Table of Contents

Väljatrükkide häälestamine 3
Häälestamine 3
Parameetrid XSLis 3
XSLi parameetri kasutuse näited 4
Ribakood pildina 4
Code128 ribakoodi kasutamine skriptiga 7
Scripti kasutamine XSLis 7
Ribakoodi funktsioon 8
Funktsiooni väljakutse 8
Näidis 8
Lingi kasutamine 11
Signeerimise tugi 11
EveryPay Linkpay 12
Alustamine 13
Seadistamine 13
HMAC signatuuri loomine välisele lingile 13
Näide 14
Nipid 14
0 (mitte NaN), kui pole andmeid 14
Mingi bloki X korda joonistamine 14
Node-set XML muutuja, sealt unikaalsete kirjete kuvamine 15
dateAdd - kuupäevale mingi päevade arvu lisamine 16
Splitter - Mingi tekstilise sisu laiali jagamine separaatori alusel 16
dateDiff - kuupäevade vahemiku päevade arvu arvutamine 17
Textarea sisu (rea vahedega, näiteks sündmuse kirjeldus) kuvamine ka väljatrükis reavahedega 17
Page break 18
Summeerimine 18
Palgateatisele puhkusejäägi kuvamine 19

Väljatrükkide häälestamine

Directo väljatrükk on oma olemuselt HTML kujundus, mille kirjeldamise toimub XSL Stylesheet-i abil. Väljatrükkides võtakse andmed XML-ist (igale dokumendile ja aruandele on oma XML kogum).

Häälestamine

- `ÜLDINE > Väljatrükkide häälestamine`
  või
- `ÜLDINE > Üldised seadistused > Väljtrükkide häälestamine`

Parameetrid XSLis

Väljatrükis saab kasutada järgnevaid võtmesõnu, kui on vaja muuta vaikimisi seadistusi:

- <!--mailfrom:saatja_mailiaadress -->
- <!--mailname:saatja_nimi-->
- <!--mailto:kellele saadetakse -->
- <!--mailnopdf:1 -->pdfi asemelt tehakse html
- <!--mailnoattachment: -->pdfi manust ei lisata üldse, ainult enda kirjutatud tekst
- PDF parameetrid
  Väljund landscape:
  <!--
  pdfparams:BrowserWidth=1000,isLandscape=1,version=1,HeaderHeight=60,FooterHeight=60-
  -> **debug_pdf=1** - näha visuaalselt päise ja jaluse tsoon
  Engine=1 - Gecko renderer (IE asemel)
  FontEmbed=1 - lisab PDFi fondid kaasa (lisada saab serverile installitud fontide seast)
  InsetX=10,InsetY=10 - X ja Y koordinaadid joonistamise alguseks
  BrowserWidth=800 võimaldab määrata nö Zoom astet, ehk kui suur vaade välja näeb (vaikimisi 800).
- <!--output-excel -->lisab exceli headeri - fail avatakse exceliga
- <!--clear-output -->väljundiks on ainult puhas XSL-i transformatsioon (ilma Directo päiseta)
- Meili saates luuakse sündmus, soovikorral saab muuta loodud sündmuse parameetreid vastavate XSL parameetritega <!--ev_type:xxx --> <!--ev_status:xxx --> <!--ev_status:xxx --> <!--ev_object:xxx --> <!--ev_start:xxx --> <!--ev_end:xxx -->
- <!--output-pdf--> väljatrükk avaneb alati PDFina (aruannete puhul faili nime määramiseks tasub ka mailattachment parameeter määrata - vt järgmine punkt)
- <!--mailattachment:attachmendi/faili_nimi -->saab kasutada tagi {nr} ning parameetreid {param:parami_nimi}
- <!--mailsubject:uus_subjekt -->saab kasutada tagi {nr} ning parameetreid {param:parami_nimi}
- <!--output-file:failinimi.txt -->väljund faili
- <!--charset:windows-xxx -->tavapäraselt on *output-file* puhul on encodinguks windows-1257, seda juhul, kui xsl-is pole kusagil *charset=utf-8* (siis on encodinguks mõistagi UTF-8). Kui aga määrata charset parameeter eraldi, siis võetakse siin määratud encoding

Parameetrite **mailsubject, mailattachment, output-file, mailfrom, mailto ja mailname** puhul saab kasutada:

- lisaparameetreid {param:parami_nimi} nimena on kasutuses: aeg1, aeg2, projekt, objekt, keel, klient_kood, ladu, tingimus, nimi võib dokumenditi veidi erineda (klient_kood vs kl_kood), vastavalt nagu ta XMLis defineeritud on
- dokumendi numbrit {nr} ning kuupäeva kuu aasta päev vastavalt {mm} {yy} {dd}
- mingit andmevälja XMList {param:xml:/documents/kontakt/klass}

XSLi parameetri kasutuse näited

- <!--mailsubject:ToreAruanne alates {param:aeg1} kuni {param:aeg2} -->
- <!--mailsubject: pakkumiskutse {nr} Objektile: {param:objekt} -->
- <!--mailattachment: minuManus_number_{nr}_klassile{param:xml:/documents/kontakt/klass} -->

Vihje: et testida, võib panna Kasutaja kaardilt peale „Maili redigeerimine: Jah“, siis avatakse meili aken juba valmis ehitatud subjektiga (et näha kas kõik parameetrid toimisid)

Ribakood pildina

Ribakoodi saab luua pildina, selleks tuleb luua **img** element ning selle **src** väärtuseks sisestada ribakoodi generaatori aadress konkreetsete parameetritega. Näide annab parameetrina kaasa XMList välja *artikkel*

```
<img><xsl:attribute name="src">/logos/qr.asp?t=<xsl:value-of
select="artikkel"/>&amp;code=C128&amp;h=30</xsl:attribute></img>
```

Võimalikud parameetrid:

- **t** - sisend, millest ribakood luuakse (olenevalt standardist võib selleks vabalt olla ka tekst)
- **h** - kõrgus pixelites
- **code** - soovitud ribakoodi standard (vastavalt järgnevale tabelile)

Standard	**code parameetri väärtus**
QR Code	qr
CODE 39	C39
CODE 39 CHECKSUM	C39c
CODE 39E	C39E
CODE 39E CHECKSUM	C39Ec
CODE 93	C93
STANDARD 2 5	S25
STANDARD 2 5 CHECKSUM	S25c
INTERLEAVED 2 5	I25
INTERLEAVED 2 5 CHECKSUM	I25c
CODE 128	C128

Standard	**code parameetri väärtus**
CODE 128 A	C128A
CODE 128 B	C128B
CODE 128 C	C128C
EAN 2	EAN2
EAN 5	EAN5
EAN 8	EAN8
EAN 13	EAN13
UPC A	UPCA
UPC E	UPCE
MSI	MSI
MSI CHECKSUM	MSIc
POSTNET	POSTNET
PLANET	PLANET
RMS4CC	RMS4CC
KIX	KIX
IMB	IMB
CODABAR	CODABAR
CODE 11	CODE11
PHARMA CODE	PHARMA
PHARMA CODE TWO TRACKS	PHARMA2T
AusPost 4 State Customer Code	auspost
Aztec Code	azteccode
Compact Aztec Code	azteccodecompact
Aztec Runes	aztecrune
BC412	bc412
Channel Code	channelcode
Codablock F	codablockf
Code 11	code11
Code 128	code128
Code 16K	code16k
Code 25	code2of5
Italian Pharmacode	code32
Code 39	code39
Code 39 Extended	code39ext
Code 49	code49
Code 93	code93
Code 93 Extended	code93ext
Code One	codeone
COOP 2 of 5	coop2of5
Custom 4 state symbology	daft
GS1 DataBar Expanded	databarexpanded
GS1 DataBar Expanded Composite	databarexpandedcomposite
GS1 DataBar Expanded Stacked	databarexpandedstacked
GS1 DataBar Expanded Stacked Composite	databarexpandedstackedcomposite
GS1 DataBar Limited	databarlimited

Standard	**code parameetri väärtus**
GS1 DataBar Limited Composite	databarlimitedcomposite
GS1 DataBar Omnidirectional	databaromni
GS1 DataBar Omnidirectional Composite	databaromnicomposite
GS1 DataBar Stacked	databarstacked
GS1 DataBar Stacked Composite	databarstackedcomposite
GS1 DataBar Stacked Omnidirectional	databarstackedomni
GS1 DataBar Stacked Omnidirectional Composite	databarstackedomnicomposite
GS1 DataBar Truncated	databartruncated
GS1 DataBar Truncated Composite	databartruncatedcomposite
Datalogic 2 of 5	datalogic2of5
Data Matrix	datamatrix
Data Matrix Rectangular	datamatrixrectangular
DotCode	dotcode
EAN-13	ean13
EAN-13 Composite	ean13composite
GS1-14	ean14
EAN-2 (2 digit addon)	ean2
EAN-5 (5 digit addon)	ean5
EAN-8	ean8
EAN-8 Composite	ean8composite
Flattermarken	flattermarken
GS1-128	gs1-128
GS1-128 Composite	gs1-128composite
GS1 Composite 2D Component	gs1-cc
GS1 Data Matrix	gs1datamatrix
GS1 Data Matrix Rectangular	gs1datamatrixrectangular
GS1 North American Coupon	gs1northamericancoupon
GS1 QR Code	gs1qrcode
Han Xin Code	hanxin
HIBC Aztec Code	hibcazteccode
HIBC Codablock F	hibccodablockf
HIBC Code 128	hibccode128
HIBC Code 39	hibccode39
HIBC Data Matrix	hibcdatamatrix
HIBC Data Matrix Rectangular	hibcdatamatrixrectangular
HIBC MicroPDF417	hibcmicropdf417
HIBC PDF417	hibcpdf417
HIBC QR Code	hibcqrcode
IATA 2 of 5	iata2of5
Deutsche Post Identcode	identcode
Industrial 2 of 5	industrial2of5
Interleaved 2 of 5 (ITF)	interleaved2of5
ISBN	isbn
ISMN	ismn
ISSN	issn

Standard	code parameetri väärtus
ITF-14	itf14
Japan Post 4 State Customer Code	japanpost
Royal Dutch TPG Post KIX	kix
Deutsche Post Leitcode	leitcode
Matrix 2 of 5	matrix2of5
MaxiCode	maxicode
MicroPDF417	micropdf417
Micro QR Code	microqrcode
MSI Modified Plessey	msi
USPS Intelligent Mail	onecode
PDF417	pdf417
Compact PDF417	pdf417compact
Pharmaceutical Binary Code	pharmacode
Two-track Pharmacode	pharmacode2
USPS PLANET	planet
Plessey UK	plessey
PosiCode	posicode
USPS POSTNET	postnet
Pharmazentralnummer (PZN)	pzn
QR Code	qrcode
Codabar	rationalizedCodabar
Custom 1D symbology	raw
Royal Mail 4 State Customer Code	royalmail
SSCC-18	sscc18
Miscellaneous symbols	symbol
Telepen	telepen
Telepen Numeric	telepennumeric
Ultracode	ultracode
UPC-A	upca
UPC-A Composite	upcacomposite
UPC-E	upce
UPC-E Composite	upcecomposite

Code128 ribakoodi kasutamine skriptiga

Code 128 standardile vastavat ribakoodi saab väljatrükis kasutada ilma väliste fontide olemasoluta. Lahendusena on pakkuda väljatrükk, kus ribakoodi tarbeks genereeritakse pilt html-is

Tähelepanu tuleks pöörata järgnevale:

Scripti kasutamine XSLis

<xsl:stylesheet> lehe deklareerimisel, peavad olema defineeritud ka järgnevad script atribuudid:

- `xmlns:script = "http://topxml.com/forum/script"`

- `xmlns:x="ignore" exclude-result-prefixes="script"`

```
<xsl:stylesheet version="1.0"  xmlns:str="http://xsltsl.org/string"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format" xmlns:msxsl="urn:schemas-
microsoft-com:xslt" extension-element-prefixes="str" xmlns:script =
"http://topxml.com/forum/script"   xmlns:x="ignore" exclude-result-
prefixes="script">
  <xsl:output method="html"/>
  <xsl:decimal-format name="ocra" decimal-separator='.' grouping-separator='
' />

 <msxsl:script language="JScript" implements-prefix="script">
<![CDATA[
]]>
  </msxsl:script>

</xsl:stylesheet>
```

Ribakoodi funktsioon

Kõik <msxsl:script> blokis tuleks ka oma kujundusse kopeerida (siin asub funktsioon, mille abil tehakse ribakoodi pilt)

Funktsiooni väljakutse

- Ribakoodi kuvamiseks kutsutakse välja eelpool mainitud funktsioon

  ```
  <xsl:value-of disable-output-escaping="yes"
  select="string(script:code128(string(artikkel)))"/>
  ```

- Võib juhtuda, et mõningate sümbolite korrektsena kuvamiseks tuleb ribakoodilugejat seadistada õiget tüüpi klaviatuuri emuleerima (testitud käpaga õnnestus kood lugeda, kui klaviatuuri tüübiks oli määratud SWE/FIN)

Näidis

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format"
  xmlns:msxsl="urn:schemas-microsoft-com:xslt"
  xmlns:script = "http://topxml.com/forum/script"
  xmlns:x="ignore" exclude-result-prefixes="script">
```

```
   <xsl:output method="html"/>
   <xsl:decimal-format name="ocra" decimal-separator='.' grouping-
separator=' ' />
   <xsl:template match="/">
      <meta http-equiv="Content-Type" content="text/html;
charset=utf-8"></meta>
    <html>
    <style>
                .bcdiv {float:left;height: 30px; /*size*/}
        .bar1 { border-left:1px solid black; }
        .bar2 { border-left:2px solid black; }
        .bar3 { border-left:3px solid black; }
        .bar4 { border-left:4px solid black; }
        .space0 { margin-right:0 }
        .space1 { margin-right:1px }
        .space2 { margin-right:2px }
        .space3 { margin-right:3px }
        .space4 { margin-right:4px }r
        label {clear:both;display:block;text-align:center; font:
0.125in/100% helvetica;}
    </style>
    <body>
        <table>
        <xsl:for-each select="/documents/document/rows/row">
            <tr>
                <td><xsl:value-of select="artikkel"/></td>
                <td><xsl:value-of disable-output-escaping="yes"
select="string(script:code128(string(artikkel)))"/></td>
            </tr>
        </xsl:for-each>
        </table>
    </body>
    </html>
</xsl:template>

<msxsl:script language="JScript" implements-prefix="script"><![CDATA[
var BARS       =
[212222,222122,222221,121223,121322,131222,122213,122312,132212,221213,22131
2,231212,112232,122132,122231,113222,123122,123221,223211,221132,221231,2132
12,223112,312131,311222,321122,321221,312212,322112,322211,212123,212321,232
121,111323,131123,131321,112313,132113,132311,211313,231113,231311,112133,11
2331,132131,113123,113321,133121,313121,211331,231131,213113,213311,213131,3
11123,311321,331121,312113,312311,332111,314111,221411,431111,111224,111422,
121124,121421,141122,141221,112214,112412,122114,122411,142112,142211,241211
,221114,413111,241112,134111,111242,121142,121241,114212,124112,124211,41121
2,421112,421211,212141,214121,412121,111143,111341,131141,114113,114311,4111
13,411311,113141,114131,311141,411131,211412,211214,211232,23311120]
    , START_BASE = 38
    , STOP       = 106 //BARS[STOP]==23311120 (manually added a zero at the
end)
    ;
```

```
  function code128(code, barcodeType) {
    if (arguments.length<2) barcodeType = code128Detect(code);
    if (barcodeType=='C' && code.length%2==1) code = '0'+code;
    var a = parseBarcode(code,  barcodeType);
    return bar2html(a.join('')) + '<label>' + code + '</label>';
  }

  function bar2html(s) {
    for(var pos=0, sb=[]; pos<s.length; pos+=2) {
      sb.push('<div class="bcdiv bar' + s.charAt(pos) + ' space' +
s.charAt(pos+1) + '"></div>');
    }
    return sb.join('');
  }

  function code128Detect(code) {
    if (/^[0-9]+$/.test(code)) return 'C';
    if (/[a-z]/.test(code)) return 'B';
    return 'A';
  }

  function parseBarcode(barcode, barcodeType) {
    var bars = [];
    bars.add = function(nr) {
      var nrCode = BARS[nr];
      this.check = this.length==0 ? nr : this.check + nr*this.length;
      this.push( nrCode || ("UNDEFINED: "+nr+"->"+nrCode) );
    };
    bars.add(START_BASE + barcodeType.charCodeAt(0));
    for(var i=0; i<barcode.length; i++) {
      var code = barcodeType=='C' ? +barcode.substr(i++, 2) :
barcode.charCodeAt(i);
      converted = fromType[barcodeType](code);
      if (isNaN(converted) || converted<0 || converted>106) throw new
Error("Unrecognized character ("+code+") at position "+i+" in code
'"+barcode+"'.");
      bars.add( converted );
    }
    bars.push(BARS[bars.check % 103], BARS[STOP]);
    return bars;
  }
  var fromType = {
    A: function(charCode) {
      if (charCode>=0 && charCode<32) return charCode+64;
      if (charCode>=32 && charCode<96) return charCode-32;
      return charCode;
    },
    B: function(charCode) {
      if (charCode>=32 && charCode<128) return charCode-32;
      return charCode;
```

```
    },
    C: function(charCode) {
      return charCode;
    }
  }

]]></msxsl:script>

</xsl:stylesheet>
```

Lingi kasutamine

```
<a target="_blank">
    <xsl:attribute name="href">
       <xsl:value-of disable-output-escaping = "yes"
select="artikkel_andmed/url"/>
    </xsl:attribute>
    <xsl:value-of disable-output-escaping = "yes" select="keelne_seletus"/>
</a>
```

* Antud näites artikli pealkirjale vajutades suunatakse lingile, mis on seadistatud artiklikaardi väljale URL. Väline link peab olema kujul

```
http://www.directo.ee
```

Signeerimise tugi

Signeerimine väljatrüki kontekstis tähendab seda, et dokumendile kuvatakse nupp **Signeeri**. Tavaliselt on see kasutuses olukorras, kus on vaja kliendilt küsida allkirja, näiteks mingi akti allkirjastamine kliendi juuresolekul . Selleks luuaks spetsiaalne väljatrükk signeerimiseks, Signeeri nupu vajutus dokumendil avab selle ning väljatrükil kuvatakse ala allkirjastamisesk, mida saab siis kas arvutist hiirega või tahvlist näpuga „sodides“ allkirjastada. Peale **Valmis** nupu vajutamist salvestub väljatrükk koos joonistatud allkirjaga väljatrükis ettenähtud kohta selle sama dokumendi külge PDF manuseks.

Dokumendid, mis toetavad signeerimist:

- Pakkumine
- Tellimus
- Lähetus
- Arve
- Sündmus
- Klient
- Leping
- Liikumine
- Personal

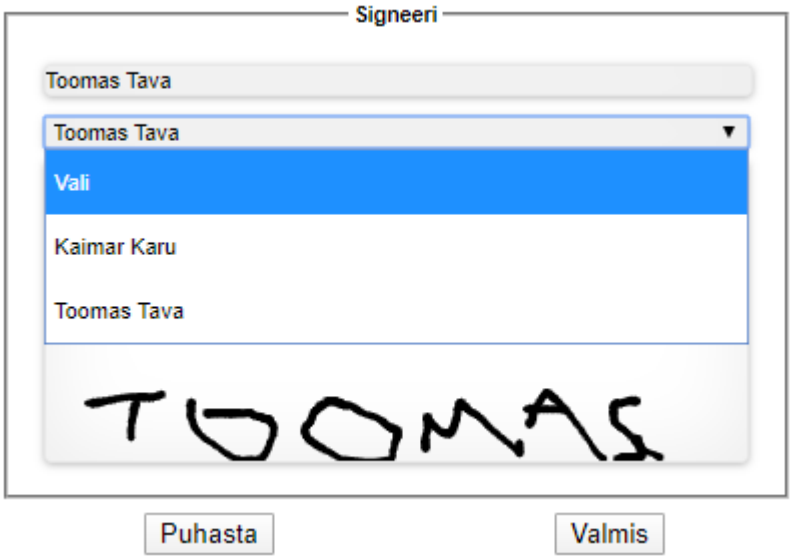
Signeerimise väljatrüki lisamiseks tuleb väljatrüki definitsioonile määrata tulbast **Signeeri**

signeerimise ala laius :

XSL-is tuleb allkirja koht määrata järgmiselt:

```
<img sign="signhere" width="600" border="0"  />
```

- Soovikorral kuvatakse eraldi sisestuskast, kuhu saab tekstina sisestada allkirjastaja nime.



Selleks tuleb lisada väljatrükki eraldi *div* näites toodud id-ga, sisestatud nimi kuvatakse *div*i sisse. Võimalus on ka kuvada eeldefineeritud nimedega rippmenüü (lang atribuut)

```
<div lang="Kaimar Karu,Toomas Tava" id="signer"></div>
```

atribuut id=„signer“ peab olema div-i viimane atribuut.

- võimalus on signeerimist kasutada ka lihtsalt nö manusena PDFi salvestajana. Sellisel juhul signeerimist ei toimu, lihtsalt signeerimise väljatrükk salvestatakse PDFina dokumendi manuseks. Selle saavutamiseks tuleks lisada XSLi ülal mainitud **img** tagi **emulator** atribuut:

```
<img sign="signhere" emulator="1" width="1" border="0"    />
```

EveryPay Linkpay

EveryPay Linkpay abil on võimalik arve väljatrükile luua unikaalne makselink.

Alustamine

LinkPay lahenduse kasutamiseks Directos võta ühendust endale sobivaima EveryPay partnerpangaga, milleks on kas LHV, SEB või Swedbank. Sind saab aidata sinu kliendihaldur või täida ise ära vastava panga taotlusvorm. Hinnainfo koos lepingu tingimustega saad otse pangalt. Loe, kuidas täpsemalt kasutada LinkPayd EveryPay portaalist.

Seadistamine

1. EveryPay portaalis tuleb luua **LINKPAY>Lingid** alt link
2. Lingi detailvaates on kuvatud **Lingi token**, mis tuleb kopeerida ning määrata Directo süsteemi seadistustes `EveryPay LinkPay lingi token` väärtuseks
3. Lingi *Muuda* vaates tuleb *Aktiivne* tulbas valida aktiivseks ainult *Arve number* väli
4. EveryPay portaali **SEADED>Üldised seaded** alt kopeerida **API salasõna** Directo süsteemiseadistuse `EveryPay LinkPay API salasõna` väärtuseks
5. Süsteemi seadistus `EveryPay LinkPay URL` on vaikimisi täidetud testkeskkonna aadressiga https://igw-demo.every-pay.com Pärast testimist tuleb see täita live teenuse aadressiga https://pay.every-pay.eu
6. Link tekib arve XMLi, kui väljatrüki definitsiooni aknasse **Väljad** alt on valitud *EveryPay LinkPay link*

   See valik tekib automaatselt peale salvestamist, kui lisada *everypay_link* XMLi tag järgmises punktis toodud näite alusel.
7. Lingi võib lisada väljatrükile sobivasse kohta, loomise näide:

```
<a><xsl:attribute name="href"><xsl:value-of
select="/documents/document/everypay_link" disable-output-escaping =
"yes"/></xsl:attribute>MAKSA SIIN</a>
```

Vajadusel saab mitme lingi olemasolul luua eraldi loogika, mis valib mis iganes kriteeriumi põhjal sobiva (eeldefineeritud) lingi, mida konkreetsele väljatrükile luuakse. Seadistamiseks kirjuta palun info@directo.ee

HMAC signatuuri loomine välisele lingile

Võimalus on luua signatuur arvest (või tellimusest) välisele osapoolele.

- Signatuur luuakse dokumendi summa täisosa ja numbri omavahel liitmisel eraldatuna **&** märgiga ning selle signeerimisega osapoolte vahel jagatud salatunnusega **SHA2 256** algoritmi alusel.
- Salatunnus määratakse Directos süsteemiseadistuse `Väljatrüki HMAC signatuuri salatunnus` väärtuseks.
- Signatuuri saab lisada väljatrükile sobivasse kohta (mingi välise lingi osana):

```
<a><xsl:attribute
name="href">http://www.minukoht.ee/?hmac=<xsl:value-of
```

```
select="/documents/document/hmac_signature" disable-output-escaping =
"yes"/></xsl:attribute>Mingi link</a>
```

Näide

Arve nr: 201400285 Summa : 136.30

```
$secret = "85b97cd7a1"; # Directo süsteemiseadistustes salatunnus
$data = "sum=136&num=201400285" #hashitav string NB! arve summas ainult
täisosa
$hmac = hash_hmac("sha256", $data, $secret); #
d62f1b4761cede20b37c189aab95a55fb60e8b4f8a98c4e74194a47622b7a07a
```

Nipid

0 (mitte NaN), kui pole andmeid

- Päisesse numbri formaat paika

```
<xsl:decimal-format name="N" NaN="0" decimal-separator='.' grouping-
separator=' ' />
```

- Kasutamine

```
<xsl:value-of select="format-number(mingi_vali, '0.00', 'N')"/>
```

Mingi bloki X korda joonistamine

- Luua eraldi template, kus on väljund, mida joonistatakse

```
<xsl:template name="kast">
    <xsl:param name="count"/>

    <xsl:choose>
      <xsl:when test="$count &lt;= 0"/>

      <xsl:otherwise>
        <!-- Sisu mida tahame X korda kuvada-->
        <table border="1">
          <tr>
            <td>Mingi sisu</td>
          </tr>
        </table>

        <xsl:call-template name="kast">
```

```
          <xsl:with-param name="count" select="$count - 1"/>
        </xsl:call-template>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>
```

- Kutsume joonistaja välja nii palju kui vaja

```
<xsl:call-template name="kast">
          <xsl:with-param name="count" select="2"/>
        </xsl:call-template>
```

Node-set XML muutuja, sealt unikaalsete kirjete kuvamine

- stylesheet päises peab olema atribuut **xmlns:msxsl="urn:schemas-microsoft-com:xslt"**
- XML struktuuriga muutuja **andmed** loomine mingi olemasoleva XMLi põhjal

```
<xsl:variable name="andmed" xmlns="">
    <xsl:for-each select="/document/rows/row">
        <rida>
            <artikkel><xsl:value-of select="artikkel"/></artikkel>
            <nimi><xsl:value-of select="nimi"/></nimi>
        </rida>
    </xsl:for-each>
</xsl:variable>
```

xmlns="" on vajalik Excel-i XML Spreadsheet 2003 formaadis väljundi jaoks (muidu pole võimalik loodud XMList näiteks sum() teha).

- Ülal loodud muutuja andmete kuvamine
- tänu filtrile **[not(artikkel = preceding-sibling::rida/artikkel)]** kuvatakse ainult unikaalsed artiklikoodid

```
<table>
    <xsl:for-each select="msxsl:node-set($andmed)/rida[not(artikkel =
preceding-sibling::rida/artikkel)]">
        <xsl:sort select="artikkel" data-type="text"
order="ascending"/>

        <tr>
            <td>Artikkel:  <xsl:value-of select="artikkel"/></td>
            <td>Nimi:  <xsl:value-of select="nimi"/></td>
        </tr>
    </xsl:for-each>
</table>
```

dateAdd - kuupäevale mingi päevade arvu lisamine

- XSL- päises peab olema skripti deklaratsioon
- dateaAdd funktsioon

```
function dateadd(time, paevi)
{

       var d, time, time_spl,aaa, a;
       time_spl=time.split(' ');
       a = time_spl[0].split('.');
       aaa = new Date(a[2], a[1]-1, Number(a[0])+Number(paevi));
       d=String(((aaa.getDate()<9)?'0':''
)+aaa.getDate())+'.'+((aaa.getMonth()<9)?'0':'') +
String(aaa.getMonth()+1)+'.'+String(aaa.getFullYear());
       return(d);
}
```

- Funktsiooni kutsumine

```
          <xsl:for-each select="/documents/document/rows/row">
            <xsl:variable name="paevi"
select="artikkel_andmed/garantii"/>
              <tr>

                <td height="10" valign="top">
                  <xsl:value-of select="ribakood"/>
                </td>
                  <td> <xsl:value-of
select="artikkel_lisavaljad/lisa[@kood='REALT']"/> <xsl:value-of
select="string(script:dateadd(string(../../lahetusaeg),
string($paevi)))"/>
                  </td>
                </td>
              </tr>
```

Splitter - Mingi tekstilise sisu laiali jagamine separaatori alusel

```
* Parameetrid 1. String, 2. separaator (näiteks tühik ' '), 3. mitmes
element tagastada (algab 0-iga, ehk siis esimene tagastatav element on 0)
* XSL- päises peab olema
[[et:yld_print_form#scripti_kasutamine_xslis|skripti deklaratsioon]]
* splitterfunktsioon <code js>function splitter( str, splitter, pos ){
    ret = '';
  var arr = str.split(splitter);
  if(arr[pos])
    ret = arr[pos]
    return ret
}</code>
```

```
* Funktsiooni kutsumine <code xml>          <xsl:for-each
select="/documents/document/rows/row">
                <td> <xsl:value-of select="string(splitter(string(nimetus),'
', 1))"/>
                </td>
            </tr>
```

</code>

dateDiff - kuupäevade vahemiku päevade arvu arvutamine

- XSL- päises peab olema skripti deklaratsioon
- dateDiff funktsioon (NB! tegemist on VbScript-iga)

```
<msxsl:script language="VbScript" implements-prefix="script">
    <![CDATA[
function ddiff(m,d1,d2)
    ddiff = DateDiff(m,d1,d2)
end function
]]>
  </msxsl:script>
```

- Funktsiooni kutsumine

```
 <xsl:value-of select="script:ddiff('d',string(aeg1),string(aeg2))"/>
```

- Funktsiooni kutsumine juhul, kui vaja, et arvestaks ka algkuupäeva

```
 <xsl:value-of select="script:ddiff('d',string(r_aeg1),string(r_aeg2))
+ 1"/>
```

Textarea sisu (rea vahedega, näiteks sündmuse kirjeldus) kuvamine ka väljatrükis reavahedega

Kuna reavahetuse ei kajastu HTML-is siis kasutame <pre> tagi.

- tähtis on, et oleks määratud ka konteineri laius, ehk siis see, mille sees soovitud algne sisu asub, peaks omama **width** definitsiooni, selle näite puhul on see **<td>**
- <pre> puhul on vaikimisi kasutusel ka teine stiil, seega tuleks fondid jms määrata ka <pre> tagile

```
<td valign="top" style="width:700px" >
    <pre style="white-space: pre-wrap; word-wrap: break-word;">
        <xsl:value-of select="/documents/document/sisu"/>
     </pre>
</td>
```

Page break

Et printeris/PDFis tekkiks uus leht:

```
    <div style="page-break-before: always;">
```

Summeerimine

Univeraalne summeerimise funktsioon, parameetrid:

1. sisend number mida lisatakse
2. grupp (suvaline konteksti kirjeldav string, kui üks summimine siis võib olla näiteks summ)
3. väljund
    - 0 - sisend number
    - 1 - grupi summa
    - 2 - tühjus

Väljund on string tüüpi, seega kui on vaja tulemusega arvutada, tuleb see numbriks konvertida enne. 1 ja 2 nullivad ka hetkel grupi.

```
<xsl:stylesheet version="1.0"  xmlns:str="http://xsltsl.org/string"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format" xmlns:msxsl="urn:schemas-
microsoft-com:xslt" extension-element-prefixes="str" xmlns:script =
"http://topxml.com/forum/script"   xmlns:x="ignore" exclude-result-
prefixes="script">
  <xsl:output method="html"/>
  <xsl:decimal-format name="ocra" decimal-separator='.' grouping-separator='
' />


<xsl:for-each select="/documents/document/rows/row[rv=rn]">

  <xsl:choose>
    <xsl:when test ="artikkel='VAHESUMMA'">
        <xsl:value-of select="string(script:summer(0,'summ', 1))"/><!-- siin
kuvatakse grupi summa-->
    </xsl:when>
    <xsl:otherwise>
        <xsl:if test="kogus!='' and kogus!=0">
            <xsl:value-of select="format-number( string(script:summer(
number(rv_summa),'summ', 0)) , '### ##0.00', 'ocra')"/> <!-- siin lisatakse
grupile mingi väärtus-->
        </xsl:if>
    </xsl:otherwise>
  </xsl:choose>

</xsl:for-each>
```

```
 <msxsl:script language="JScript" implements-prefix="script">
<![CDATA[

    var sums = [];

    function summer( v, grp, ret ){
        var rt = '';
        if( !sums[grp] ){
            sums[grp] = 0;
        }
        sums[grp]+=Number(v);

        if( ret == 0 )
            rt = ''+v;
        if( ret == 1 )
            rt = ''+Number(sums[grp]);
        if( ret == 1 || ret == 2 )
            sums[grp] = 0;
        return rt
    }

]]>
  </msxsl:script>

</xsl:stylesheet>
```

Palgateatisele puhkusejäägi kuvamine

```
<xsl:variable name="pjaak1" select="substring-before(substring-after(pjaak,
';'), ';')"/>
<xsl:variable name="pjaak2" select="substring-before(substring-
after(substring-after(pjaak, ';'), ';'), ';')"/>
<xsl:variable name="pjaak3" select="substring-before(substring-
after(substring-after(substring-after(pjaak, ';'), ';'), ';'), ';')"/>
<xsl:variable name="pjaak4" select="substring-after(substring-
after(substring-after(substring-after(pjaak, ';'), ';'), ';'), ';')"/>

<xsl:value-of select="format-number($pjaak2 - $pjaak3 - $pjaak1 - $pjaak4,
'#####0')"/>
```

From:
https://wiki.directo.ee/ - **Directo Help**

Permanent link:
https://wiki.directo.ee/et/yld_print_form?rev=1601895923

Last update: **2020/10/05 14:05**