Automatic styling of form errors

My team is developing a web application with many forms, and many validations of form elements. Managing the styling of input elements and their labels was a constant nuisance. The conventional tools Wicket makes available such as behaviors are only partial solutions since the application code still needs to decide which components to render in AJAX onError() and onSubmit() callbacks.

The following class handles these cross-cutting concerns by automatically

1. adding a CSS error class to FormComponents and their FormComponentLabels when they are invalid,
2. adding all invalid components to AjaxRequestTargets, and
3. adding previously invalid (but now valid) components to subsequent AJAX responses.

This class is developed against Wicket 1.4.17, and must be registered with Application#addPreComponentOnBeforeRenderListener() and AjaxRequestTarget#addListener() within Application#newAjaxRequestTarget().

Note that this class should be used in conjunction with a separate AjaxRequestTarget.IListener that adds your FeedbackPanel to AjaxRequestTargets, e.g. as described in chapter 7 of *Apache Wicket Cookbook*.

FormErrorDecorator.java

```
/**
 * Adds CSS classes to form inputs and their labels before they're rendered.
 */
public class FormErrorDecorator implements IComponentOnBeforeRenderListener, AjaxRequestTarget.IListener {
        private static final IBehavior COMPONENT_ERROR = new CssClassAppender("inputerror");
        private static final IBehavior LABEL_ERROR = new CssClassAppender("labelerror");

        private static final MetaDataKey<Serializable> RENDERED_WITH_ERROR = new MetaDataKey<Serializable>() {
};

        /**
         * Force all FormComponents and FormComponentLabels to output a markupId. If the associated form
component
         * is invalid, add a temporary behavior to append a CSS error class.
         */
        @Override
        public void onBeforeRender(Component component) {
                if (component instanceof FormComponent || component instanceof FormComponentLabel) {
                        component.setOutputMarkupId(true);

                        if (isInvalidFormComponent(component)) {
                                component.add(COMPONENT_ERROR);
                                component.setMetaData(RENDERED_WITH_ERROR, Boolean.TRUE);
                        }
                        else if (isInvalidFormComponentLabel(component)) {
                                component.add(LABEL_ERROR);
                                component.setMetaData(RENDERED_WITH_ERROR, Boolean.TRUE);
                        }
                }
        }

        /**
         * Before responding to AJAX requests, make sure that all form components and their labels that
         * need to be rendered get added to the AjaxRequestTarget.
         */
        @Override
        public void onBeforeRespond(Map<String, Component> map, AjaxRequestTarget target) {
                target.getPage().visitChildren(new AjaxRenderingVisitor(target));
        }

        @Override
        public void onAfterRespond(Map<String, Component> map, IJavascriptResponse response) {
        }

        private static boolean isInvalidFormComponentLabel(Component component) {
                if (component instanceof FormComponentLabel) {
                        LabeledWebMarkupContainer labeled = ((FormComponentLabel) component).getFormComponent();
                        return isInvalidFormComponent(labeled);
                }
                return false;
```

```
	}

	private static boolean isInvalidFormComponent(Component component) {
		if (component instanceof FormComponent) {
			FormComponent<?> formComponent = (FormComponent<?>) component;
			return !formComponent.isValid();
		}
		return false;
	}

	/**
	 * Adds components to an AjaxRequestTarget that are invalid, or that are valid after being invalid
	 * last request. This class does NOT add the error styling.
	 */
	private static class AjaxRenderingVisitor implements IVisitor<Component> {
		private final AjaxRequestTarget target;

		public AjaxRenderingVisitor(AjaxRequestTarget target) {
			this.target = target;
		}

		@Override
		public Object component(Component component) {
			if (isInvalidFormComponent(component) || isInvalidFormComponentLabel(component)) {
				target.addComponent(component);
			}
			else if (component.getMetaData(RENDERED_WITH_ERROR) != null) {
				component.setMetaData(RENDERED_WITH_ERROR, null);
				target.addComponent(component);
			}
			return IVisitor.CONTINUE_TRAVERSAL;
		}
	}

	/**
	 * Temporary behavior that appends a CSS class to the component tag.
	 */
	private static class CssClassAppender extends AbstractBehavior {
		private final String cssClass;

		public CssClassAppender(String cssClass) {
			this.cssClass = cssClass;
		}

		@Override
		public void onComponentTag(Component component, ComponentTag tag) {
			tag.append("class", cssClass, " ");
		}

		@Override
		public boolean isTemporary() {
			return true;
		}
	}
}
```