

com.hof.mi.interfaces.CustomFormatterclassclass

- `public abstract String getName();`
- `public abstract boolean acceptsNativeType(int type);`
- `public abstract String render(Object value, int renderType) throws Exception;`

- `public boolean returnsHtml();`
- `public boolean compareConditionalAgainstFormattedValue();`

public abstract String getName();

String

```
public String getName() {  
    return "My Formatter";  
}
```

public abstract boolean acceptsNativeType(int type);

truefalse

1

```
public boolean acceptsNativeType(int type) {  
  
    //accept text and numeric data  
    if (type == TYPE_TEXT  
        || type == TYPE_NUMERIC) {  
  
        return true;  
    } else {  
  
        //don't allow any other types  
        return false;  
  
    }  
}
```

2

```

public boolean acceptsNativeType(int type) {

    // we can handle any type
    return true;
}

```

public abstract String render(Object value, int renderType) throws Exception;

renderTypeYellowfin**renderType**PDF/XLS**RENDER_TEXT****renderType**

```

public String render(Object value, int renderType) throws Exception {
    if (value == null) return null
    if (renderType == RENDER_LINK)
        // Render the value for a drill-through link.
        // In this case you almost always want to return a generic
        // representation of the value.

        return value.toString();
    }

    // Return the formatted value
    return "Value: " + value.toString();
}

```

public boolean returnsHtml();

HTMLtruefalsefalse

public boolean compareConditionalAgainstFormattedValue();

truefalsefalse

HTML