

# Custom Formatter Code Examples

These handy Java examples create different types of custom formatters.

## CustomFormatter.java

```
package com.hof.mi.interfaces;

/**
 * Base class for Custom Formatters
 */

public abstract class CustomFormatter extends UserInputParameters {

    private static final String cvsId = "$Id: CustomFormatter.java,v 1.1 2008-02-29 04:07:06 steve Exp $";
    public static final int RENDER_HTML = 1;
    public static final int RENDER_TEXT = 2;
    public static final int RENDER_CSV = 3;
    public static final int RENDER_LINK = 4;

    /**
     * Returns the display name for this custom formatter
     * @return name
     */
    public abstract String getName();

    /**
     * Returns true if this custom formatter accepts the given native type.
     * @param type the data type to check
     * @return true/false
     */
    public abstract boolean acceptsNativeType(int type);

    /**
     * Returns whether or not the value returned from this formatter
     * includes html code. If this returns false, a String value
     * will have any html entities parsed before output.
     * @return true/false
     */
    public boolean returnsHtml() {
        return false;
    }

    /**
     * Method to render a value using this custom defined data type format.
     * This is called internally when the report is rendered. The current render type is passed in so
     * that rendering can be modified for each type (ie. CSVs cannot contain image data)
     * @param value - Object to be formatted
     * @param renderType - How the report is being rendered.
     * @return String - The value rendered to a String
     * @throws Exception - Throw an exception, Yellowfin will do an Object.toString() call if an exception is
     encountered
     */
    public abstract String render(Object value, int renderType) throws Exception;
}
```

## CurrencyFormatter.java

In this example, a custom formatter accepts a numeric value and formats it as a currency value.

```

package com.example.formatters;
import com.hof.mi.interfaces.CustomFormatter;
import java.text.*;

public class CurrencyFormatter extends CustomFormatter {
    public String getName() {
        return "Currency Formatter";
    }
    public boolean acceptsNativeType(int type) {
        // We only handle numeric types
        if (type == TYPE_NUMERIC) return true;
        return false;
    }
    public String render(Object value, int renderType) throws Exception {
        if (value == null) return null;
        if (renderType == RENDER_LINK) {
            // Return a generic version of the value
            return value.toString();
        }

        // Create a String representing the value as a currency value
        NumberFormat nf = new DecimalFormat("0.00");
        return "$" + nf.format((Number)value); }
}

```

**Previous topic:** [Formatter packaging](#)

**Next topic:** [Appendix](#)