

```
...

// this can be any Component
new TextField("my-text-field", myModel){

        @Override
        protected void onComponentTag(final ComponentTag tag){
                super.onComponentTag(tag);
                tag.put("onmouseover", "foo();return false;");
        }
};

...
```

You can also add to existing tag attributes using the onComponentTag:

```
Image img = new Image("my-img-id") {
        @Override
        protected void onComponentTag(ComponentTag tag) {
                // illustrates how to prevent an image from caching by adding a random value to the src
attribute
                // This is similar to the code thats in NonCachingImage and would be the preferable solution
for this
                super.onComponentTag(tag);
                String src = (String) tag.getAttributes().get("src");
                src = src + "&rand=" + Math.random();
                tag.getAttributes().put("src", src);
        }
};
```

A decoupled approach is to use a generic AbstractBehavior. In your WebPage:

```
...

new TextField("my-text-field", myModel).add(new AbstractBehavior(){

        @Override
        public void onComponentTag(Component component, ComponentTag tag) {
                tag.put("onmouseover", "foo();return false;");
        }
});

...
```

Another decoupled approach is to use an AttributeModifier/AttributeAppender.

To modify a tag attribute, one can use a AttributeModifier. Just add an instance of AttributeModifier to the tag's Java component. When creating the AttributeModifier class the name of the attribute you want to modified must be passed, along with a model containing the value to use. For example:

In your HTML you would have something like this:

```
<span wicket:id="alarm" class="unknown"/>Alarm Status</span>
```

And in your WebPage Java something like this:

```
Label alarmLabel = new Label("alarm", new PropertyModel(test,  "alarmState"));
 alarmLabel.add(new AttributeModifier("class", new Model() {
   public Object getObject(final Component component) {
         String cssClass;
         if ( test.getAlarmState() )
             cssClass = "alarm";
         } else {
             cssClass="noAlarm";
         }
         return cssClass;
   }
 }));
 add(alarmLabel);
```

If the attribute is not already present it will not, by default, be created. To quote the Javadoc: "If an attribute is not in the markup, this modifier will add an attribute to the tag only if addAttributeIfNotPresent is true and the replacement value is not null." (see constructor AttributeModifier for more details).

There is a SimpleAttributeModifier which always adds/replaces the attribute, but like the name suggests it is pretty simple and requires the attribute value upfront. If your attribute value is changing all the time you need a modifier that accepts a model, like AttributeModifier.

```
// adds or replaces the class attribute on a component with my-css-class
component.add(new SimpleAttributeModifier("class", "my-css-class"));
```

Another example would be to use the AttributeAppender:

```
...

new TextField("my-text-field", myModel).add(new AttributeAppender("onmouseover", new Model("foo();return
false;"), ";"));

...
```

Raw Markup

Use IMarkupFilter if you are using markup that is not attached to any component