Styling your Applications

For styling applications Servoy supports the use of HTML *cascading style sheets* CSS. In order to allow a more powerful way to interact with CSS syntax Servoy also implements the Less files compiler.

In This Chapter

CSS Overview

CSS is a plain text file that contains the style that is applied to every form of an application and to every component that can be shown inside it.

CSS file sample

```
.svy-field
{
        background-color:#f0f0f0;
        border-style: solid;
        border-width: 1px 1px 1px 1px;
        border-color: #cccccc;
        font-family: Verdana, sans-serif;
        font-size: 8pt;
        color: #333;
        text-align: left;
        margin: 2px;
        font-weight: normal;
}
.svy-form
{
        background-color: #ffffff;
        border-style: none;
}
.svy-label
{
        font-family: Verdana, sans-serif;
        font-weight: bold;
        font-size: 7pt;
        color: #666666;
    text-align: left;
}
```

Benefits of using CSS in Servoy include:

- Adds to the consistency of the user interface; form elements across the application can use the same styles.
- Gives the developer to make style changes over many/all forms by changing a single value in the CSS.
- Style sheets can be used over multiple applications, adding consistency to the all the company's applications.
- Adds consistency to an application in team development, making it easier for development teams to use the same styling on all forms.
- Style sheets can be changed programmatically, allowing a developer to change styles to different users' taste or to have periodic style changes in their applications. See overrideStyle function for more details.

Less Overview

Less (stands for Leaner Style Sheets) is a backwards-compatible language extension for CSS that that helps you write .css that is more dynamic. It supports variables and other helpful concepts. In Servoy Developer the .less file will compile into an actual .css automatically. For a detailed documentation about Less language see the Official Documentation for Less.

Because Less looks just like CSS, learning it is a breeze. Less only makes a few convenient additions to the CSS language, which is one of the reasons it can be learned so quickly and is the suggested approach for a smart, consistent and modern application styling.

Custom less/css sample

```
// import of the custom servoy theme properties that will import the hidden servoy theme, this imported file is
for customizing the default servoy theme properties
@import 'custom_servoy_theme_properties.less';

// Add your custom less/css to this file you can use the same less properties that are specified in the above
properties.less
.pitc-bkg-blue
{
   background-color: #95c0cb;
}
.pitc-txt-default
{
   font-weight: normal;
   font-size: 12pt;
   color:#000000;
   margin: 2px 2px 2px 2px;
   padding: 0px 0px 0px 0px;
}
.pitc-txt-default-med:extend(.pitc-txt-default)
{
   font-size: 12pt;
}
.pitc-txt-default-med-blue:extend(.pitc-txt-default-med)
{
   color:#0000ff;
}
```

Styling differences between Solution Types

Some difference exist regarding styling between solutions developed for NG client and solutions developed for smart or web client.

Styling for Smart or Web Client

- only CSS stylesheets are supported, Less styling language is not supported,
- CSS text files are stored in the Styles node of Solutions resources
- limited selection of available classes

Creating a Style for Smart or Web Client

To create a new Style for a Smart or Web Client

Right click on the **Styles** node of the **Resources** branch of the **Solution Explorer**

A window will appear from where a new Style can be created

As a stating point a default file will all the classes preloaded will be proposed.

Styling for NG Client

- unprocessed CSS (3.0) to do all styling of solutions.
- Less styling language is supported
- CSS and less text files are stored in the Media node of the active solution
- All possible CSS properties can be used, like:
    - All types of selectors
    - Pseudo-classes

For styling the NG client see this chapter: Styling in the NGClient

Creating a Style for NG

To create a new Style for a NG application:

Double click on the **StyleSheet** field in the **Properties** panel of the active solution

A window will appear from where a new Style (CSS or Less) can be created, or an existing one selected.

As a stating point a default file will all the classes preloaded will be proposed.

During the creation of a new Solution, if the desired solution is NG Client or NG Module type, the popup creation window will show a clickable checkbox that will automatically add .less files to the solution. If checked the solution will be created with the default ***custom_servoy_theme_properties.less*** file and a ***mysolutioName.less*** file in the solution media node. The ***custom_servoy_theme_properties.less*** file c*ontains the default servoy theme properties and can be used to modify them while the* **mysolutionName.less** file can be used to add new classes to the solution theme.

TIP

It is normally easier to work with an existing style. Most of the sample solutions have a style associated with them and these style are imported when the solution is imported into Servoy Developer. You can also create a new style and copy/paste different entries from one style to another.

Working With a Style

To use the styling a Style must be associated to the solution. To associate a style select the desired style on the **StyleSheet** field in the **Properties** panel of the active solution.

Structure of the Style Sheet

A style sheet for Servoy has basic style definitions and style definition classes.

The style definitions for Servoy are as follows:

- .svy-form
- .svy-flabel
- .svy-fbutton
- .svy-field
- ... and more

Under any of these style definitions, the developer can create many style definition classes. For example, the *label* style definition could have the style definition classes *label.title*, *label.small*, and *label.bold*.

Each definition and definition class can have one or many properties associated with it. Properties specified within the style definition are inherited (cascaded) to any style definition class under it. Study the example below

```
.svy-label {
        color: #ffffff;
        border-style: solid;
        font: bold 10pt Verdana;
}

.svy-label.mytext {
        color: red;
        vertical-align: middle;
        border-width: 1px 1px 1px 1px;
        border-color: #111111 #111111 #111111 #111111;
        margin: 2px 2px 2px 2px;
}
```

Notice that the border style and font are not modified in the *mytext* class. This means if a label were specified to use *mytext* for its style, it would be bold, 10pt, Verdana because that is what is specified in the parent style definition. The color would be red (not black) because that was overridden by the *mytext* definition class.

Setting Styles to Form Elements

Any elements on a form will take the styling from the style definition entry. For example, if using the style *test*, and there is a *label* entry in the *test* style sheet, then ANY label placed on the form will take the *label* style from the style sheet.

To use something other than the default style for any given element, change the styleClass entry in the Properties view for that element. To do this:

1. Select the desired element.

2. In the Properties view, find the property styleClass.
3. When clicking in the field, a drop down list will appear with all the available style classes for the element. This list is contextual for the element (meaning you will only see label style classes if the element is a label, field classes if the element is a field, and so on.)
4. Select the desired class from the list.
5. If you want to go back to the default style definition, select DEFAULT.

Working With a Style

For style a solution a Style must be associated to it. To associate a style select the desired style on the **StyleSheet** field in the **Properties** panel of the active solution.

1. Click on **Styles** under the Resources project in the Solution Explorer and do one of the following:
    a. Select the style desired (shown in the Solution Explorer list view) and click on the **Open style** button in the Solution Explorer list view toolbar.
    b. Double click the style desired.
    c. Right click on the style desired and select **Open style** from the menu.
2. The style will open in the center of the Workspace in an editor view.

Webinars

Servoy periodically produces Technical Webinars targeted to developers covering a broad range of topics, from new features to new capabilities to best practices. The following webinars are focused on the application styling: