GlobalStore™
XSL UX Developer Guide

version 2.2 (01/03/2022)

Contents

1. Introduction

GlobalStore™ is an webshop application with shopping cart and basic order processing including online credit card payments via iPayServlet (at processor), bank link to internet banking (u-net, hansa-net etc), and others on request. The future releases may contain additional functionality like payments via other gateways or links.

2. Main features

- Shopping cart/order processing
- Product search, group browsing, new products, offers
- Automatic email to customer support (order notification)
- On-line credit card payments (via iPayServlet interface protocol version 0,1,2)
- Customer/account registration
- Display prices to user in multiple currencies (depending language/region preferences)
- Localized prices for locales/languages
- Customer order history
- Discounts: promotion coupons, fixed customer discounts
- Shipping: calculation of shipping based weight and country, multiple shipping services possible
- Sales/VAT tax calculation, multiple item rates, exports, EU residents and VAT regitered
- Customs tax calculation for export orders (if order is above customs floor limit)
- Multi language support (English, Estonian, others may be configured and translated through administration interface
- Item quantity management, reservation on order.
- Image database/management/upload
- Full administration: order tracking/management, user management , product management,
  taxes, languages/locales, settings, locales, shipping, coupons, currencies and rates
- Additional features on request
- Supports custom designed themes by user design or description, using customized XSL stylesheet and CSS styles

3. XSL Templates in relation to GlobalStore functionality

Whole XSL rendring system is combined with three XSL files:

main.xsl – main xsl template to rendering user views
admin.xsl – xsl template to rendere adminitrative only views
base.xsl – contains common finctionalities and subtemplates shared by both above

Both main and admin xsl imports base.xsl to use shared subtemples as necessary.

```
<xsl:import href="base.xsl"/>
```

if there is custom base xsl name it with prefix like in main

```
<xsl:import href="theme1.base.xsl"/>
```

Note if actively developing on server if making changes to base.xsl to reflect them in main.xsl touch also main.xsl that both get reloaded by server.

Typically no changes needed on admin.xsl as administrative layout and functionality is only for sop administrator and rarely needs changes. So in following we'll focus on user layouts and main.xsl.

For custom themes or merchant custom telplates template names can have theme name prefix eg: theme1.main.xsl or similar. Theme is then configured in shop settings to be used.

In main.xsl there root template

```
<xsl:template match="/">
      <xsl:call-template name="page"/>
</xsl:template>
```

calls template page in base.xsl that then implements general html layout headers, trailers etc so Your generic layout will be implemented in page template in base.xsl.
Inside page is called sub-template for content section depending on what content needs to be displayed this is implemented in main.xsl template selectView. For users following content pages can be shown depending on URLs its opens:

```
<xsl:template name="selectView">
<xsl:choose>
 <xsl:when test="Page/actionObject/viewName='shelfView'">
      <xsl:call-template name="shelfView"/>
 </xsl:when>
 <xsl:when test="Page/actionObject/viewName='shelfItem'">
      <xsl:call-template name="itemView"/>
 </xsl:when>
  <xsl:when test="Page/actionObject/viewName='cartView'">
      <xsl:call-template name="cartView"/>
 </xsl:when>
 <xsl:when test="Page/actionObject/viewName='orderPay'">
      <xsl:call-template name="orderPay"/>
 </xsl:when>
 <xsl:when test="Page/actionObject/viewName='orderCheckout'">
      <xsl:call-template name="orderCheckout"/>
 </xsl:when>
 <xsl:when test="Page/actionObject/viewName='orderView'">
      <xsl:call-template name="orderView"/>
 </xsl:when>
 <xsl:when test="Page/actionObject/viewName='loginView'">
      <xsl:call-template name="loginView"/>
 </xsl:when>
 <xsl:when test="Page/actionObject/viewName='registerView'">
      <xsl:call-template name="registerView"/>
```

```
 </xsl:when>
 <xsl:when test="Page/actionObject/viewName='orderListView'">
      <xsl:call-template name="orderListView"/>
 </xsl:when>
 <xsl:when test="Page/actionObject/viewName='error'">
            <xsl:call-template name="error"/>
 </xsl:when>
 <xsl:when test="Page/actionObject/viewName='passwordView'">
      <xsl:call-template name="passwordView"/>
 </xsl:when>
 <xsl:when test="Page/actionObject/viewName='resetPassView'">
      <xsl:call-template name="resetPassView"/>
 </xsl:when>
  <xsl:when test="Page/actionObject/viewName='main'">
     <div class="head">
     <xsl:value-of select="Page/actionObject/message" disable-output-
escaping="yes"/>
     </div>
 </xsl:when>
  <xsl:when test="Page/actionObject/viewName='static'">
     <xsl:call-template name="staticcontent"/>
  </xsl:when>
 <xsl:otherwise>
 </xsl:otherwise>
</xsl:choose>
</xsl:template>
```

About sub-templates view name in xml template name of xsl:

main – welcome page, may have custom content

shelfView – implements shelf/listing of products by category selected or search results and pagination also may sutpport directly adding items to card with javascript etc.’

itemView – implements individual product view and supports adding this item to cart

cartView – implements displaying items in cart, modify quantities etc

orderPay – implements redirection form to start payment with bank or other eternal system, there most likely no changes needed if any then cosmetic.

orderCheckout – finalizing order and proceeding to payment or checkout. On this view order items are displayed and additionally user can change transport options and finally select payment method to complete order online or offline.

orderListView – user sees its made ordersin list and open order and pagination

orderView – user sees its made order, static content no actions

loginView – user login with password

registerView – intial user registration or update existing user data.

error – unexpected system error, static content

passwordView – user can change password

resetPassView – user can reset forgotten password using email

static → staticcontent – displays static html content from database as is

So depending on what viewname is specified by shop core in xml xpath ***/Page/actionObject/viewName*** then XSL renders that set of views as desired.

3.1 Details of templates views

To full details of data available and forms and other content use explore default templates bundled with product and sample xml of each view for available dynamic data. Following is just logical guide to the most important features and data.

page – renders contents of most pages like menus and other useful information

menuItems – renders what menus have been set up

```
<xsl:for-each select="Page/menu/*"> ....
```

dynamic metatags if any

```
<xsl:value-of select="Page/metatags" disable-output-escaping = "yes"/>
```

dynamic header if any

```
<xsl:value-of select="Page/header" disable-output-escaping = "yes"/>
```

a default product search from

```
<form name="searchForm" action="shelf.do" method="POST">...
```

To access page specifix dynamic localized teexts see example templates, general xpath to render labels is

```
Page/labels/label[key='the text key']/label
```

example

```
<xsl:value-of select="Page/labels/label[key='order.discount']/label"/>
```

typically each view also have a localized head message that xpath is

```
<xsl:value-of select="Page/actionObject/message"/>
```

if a form is submitted and errors detected by server then
resulting errors are in xpath

```
Page/actionObject/errors/Error
```

etc

Inside page called templates:

main – just displays dynamic welcome message rest of contents static if desired.

```
<xsl:value-of select="Page/actionObject/message" disable-output-escaping="yes"/>
```

if message contains xml then `disable-output-escaping="yes"`
if just text content then enable escaping

shelfView – renders list of products.

Iteration of items with for-each

```
<xsl:for-each select="Page/actionObject/items//ShelfListItem">
```

itemView renders individual product data. Product data object xpath is

```
/Page/actionObject/item
```

cartView renders what items in current cart modifieable

```
<xsl:for-each select="Page/actionObject/items//CartListItem">
```

orderCheckout - renders what items in current cart fixed and shipping options and payment method selection and optionally a promo code entry for additional discount

Dynamic shippings xpath for for-each

```
<xsl:for-each select="/Page/actionObject/shippingItems//ShippingItem">
```

Dynamic payment methods xpath for for-each

```
<xsl:for-each select="/Page/actionObject/payMethodList//MenuItem">
```

orderListView *user sees list of its order history*

Order objects xpath is

```
<xsl:for-each select="Page/actionObject/orderList//OrderListItem">
```

orderView *– displays contents of individual order*

Order object xpath is

/Page/actionObject

To render items

```
<xsl:for-each select="Page/actionObject/items//CartListItem">
```

Main sub objects – customer, shipping

/Page/actionObject/customer

/Page/actionObject/customer

For more see xml.

registerView *– user data entry user data object xpath is*

Page/actionObject