NSColor

Category Name:	MiscExtensions
Declared In:	NSColor+MiscExtensions.h

Category Description

This category is mainly intended to simplify the storage of colors in the defaults database. For this task we need a NSPPL compliant object like an NSString. The general representation of a color is a string which contains multiple parameters separated by whitespaces (blanks). The first parameter is considered to be the name of the color space. The meaning of all subsequent parameters depends on the color space. Our format is almost identical with the output created by the **description** method of NSColor.

Since calibrated RGBA colors are the most common colors in non publishing Openstep applications they are an exception to the rule. Their string representation is allowed to have no color space parameter in the string. So if the color space is unknown it is considered to be a NSCalibratedRGBColorSpace. This exception makes it easier to manually create, adjust, modify or reuse already stored values from the defaults database.

NSDynamicSystemColorSpace colors can be archived but will we treated as regular RGB colors. However this is not really useful since colors from that space are not static but unarchiving them will create a static RGB color. Since this is a conceptual problem it is not clear if we can provide a fix.

The regular NSColor methods refer to CMYK as CMYB since they are more focused on the real color components. Our methods are more focused on the color spaces and therefore stick to CMYK for naming.

Note: We should come up with a more general and "bulletproof" archiving for other color spaces. Perhaps archiving a dictionary with a NSString type and NSData entry. This should allow for proper reconstructuion. Maybe the Pasteboard mechanims could be "misused" for this purpose sicne it then would even scale to yet unknown color spaces. The current solution does not really scale nicely since there is too much state hardcoded into this category. We would have to make categories for private classes (like NSCachedRGBColor) having a **stringRepresentation** method which knowns about the encoding... being nicer OO this has the problem of using undocumented classes.

Method Types

Creating colors from string representations

+ colorWithStringRepresentation:
+ colorWithRGBColorStringRepresentation:
+ colorWithHSBColorStringRepresentation:
+ colorWithCMYKColorStringRepresentation:
+ colorWithWhiteColorStringRepresentation:
+ colorWithNamedColorStringRepresentation:

Creating colors from string representations

- stringRepresentation
- rgbColorStringRepresentation
- hsbColorStringRepresentation
- cmykColorStringRepresentation
- whiteColorStringRepresentation
- namedColorStringRepresentation

Class Methods

colorWithCMYKColorStringRepresentation:
+ (NSColor *)**colorWithCMYKColorStringRepresentation:**(NSString *)*aString*

Returns the NSColor instance created from the valus stored inside *aString*. If the string did not contain a valid encoding of a NSDeviceCMYKColorSpace instance this method returns **nil**.

colorWithHSBColorStringRepresentation:
+ (NSColor *)**colorWithHSBColorStringRepresentation:**(NSString *)*aString*

Returns the NSColor instance created from the valus stored inside *aString*. If the string did not contain a valid encoding of a HSBEncoded NSDeviceRGBColorSpace or NSCalibratedRGBColorSpace instance this method returns **nil**. This method is automatically call from **colorWithRGBColorStringRepresentation** if a HSB encoded value is encountered.

colorWithNamedColorStringRepresentation:
+ (NSColor *)**colorWithNamedColorStringRepresentation:**(NSString *)*aString*

Returns the NSColor instance created from the valus stored inside *aString*. If the string did not contain a valid encoding NSNamedColorSpace instance this method returns **nil**.

Bug: Due to a hacky parameter decoding method this will fail if catalog or color names contain whitespaces .. yuck.

colorWithRGBColorStringRepresentation:

+ (NSColor *)**colorWithRGBColorStringRepresentation:**(NSString *)*aString*

Returns the NSColor instance created by **colorWithDeviceRed:green:blue:alpha:** based on the four values stored in the string. The values must be seperated by blank. Since RGB colors are the most common colors, they don't necessarily have to contain the colorspace information in the specified string parameter. If a HSB encoded color is found **colorWithHSBColorStringRepresentation** is called.

colorWithStringRepresentation:

+ (NSColor *)**colorWithStringRepresentation:**(NSString *)*aString*

Returns a color instance which is described by *aString*. The first parameter of *aString* is supposed to be the color space name. If no decoding method for the color space is known it is considered to be a calibrated RGB representation. On failure this method returns **nil**.

colorWithWhiteColorStringRepresentation:

+ (NSColor *)**colorWithWhiteColorStringRepresentation:**(NSString *)*aString*

Returns the NSColor instance created from the valus stored inside *aString*. If the string did not contain a valid encoding NSDeviceWhiteColorSpace or NSCalibratedWhiteColorSpace instance this method returns **nil**.

Instance Methods

cmykColorStringRepresentation

- (NSString *)**cmykColorStringRepresentation**

The created string contains the colorspace name and decimal number representations of the cyan, magenta, yellow, black and alpha components of the current color. If the color is not from the NSDeviceCMYKColorSpace this method will try converting it, by using **colorUsingColorSpaceName:**. Returns **nil** if the conversion was not possible.

hsbColorStringRepresentation

- (NSString *)**hsbColorStringRepresentation**

The created string contains the colorspace name, a special HSBEncoding note and decimal number representations of the hue, saturation, brightness and alpha components of the current color. If the color is not from the NSDeviceRGBColorSpace or NSCalibratedRGBColorSpace this method will try converting it, by using **colorUsingColorSpaceName:** into a color from the NSCalibratedRGBColorSpace. Returns **nil** if the conversion was not possible.

Since "HSB colors" are colors from the RGB colorspace they are encoded with a special "HSB" remark.

namedColorStringRepresentation
- (NSString *)**namedColorStringRepresentation**

The returned string contains the colorspace name and the strings for the colors catalog and color name which need to be properly quoted and escaped when necessary. If the color is not from the NSNamedColorSpace this method returns **nil**. No conversion is attempted since it is unclear which color catalog to use.

rgbColorStringRepresentation
- (NSString *)**rgbColorStringRepresentation**

The created string contains the colorspace name and decimal number representations of the red, green, blue and alpha components of the current color. If the color is not from the NSDeviceRGBColorSpace or NSCalibratedRGBColorSpace this method will try converting it, by using **colorUsingColorSpaceName:** into a color from the NSCalibratedRGBColorSpace. Returns **nil** if the conversion was not possible.

stringRepresentation
- (NSString *)**stringRepresentation**

Returns a string which represents the values of the color instance. The recommended format of the string contains the value returned by **colorSpaceName** followed by a number of additional values. All parameters must be separated by whitespace inside the generated string. Subclasses should implement a proper stringRepresentation method since the default behavior is to encode unknown colors using the NSCalibratedRGBColorSpace.

This method never generates a HSB representation of a color since HSB is just an alternative encoding of RGB colors and are not used by default.

Note: Since we can't implement this "smoothly" for Apples private color classes we have hardcoded string representations for all supported color spaces.

whiteColorStringRepresentation
- (NSString *)**whiteColorStringRepresentation**

The created string contains the colorspace name and decimal number representations of the white and alpha components. If the color is not from the NSDeviceWhiteColorSpace or NSCalibratedWhiteColorSpace this method will try converting it, by using **colorUsingColorSpaceName:**, into a color from the NSCalibratedWhiteColorSpace. Returns **nil** if the conversion was not possible.