Cookbook - SFG cheatsheet

Overview SaaS

This document outlines key point on working with SFG templates.

Functions

Functions are injected through **ctx** variable into template execution scope and are used to provide convenience methods for some of the encapsulated raw low level functions of the platform (e.g. access to HTTP request context, template inclusion mechanism, pre-processing parameters etc).

Version	Function	Param	Example
1.x.x	viewUri		`ctx.builder.a('href', ctx.viewUri, 'link')`
1.x.x	builder		`ctx.builder.html { body { div('Some text'); } }`
1.x.x	include	relativePath	`ctx.builder.section('class': 'blog') { div('class': 'container') { div('class': 'row') { ctx.include('component/' + (centralView ?: } } }`

1.x.x	includeurl	relativePath	`ctx.includeurl '/internal/breadcrumbs';`
			
			`ctx.builder.section('class': 'blog') {`
			`  div('class': 'container') {`
			`  div('class': 'row') {`
			`  ctx.includeurl "/internal/categorybody?cate`
			`  }`
			`  }`
			`}`

3.5.x	includevariant	relativePath, defaultPath	`ctx.includevariant(_optTemplate + '.groovy', 'SkuOption`
1.x.x	pagecache	timeout, keyType	`ctx.pagecache('1d', '-');`

1.x.x	inlineResource	relativePath	`div('class': 'col-xs-12 col-md-4') {`
			
			`  mkp.yieldUnescaped( ctx.inlineResource('StandardFoo`
			
			`}`

1.x.x	content	`contentId`	`div('class': 'form-group') {`
			
			`  mkp.yieldUnescaped(ctx.content('reset_resetform_con`
			
			`}`

1.x.x	contentExists	contentId	`if (ctx.contentExists(_optCms)) { // shop specific rende`
			
			`mkp.yieldUnescaped(ctx.dynamicContent(_optCms, [`
			`'product': product,`
			`'skus': _skus,`
			`'sku': _sku,`
			`'fc': _fc,`
			`'optionsModel': _optionsModel,`
			`'optionsModelItem': _option`
			`]));`
			
			`}`

1.x.x	dynamicContent	contentId,
		parameters
			`if (ctx.contentExists(_optCms)) { // shop specific rende`
			
			`mkp.yieldUnescaped(ctx.dynamicContent(_optCms, [`
			`'product': product,`
			`'skus': _skus,`
			`'sku': _sku,`
			`'fc': _fc,`
			`'optionsModel': _optionsModel,`
			`'optionsModelItem': _option`
			`]));`
			
			`}`

1.x.x	msg	messageKey	`span(ctx.msg('addToCompare'));`

1.x.x	msgf	messageKey,
		parameters
			`a('href': ctx.URL('logout'), 'rel': 'nofollow', 'title'`
			`['customer': sf?.managerName])) {`
			`span ctx.msg('logoffLabel');`
			`}`

1.x.x	contentURL	contentId	`a('href': ctx.contentURL('license'), 'rel': 'bookmark')`
			`span(ctx.msg('licenceInfo'));`
			`}`

1.x.x	categoryURL	categoryId	`a('rel': 'bookmark', 'href': ctx.categoryURL('notebooks`
			`img('src': ctx.resourceURL('image/baner1.png'));`
			`}`

1.x.x	productURL	productId, fc	`def _url = ctx.productURL(_item.product.uri, _item.supp`

1.x.x	skuURL	productId,
		fc
			`a('rel': 'bookmark', 'href': ctx.skuURL(_item.productSl`
			`span(ctx.macroProductName.name(_item.productSku));`
			`}`

1.x.x	resourceURL	path	`a('rel': 'bookmark', 'href': ctx.categoryURL('notebooks`
			`img('src': ctx.resourceURL('image/baner1.png'));`
			`}`

1.x.x	filteredURL	path	`def _selected = _multi &&`
			`ctx.macroFormUtils.values(_filteredNavBlock.code).conta`
			
			`def _path = _filteredNavBlock.code + '/' + ctx.encodeUR`
			`def _href;`
			`if (_selected) {`
			`_href = ctx.filteredURL('').replace(_path, '').repla`
			`} else {`
			`_href = ctx.filteredURL(_path);`
			`}`

1.x.x	URL	path	`a('href': ctx.URL('management/customers'), 'rel': 'nofo`
			`span ctx.msg('managedCustomers');`
			`}`

4.1.x	toAbsoluteURL	url,
		scheme
			`a('href': ctx.toAbsoluteURL(ctx.URL('management/custome`
			`span ctx.msg('managedCustomers');`
			`}`

1.x.x	encodeURI	uri	`input('type': 'hidden', 'name': 'returnTo', 'value': ctx.encodeURI('checkout?step=address'));`
1.x.x	decodeURI	uri	`a('href': ctx.URL(ctx.decodeURI(ctx.macroFormUtils.valu 'btn btn-default', 'title': ctx.msg('cancel')) { mkp.yieldUnescaped(ctx.msg('cancel')); }`

Variables

Variables are injected through **sf** variable into template execution scope and provide runtime context for current request (e.g. access to current shop, cart, filter navigation scope etc)

Example usage

```
def _total = sf.cartTotal;
```

Version	Variable	Description
1.x.x	deploymentMode	Deployment mode as defined by "webapp.configuration" runtime constant
4.1.x	serverName	Request server name
1.x.x	contextPath	Request context path
4.1.x	ssoContext	SSO context, injected by cart filter through SSO bridge
1.x.x	rawRequestURI	Raw request URI
1.x.x	rawRequestURL	Raw request URL
1.x.x	rawIncludeURI	Raw internal request URI
1.x.x	rawRequestCookies	Raw request cookies
1.x.x	rawRequestParameters	Raw request parameters
1.x.x	navRequestParameters	Navigation request parameters
1.x.x	skuIdAndUri	Sku ID and URI pair
1.x.x	productIdAndUri	Product ID and URI pair
1.x.x	categoryIdAndUri	Category ID and URI pair
1.x.x	contentIdAndUri	Content ID and URI pair
1.x.x	supplier	Fulfilment centre code
1.x.x	fc	Fulfilment centre code (synonym for supplier variable)
1.x.x	shopId	Current sales channel ID
1.x.x	shopCode	Current sales channel code
1.x.x	customerShopId	Current customer sales channel ID
1.x.x	customerShopCode	Current customer sales channel code

1.x.x	categoryId	Category ID
1.x.x (deprecated)	shopLogoURL	Shop IMAGE0
1.x.x	locale	Locale ISO-2 code
1.x.x	localeName	Locale full name
1.x.x	localeObject	Locale java object
1.x.x	supportedLocales	Supported locales ISO-2 code list
1.x.x	currency	Currency three ISO-3 code
1.x.x	currencyName	Currency name
1.x.x	supportedCurrencies	Supported currencies ISO-3 code list
1.x.x	currenciesConfig	Supported currencies ISO-3 code and name pairs list
1.x.x	loggedIn	Flag to indicate customer is logged in
1.x.x	requireCustomerLogin	Flag to indicate if customer login is required
1.x.x	customerName	Logged in customer name
1.x.x	customerType	Logged in customer type
1.x.x	customerTags	Logged in customer tags
1.x.x	managerName	Logged in manager name
1.x.x	guestCheckoutEnabled	Flag to indicate guest checkout is enabled (shop specific config "SHOP_CHECKOUT_ENABLE_GUEST")
1.x.x	checkoutAvailable	Flag to indicate that checkout is available for current customer
1.x.x	rfqAvailable	Flag to indicate that request for quote is available for current customer
1.x.x	orderRequireApprove	Flag to indicate order requires approval
1.x.x	meta	Page meta data (available in /internal/pageheadmetaseodata include only)
1.x.x	cart	Shopping cart object
1.x.x	cartTotal	Cart total
1.x.x	useManufacturerSku	Flag to indicate to use manufacturer SKU instead of product SKU to render (shop specific config "PRODUCT_DISPLAY_MANUFACTURER_CODE_SHOP")
1.x.x	useQuantityPicker	Flag to indicate to use quantity picker (shop specific config "CART_ADD_ENABLE_QTY_PICKER")
1.x.x	useProductOverride	Attribute to use for product name override (shop specific config "SHOP_PRODUCT_NAME_OVERRIDE")
1.x.x	addressbookEditable	Flag indicating that address book is editable by customer
1.x.x	addressbookBillingEditable	Flag indicating that address book is editable by customer
1.x.x	earliestNewProdInstant	Time indicating earliest time to be considered as "new product"
1.x.x	searchGlobalOnly	Flag to indicate that "search global only" mode is enabled
1.x.x	searchDisableCompound	Flag to indicate that "compound search" mode should be disabled
1.x.x	searchDisableSuggest	Flag to indicate that search suggest feature is disabled
1.x.x	searchSuggestMinChars	Search suggest minimum characters trigger
1.x.x	searchSuggestMaxItems	Search suggest maximum number of results to display
1.x.x	searchSuggestFadeOut	Search suggest "fade out" timeout
1.x.x	shopNewsletterEnable	Enable newsletter feature

1.x.x	categoryDisablePagination	Disable pagination on category pages
1.x.x	categoryDisableSorting	Disable sorting on category pages
1.x.x	categoryDisablePageSize	Disable page size selector on category pages
1.x.x	categoryDisableShowMore	Disable "show more" button on category pages