Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ульяновский государственный технический университет»

Факультет информационных систем и технологий

Кафедра: «Вычислительная техника»

«Программное обеспечение АС»

Отчет по выполнению лабораторной работ

«Разработка программ, реализующих пользовательский интерфейс»

Отчет составил

Магистр группы ИВТАСмд-11

Кондратьев П.С.

Отчет принял:

Проф. каф. ВТ, д.т.н

Токмаков Г. П.

г. Ульяновск, 2021

**Цель:** создание документа, в привычный пользователь в виде, на основе шаблона XML - документа.

**Ход работы:**

Приступим к созданию информации, которая будет хранить данные о форматировании документа, для этого для создадим новое визуальное представление документа в программе Altova StyleVision выберем на панели инструментов New design from Xml Schema, выбираем New Xml Schema.

После чего выбираем Xml Schema, которую создали в прошлых лабораторных работах, далее программа запросит рабочий Xml документ (xml-документ с заполненными данными), это делается для того чтобы мы получили конкретные данные для создания экземпляра документа. Следующим действием выбираем Create a free-flow document.

После чего у нас создастся файл с расширением .sps, в котором мы и будем создавать дизайн нашего будущего документа.

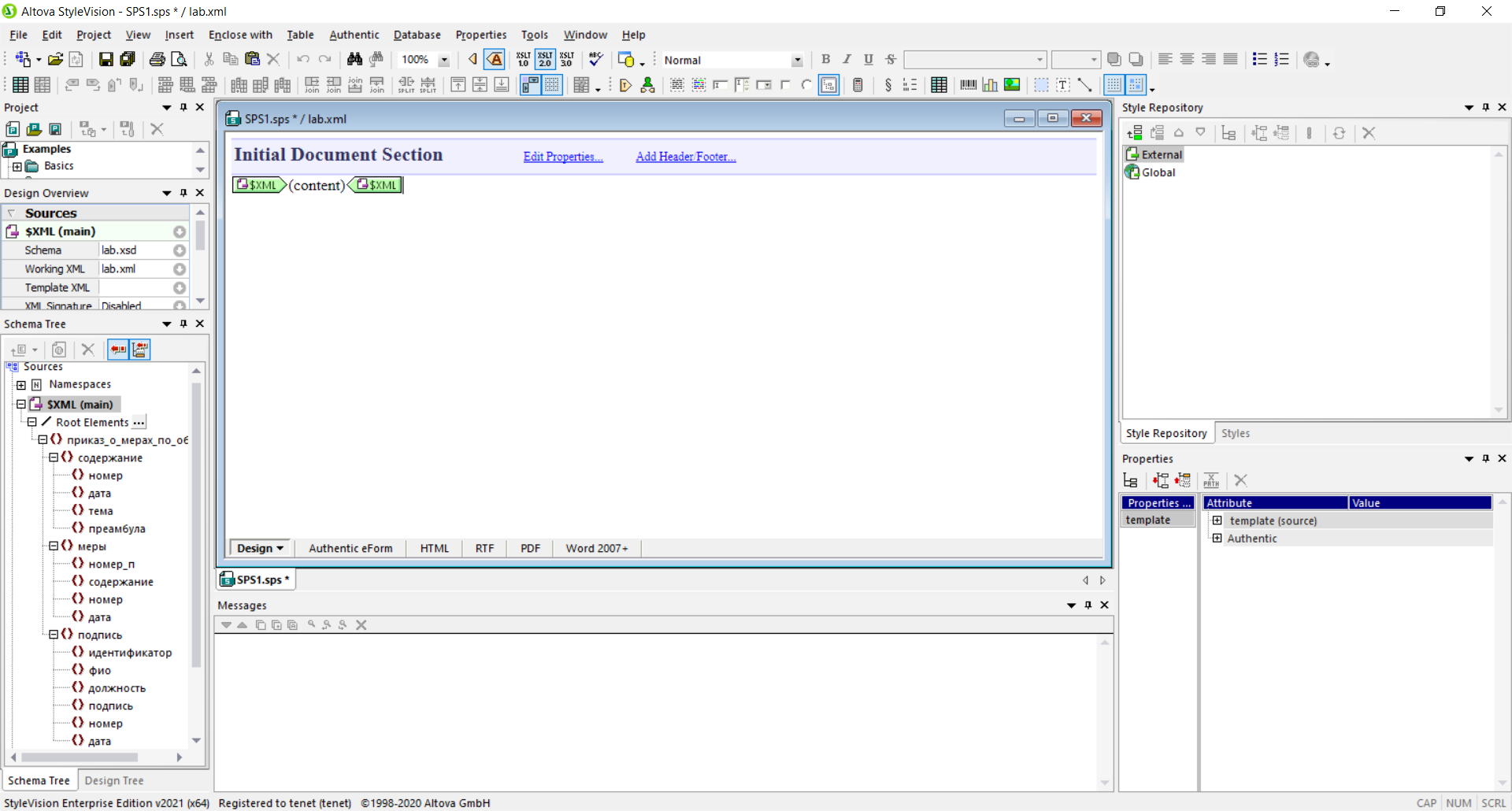


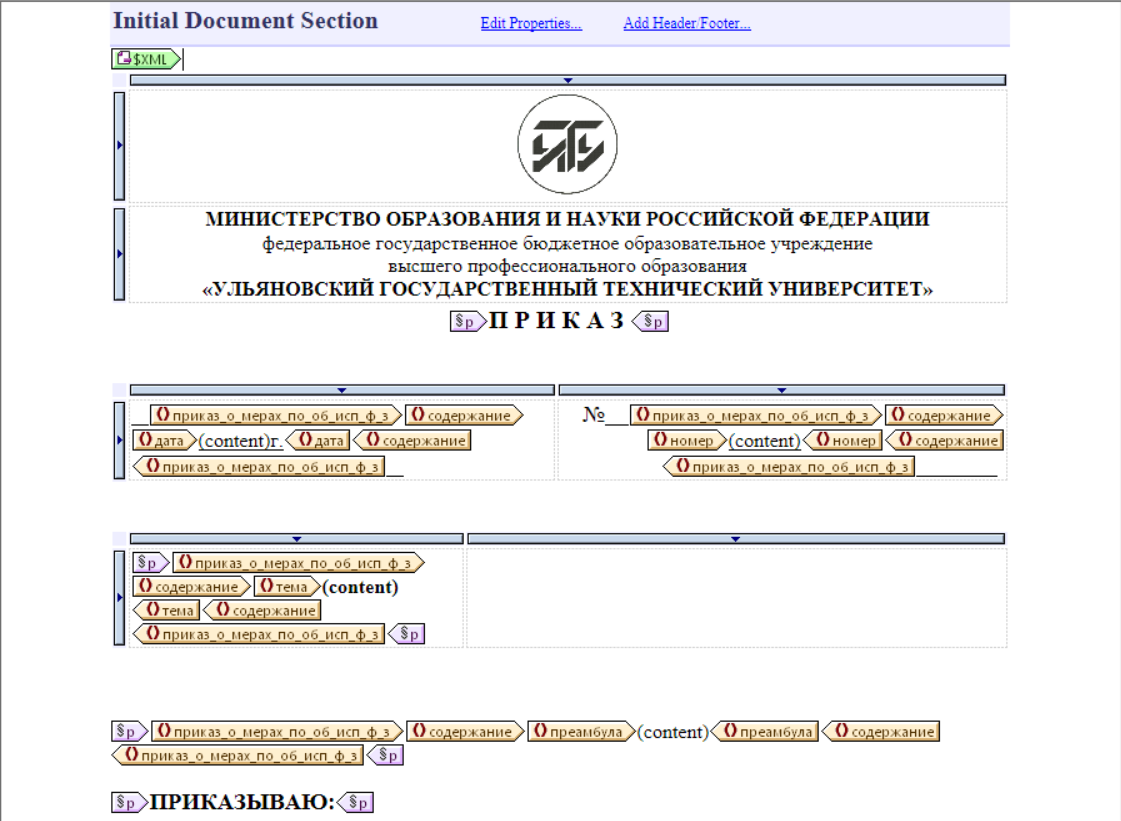
Рис. 1 Созданный sps файл документа

В левой части экрана находятся боковая панель с дерево-решений проекта (Project), выбранные (загруженные) схемы (Design Overview) и древовидная схема xml файла (Schema Tree), с данными, которые будут использоваться для проектирования документа.

В центральной части экрана виден наш файл sps, в котором и будет производиться проектирование документа, в нем мы будет добавлять контент (таблицы, картинки, параграфы и т.д.) и составлять реальный документ.

В правой части экрана находится надстройки, связанные со стилями нашего документа, которые позволят установить правильные отступы, размеры, стиль текста, переносы строк и т.д.

Ниже на Рис.2. приведен шаблон искомого документа.



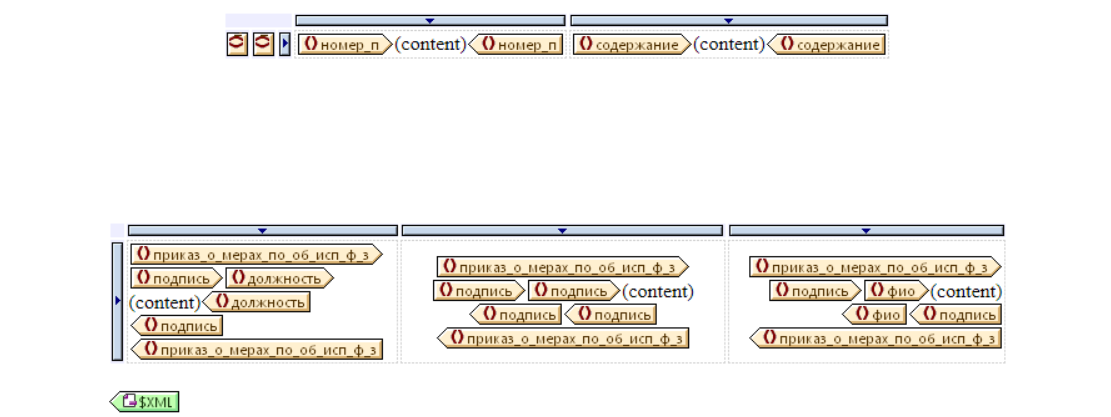


Рис.2. Шаблон документа

Все выше указные данные будут браться из Schema Tree, в котором мы выбираем нужный нам узел Xml документа и перетаскиваем его на файл редактора, после чего выбираем Create Content.

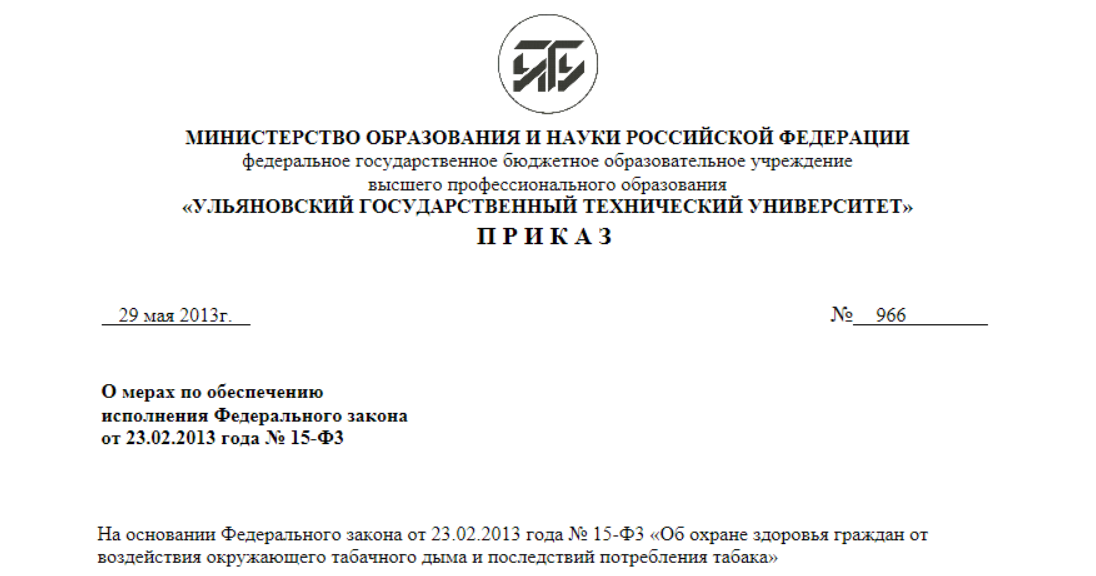
Текущий шаблон документа состоит из 3 частей: шапки, тела и подвала.

Шапка документа, сделанная с помощью таблицы размерами 2x1, где в первой строке добавлен логотип УлГТУ, во второй наименование органа управления. Далее идет параграф, добавленный с помощью sps редактора, после которого выведем информацию об документе (номер документа и дата), загруженного нами раннее Xml файла, используя туже самую таблицу.

Ниже в шаблоне часто используется таблица, для отображения данных, это нужно для более простой организации данных.

Тело документа состоит из преамбулы, которая берется из таблицы Содержание после перечисляются Меры, которые должны быть выполнены в документе. Для вывода конкретных полей из таблицы, создаются динамические таблицы, в которых указываем какие поля будут выведены в редакторе.

Подвал же состоит из таблицы 3x1, в которой просты выводим должность человека, ФИО и его подпись. По итогам получаем документ следующего содержания:



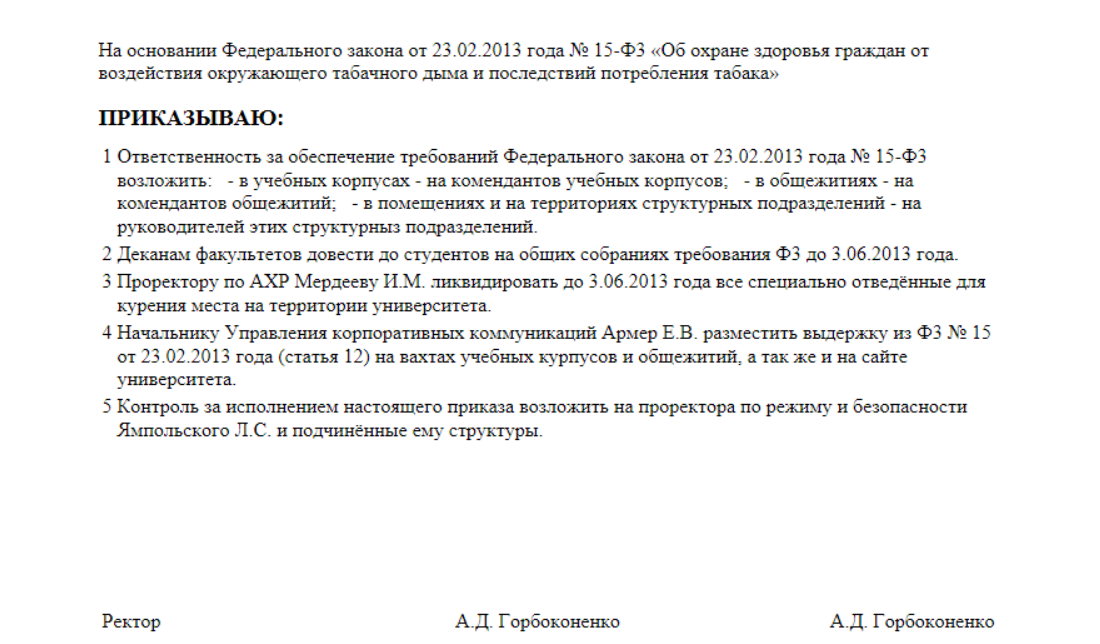
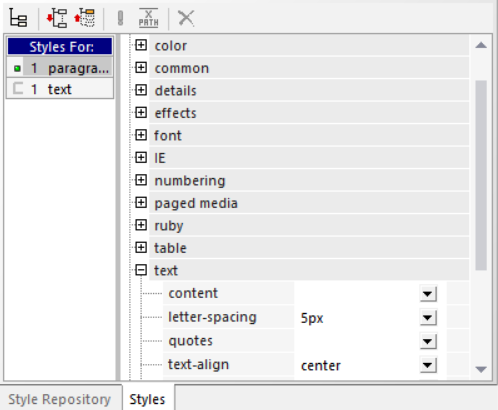


Рис.3. Составленный документ

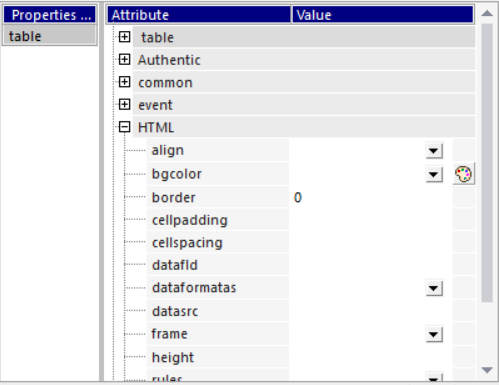
Так же для красоты, можем немного подправить стили страницы, делается это в правом боковой меню (Style Repository).

**Примеры**

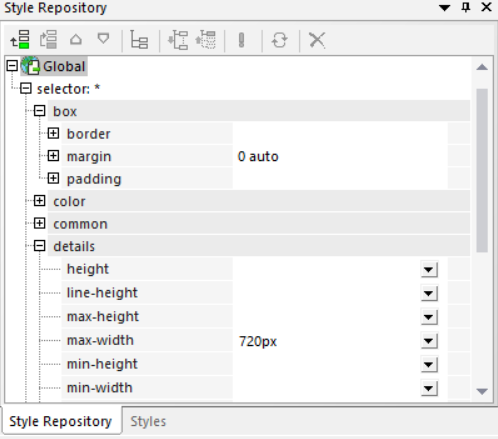
* Для слова **ПРИКАЗ**, было добавлено расстояние между буквами



* Для всех **таблиц** на страницы убраны границы (border)



* Для всей страницы было задано ограничение по ширине и центровка документа



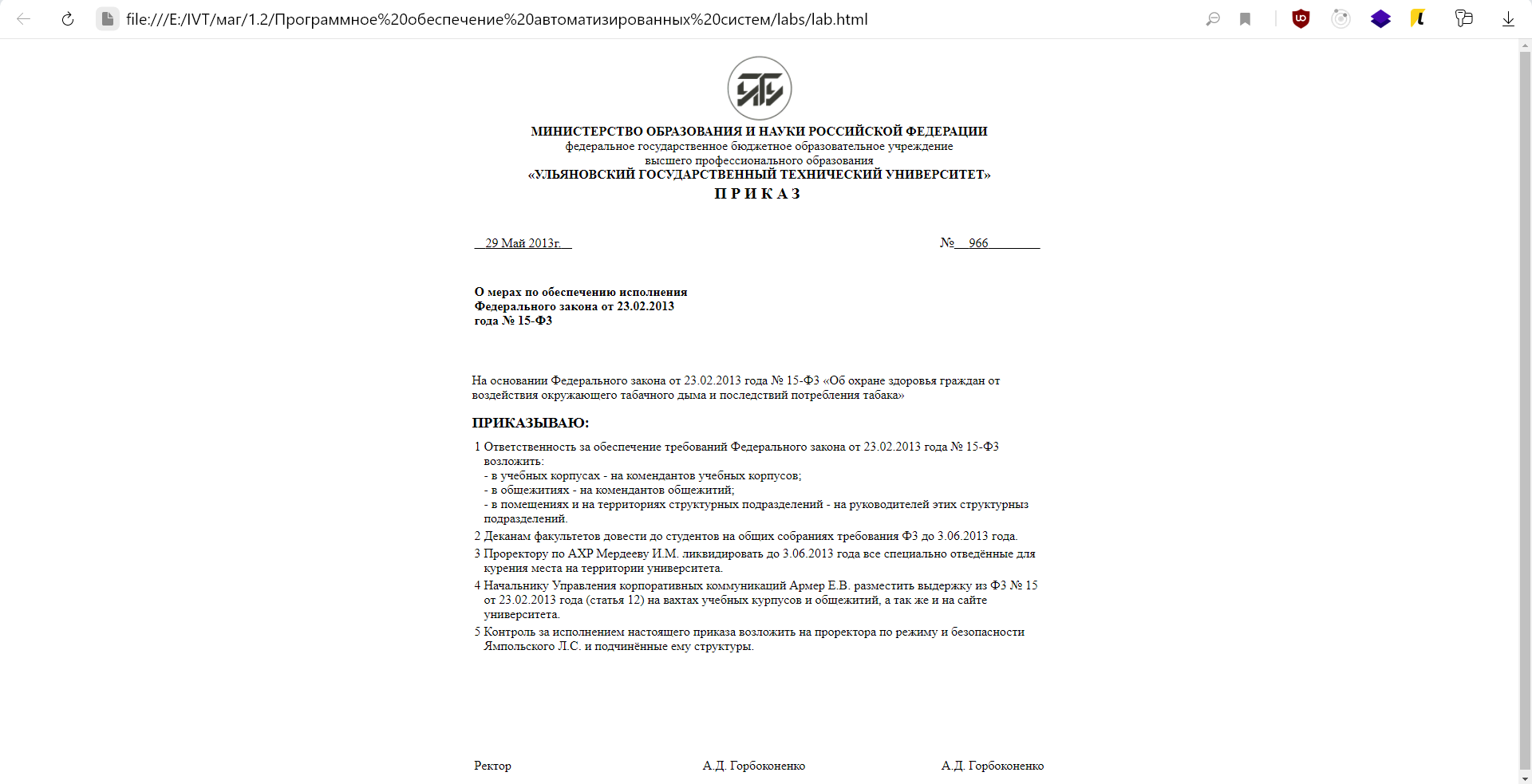
Таким образом получили образец искомого документа, после чего это оформление документа сохраняем в sps файла.

Также документ можно сохранить в:

* Xslt – см. Приложение 1
* Html – см. Приложение 2

**Вывод:** я научился создавать документ в привычном для пользователя в виде, на основе шаблона XML – документа, используя Altova StyleVision.

**Приложение 1**



**Приложение 2**

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <!--Designed and generated by Altova StyleVision Enterprise Edition 2021 (x64) - see http://www.altova.com/stylevision for more information.-->  <xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:altova="http://www.altova.com" xmlns:altova-xfi="http://www.altova.com/xslt-extensions/xbrl" xmlns:altovaext="http://www.altova.com/xslt-extensions" xmlns:array="http://www.w3.org/2005/xpath-functions/array" xmlns:clitype="clitype" xmlns:fn="http://www.w3.org/2005/xpath-functions" xmlns:iso4217="http://www.xbrl.org/2003/iso4217" xmlns:java="java" xmlns:link="http://www.xbrl.org/2003/linkbase" xmlns:map="http://www.w3.org/2005/xpath-functions/map" xmlns:math="http://www.w3.org/2005/xpath-functions/math" xmlns:sps="http://www.altova.com/StyleVision/user-xpath-functions" xmlns:vc="http://www.w3.org/2007/XMLSchema-versioning" xmlns:xbrldi="http://xbrl.org/2006/xbrldi" xmlns:xbrli="http://www.xbrl.org/2003/instance" xmlns:xff="http://www.xbrl.org/2010/function/formula" xmlns:xfi="http://www.xbrl.org/2008/function/instance" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" exclude-result-prefixes="#all">  <xsl:output version="4.0" method="html" indent="no" encoding="UTF-8" use-character-maps="spaces" doctype-public="-//W3C//DTD HTML 4.01 Transitional//EN" doctype-system="http://www.w3.org/TR/html4/loose.dtd"/>  <xsl:character-map name="spaces">  <xsl:output-character character="&#160;" string="&amp;#160;"/>  </xsl:character-map>  <xsl:param name="altova:bGeneratingFromPxf" select="false()"/>  <xsl:param name="altova:bGeneratingMime" as="xs:boolean" select="false()"/>  <xsl:param name="altova:execution-guid" as="xs:string">  <xsl:sequence select="altovaext:generate-guid()" use-when="function-available('altovaext:generate-guid', 0)"/>  <xsl:sequence select="'00000000-0000-0000-0000-NON0UNIQUE00'" use-when="not(function-available('altovaext:generate-guid', 0))"/>  </xsl:param>  <xsl:param name="SV\_OutputFormat" select="'HTML'"/>  <xsl:param name="SV\_BaseOutputFileName" as="xs:string?">  <xsl:sequence select="for $i in altovaext:get-base-output-uri(), $j in tokenize( $i, &apos;[/\\]&apos; )[last()] return replace( $j, &apos;\.[^\.\s#%;]\*$&apos;, &apos;&apos; )" use-when="function-available(&apos;altovaext:get-base-output-uri&apos;)"/>  </xsl:param>  <xsl:param name="SV\_GeneratedFileNamePrefix" select="if ( $SV\_BaseOutputFileName ) then $SV\_BaseOutputFileName else &apos;lab&apos;" as="xs:string?"/>  <xsl:variable name="XML" select="/"/>  <xsl:variable name="altova:nPxPerIn" select="96"/>  <xsl:decimal-format name="format1" grouping-separator=" " decimal-separator=","/>  <xsl:import-schema schema-location="lab.xsd" use-when="system-property('xsl:is-schema-aware')='yes'"/>  <xsl:variable name="altova:CssImages" select="()"/>  <xsl:template match="/">  <xsl:call-template name="altova:Root"/>  </xsl:template>  <xsl:template name="altova:Root">  <html>  <head>  <title/>  <meta name="generator" content="Altova StyleVision Enterprise Edition 2021 (x64) (http://www.altova.com)"/>  <meta http-equiv="X-UA-Compatible" content="IE=9"/>  <style type="text/css">  <xsl:comment>\* { margin:0 auto;  max-width:720px;  }</xsl:comment>  </style>  <xsl:comment>[if IE]&gt;&lt;STYLE type=&quot;text/css&quot;&gt;.altova-rotate-left-textbox{filter: progid:DXImageTransform.Microsoft.BasicImage(rotation=3)} .altova-rotate-right-textbox{filter: progid:DXImageTransform.Microsoft.BasicImage(rotation=1)} &lt;/STYLE&gt;&lt;![endif]</xsl:comment>  <xsl:comment>[if !IE]&gt;&lt;!</xsl:comment>  <style type="text/css">.altova-rotate-left-textbox{-webkit-transform: rotate(-90deg) translate(-100%, 0%); -webkit-transform-origin: 0% 0%;-moz-transform: rotate(-90deg) translate(-100%, 0%); -moz-transform-origin: 0% 0%;-ms-transform: rotate(-90deg) translate(-100%, 0%); -ms-transform-origin: 0% 0%;}.altova-rotate-right-textbox{-webkit-transform: rotate(90deg) translate(0%, -100%); -webkit-transform-origin: 0% 0%;-moz-transform: rotate(90deg) translate(0%, -100%); -moz-transform-origin: 0% 0%;-ms-transform: rotate(90deg) translate(0%, -100%); -ms-transform-origin: 0% 0%;}</style>  <xsl:comment>&lt;![endif]</xsl:comment>  <style type="text/css">@page { margin-left:0.60in; margin-right:0.60in; margin-top:0.79in; margin-bottom:0.79in } @media print { br.altova-page-break { page-break-before: always; } div.altova-page-break { page-break-before: always; } }</style>  </head>  <body>  <xsl:for-each select="$XML">  <br/>  <xsl:variable name="altova:table">  <table border="0" width="100%">  <xsl:variable name="altova:CurrContextGrid\_0" select="."/>  <xsl:variable name="altova:ColumnData"/>  <tbody>  <tr>  <td style="text-align:center; ">  <img style="height:83px; margin-bottom:0px; width:82px; ">  <xsl:attribute name="src">  <xsl:variable name="altova:img\_src" select="altova:reluri-to-content-id( translate(string(&apos;lab\_Images\sv\_ms\_clip0000\_image.jpg&apos;), '&#x5c;', '/') )"/>  <xsl:if test="substring( $altova:img\_src, 2, 1) = ':'">  <xsl:text>file:///</xsl:text>  </xsl:if>  <xsl:value-of select="$altova:img\_src"/>  </xsl:attribute>  <xsl:attribute name="alt"/>  </img>  </td>  </tr>  <tr>  <td style="text-align:center; ">  <span style="font-family:Times New Roman; font-size:12pt; font-weight:bold; ">  <xsl:text>МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ </xsl:text>  </span>  <br/>  <span style="color:#000000; font-family:Times New Roman; font-size:12pt; ">  <xsl:text>федеральное государственное бюджетное образовательное учреждение</xsl:text>  </span>  <br/>  <span style="color:#000000; font-family:Times New Roman; font-size:12pt; ">  <xsl:text>высшего профессионального образования</xsl:text>  </span>  <br/>  <span style="font-family:Times New Roman; font-size:12pt; font-weight:bold; ">  <xsl:text>«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»</xsl:text>  </span>  </td>  </tr>  </tbody>  </table>  </xsl:variable>  <xsl:variable name="altova:col-count" select="sum( for $altova:cell in $altova:table/table/(thead | tbody | tfoot)[ 1 ]/tr[ 1 ]/(th | td) return altova:col-span( $altova:cell ) )" xpath-default-namespace=""/>  <xsl:variable name="altova:TableIndexInfo" select="altova:BuildTableIndexInfo($altova:table)"/>  <xsl:variable name="altova:generate-cols" as="xs:boolean\*" select="for $altova:pos in 1 to $altova:col-count return true()"/>  <xsl:variable name="altova:nodeTableWithoutUnwantedParts">  <xsl:apply-templates select="$altova:table" mode="altova:generate-table">  <xsl:with-param name="altova:generate-cols" select="$altova:generate-cols"/>  <xsl:with-param name="altova:TableIndexInfo" select="$altova:TableIndexInfo"/>  </xsl:apply-templates>  </xsl:variable>  <xsl:sequence select="$altova:nodeTableWithoutUnwantedParts"/>  <p style="letter-spacing:5px; text-align:center; ">  <span style="font-family:Times New Roman; font-size:15pt; font-weight:bold; ">  <xsl:text>ПРИКАЗ</xsl:text>  </span>  </p>  <br/>  <br/>  <xsl:variable name="altova:table">  <table border="0" width="100%">  <xsl:variable name="altova:CurrContextGrid\_1" select="."/>  <xsl:variable name="altova:ColumnData"/>  <tbody>  <tr>  <td style="text-align:left; ">  <span style="font-family:Times New Roman; font-size:14pt; text-decoration:underline; ">  <xsl:text>&#160;&#160; </xsl:text>  </span>  <xsl:for-each select="приказ\_о\_мерах\_по\_об\_исп\_ф\_з">  <xsl:for-each select="содержание">  <xsl:for-each select="дата">  <span style="font-family:Times New Roman; text-decoration:underline; ">  <xsl:variable name="altova:seqContentStrings\_2">  <xsl:value-of select="format-number(number(substring(string(string(.)), 9, 2)), '00', 'format1')"/>  <xsl:variable name="sText" as="xs:string?">  <xsl:text> </xsl:text>  </xsl:variable>  <xsl:value-of select="$sText"/>  <xsl:variable name="sText" as="xs:string?">  <xsl:call-template name="DateToMonthName">  <xsl:with-param name="sDate" select="string(string(.))"/>  <xsl:with-param name="sFormat" select="'long'"/>  </xsl:call-template>  </xsl:variable>  <xsl:value-of select="$sText"/>  <xsl:variable name="sText" as="xs:string?">  <xsl:text> </xsl:text>  </xsl:variable>  <xsl:value-of select="$sText"/>  <xsl:value-of select="format-number(number(substring(string(string(string(.))), 1, 4)), '0000', 'format1')"/>  </xsl:variable>  <xsl:variable name="altova:sContent\_2" select="string($altova:seqContentStrings\_2)"/>  <xsl:value-of select="$altova:sContent\_2"/>  </span>  <span style="font-family:Times New Roman; text-decoration:underline; ">  <xsl:text>г.</xsl:text>  </span>  </xsl:for-each>  </xsl:for-each>  </xsl:for-each>  <span style="font-family:Times New Roman; font-size:14pt; text-decoration:underline; ">  <xsl:text>&#160;&#160; </xsl:text>  </span>  <span style="color:white; font-family:Times New Roman; font-size:14pt; text-decoration:underline; ">  <xsl:text>.</xsl:text>  </span>  <br/>  </td>  <td style="text-align:right; ">  <span style="font-family:Times New Roman; font-size:14pt; ">  <xsl:text>№</xsl:text>  </span>  <span style="font-family:Times New Roman; font-size:14pt; text-decoration:underline; ">  <xsl:text>&#160;&#160;&#160; </xsl:text>  </span>  <xsl:for-each select="приказ\_о\_мерах\_по\_об\_исп\_ф\_з">  <xsl:for-each select="содержание">  <xsl:for-each select="номер">  <span style="font-family:Times New Roman; text-decoration:underline; ">  <xsl:apply-templates/>  </span>  </xsl:for-each>  </xsl:for-each>  </xsl:for-each>  <span style="font-family:Times New Roman; font-size:14pt; text-decoration:underline; ">  <xsl:text>&#160;&#160;&#160;&#160;&#160;&#160;&#160;&#160;&#160;&#160;&#160;&#160;&#160; </xsl:text>  </span>  <span style="color:white; font-family:Times New Roman; font-size:14pt; text-decoration:underline; ">  <xsl:text>.</xsl:text>  </span>  </td>  </tr>  </tbody>  </table>  </xsl:variable>  <xsl:variable name="altova:col-count" select="sum( for $altova:cell in $altova:table/table/(thead | tbody | tfoot)[ 1 ]/tr[ 1 ]/(th | td) return altova:col-span( $altova:cell ) )" xpath-default-namespace=""/>  <xsl:variable name="altova:TableIndexInfo" select="altova:BuildTableIndexInfo($altova:table)"/>  <xsl:variable name="altova:generate-cols" as="xs:boolean\*" select="for $altova:pos in 1 to $altova:col-count return true()"/>  <xsl:variable name="altova:nodeTableWithoutUnwantedParts">  <xsl:apply-templates select="$altova:table" mode="altova:generate-table">  <xsl:with-param name="altova:generate-cols" select="$altova:generate-cols"/>  <xsl:with-param name="altova:TableIndexInfo" select="$altova:TableIndexInfo"/>  </xsl:apply-templates>  </xsl:variable>  <xsl:sequence select="$altova:nodeTableWithoutUnwantedParts"/>  <br/>  <br/>  <xsl:variable name="altova:table">  <table border="0" width="100%">  <xsl:variable name="altova:CurrContextGrid\_3" select="."/>  <xsl:variable name="altova:ColumnData"/>  <tbody>  <tr>  <td style="text-align:left; width:2.80in; ">  <p>  <xsl:for-each select="приказ\_о\_мерах\_по\_об\_исп\_ф\_з">  <xsl:for-each select="содержание">  <xsl:for-each select="тема">  <span style="font-family:Times New Roman; font-weight:bold; ">  <xsl:apply-templates/>  </span>  </xsl:for-each>  </xsl:for-each>  </xsl:for-each>  </p>  </td>  <td/>  </tr>  </tbody>  </table>  </xsl:variable>  <xsl:variable name="altova:col-count" select="sum( for $altova:cell in $altova:table/table/(thead | tbody | tfoot)[ 1 ]/tr[ 1 ]/(th | td) return altova:col-span( $altova:cell ) )" xpath-default-namespace=""/>  <xsl:variable name="altova:TableIndexInfo" select="altova:BuildTableIndexInfo($altova:table)"/>  <xsl:variable name="altova:generate-cols" as="xs:boolean\*" select="for $altova:pos in 1 to $altova:col-count return true()"/>  <xsl:variable name="altova:nodeTableWithoutUnwantedParts">  <xsl:apply-templates select="$altova:table" mode="altova:generate-table">  <xsl:with-param name="altova:generate-cols" select="$altova:generate-cols"/>  <xsl:with-param name="altova:TableIndexInfo" select="$altova:TableIndexInfo"/>  </xsl:apply-templates>  </xsl:variable>  <xsl:sequence select="$altova:nodeTableWithoutUnwantedParts"/>  <br/>  <br/>  <br/>  <p>  <xsl:for-each select="приказ\_о\_мерах\_по\_об\_исп\_ф\_з">  <xsl:for-each select="содержание">  <xsl:for-each select="преамбула">  <span style="font-family:Times New Roman; ">  <xsl:apply-templates/>  </span>  </xsl:for-each>  </xsl:for-each>  </xsl:for-each>  </p>  <br/>  <p style="line-height:15.84px; margin-bottom:10.67px; margin-left:0px; margin-right:0px; margin-top:0px; ">  <span style="font-family:Times New Roman; font-size:14pt; font-weight:bold; ">  <xsl:text>ПРИКАЗЫВАЮ:</xsl:text>  </span>  </p>  <xsl:variable name="altova:table">  <table border="0">  <xsl:variable name="altova:CurrContextGrid\_4" select="."/>  <xsl:variable name="altova:ColumnData"/>  <tbody>  <xsl:for-each select="приказ\_о\_мерах\_по\_об\_исп\_ф\_з">  <xsl:for-each select="меры">  <tr>  <td valign="top">  <xsl:for-each select="номер\_п">  <xsl:apply-templates/>  </xsl:for-each>  </td>  <td valign="top">  <xsl:for-each select="содержание">  <span style="white-space:pre-line; ">  <xsl:apply-templates/>  </span>  </xsl:for-each>  </td>  </tr>  </xsl:for-each>  </xsl:for-each>  </tbody>  </table>  </xsl:variable>  <xsl:variable name="altova:col-count" select="sum( for $altova:cell in $altova:table/table/(thead | tbody | tfoot)[ 1 ]/tr[ 1 ]/(th | td) return altova:col-span( $altova:cell ) )" xpath-default-namespace=""/>  <xsl:variable name="altova:TableIndexInfo" select="altova:BuildTableIndexInfo($altova:table)"/>  <xsl:variable name="altova:generate-cols" as="xs:boolean\*" select="for $altova:pos in 1 to $altova:col-count return true()"/>  <xsl:variable name="altova:nodeTableWithoutUnwantedParts">  <xsl:apply-templates select="$altova:table" mode="altova:generate-table">  <xsl:with-param name="altova:generate-cols" select="$altova:generate-cols"/>  <xsl:with-param name="altova:TableIndexInfo" select="$altova:TableIndexInfo"/>  </xsl:apply-templates>  </xsl:variable>  <xsl:sequence select="$altova:nodeTableWithoutUnwantedParts"/>  <br/>  <br/>  <br/>  <br/>  <br/>  <br/>  <br/>  <xsl:variable name="altova:table">  <table border="0" width="100%">  <xsl:variable name="altova:CurrContextGrid\_5" select="."/>  <xsl:variable name="altova:ColumnData"/>  <tbody>  <tr>  <td style="text-align:left; width:2.27in; ">  <xsl:for-each select="приказ\_о\_мерах\_по\_об\_исп\_ф\_з">  <xsl:for-each select="подпись">  <xsl:for-each select="должность">  <xsl:apply-templates/>  </xsl:for-each>  </xsl:for-each>  </xsl:for-each>  </td>  <td style="text-align:center; width:2.70in; ">  <xsl:for-each select="приказ\_о\_мерах\_по\_об\_исп\_ф\_з">  <xsl:for-each select="подпись">  <xsl:for-each select="подпись">  <xsl:apply-templates/>  </xsl:for-each>  </xsl:for-each>  </xsl:for-each>  </td>  <td style="text-align:right; ">  <xsl:for-each select="приказ\_о\_мерах\_по\_об\_исп\_ф\_з">  <xsl:for-each select="подпись">  <xsl:for-each select="фио">  <xsl:apply-templates/>  </xsl:for-each>  </xsl:for-each>  </xsl:for-each>  </td>  </tr>  </tbody>  </table>  </xsl:variable>  <xsl:variable name="altova:col-count" select="sum( for $altova:cell in $altova:table/table/(thead | tbody | tfoot)[ 1 ]/tr[ 1 ]/(th | td) return altova:col-span( $altova:cell ) )" xpath-default-namespace=""/>  <xsl:variable name="altova:TableIndexInfo" select="altova:BuildTableIndexInfo($altova:table)"/>  <xsl:variable name="altova:generate-cols" as="xs:boolean\*" select="for $altova:pos in 1 to $altova:col-count return true()"/>  <xsl:variable name="altova:nodeTableWithoutUnwantedParts">  <xsl:apply-templates select="$altova:table" mode="altova:generate-table">  <xsl:with-param name="altova:generate-cols" select="$altova:generate-cols"/>  <xsl:with-param name="altova:TableIndexInfo" select="$altova:TableIndexInfo"/>  </xsl:apply-templates>  </xsl:variable>  <xsl:sequence select="$altova:nodeTableWithoutUnwantedParts"/>  <br/>  </xsl:for-each>  </body>  </html>  </xsl:template>  <xsl:template name="DateToMonthName">  <xsl:param name="sDate"/>  <xsl:param name="sFormat"/>  <xsl:choose>  <xsl:when test="$sFormat = 'long'">  <xsl:call-template name="DateToMonthNameLong">  <xsl:with-param name="sDate" select="$sDate"/>  </xsl:call-template>  </xsl:when>  <xsl:otherwise>  <xsl:call-template name="DateToMonthNameShort">  <xsl:with-param name="sDate" select="$sDate"/>  </xsl:call-template>  </xsl:otherwise>  </xsl:choose>  </xsl:template>  <xsl:template name="DateToMonthNameLong">  <xsl:param name="sDate"/>  <xsl:variable name="nMonth">  <xsl:call-template name="DateToMonthNum">  <xsl:with-param name="sDate" select="$sDate"/>  </xsl:call-template>  </xsl:variable>  <xsl:choose>  <xsl:when test="$nMonth = 1">Январь</xsl:when>  <xsl:when test="$nMonth = 2">Февраль</xsl:when>  <xsl:when test="$nMonth = 3">Март</xsl:when>  <xsl:when test="$nMonth = 4">Апрель</xsl:when>  <xsl:when test="$nMonth = 5">Май</xsl:when>  <xsl:when test="$nMonth = 6">Июнь</xsl:when>  <xsl:when test="$nMonth = 7">Июль</xsl:when>  <xsl:when test="$nMonth = 8">Август</xsl:when>  <xsl:when test="$nMonth = 9">Сентябрь</xsl:when>  <xsl:when test="$nMonth = 10">Октябрь</xsl:when>  <xsl:when test="$nMonth = 11">Ноябрь</xsl:when>  <xsl:otherwise>Декабрь</xsl:otherwise>  </xsl:choose>  </xsl:template>  <xsl:template name="DateToMonthNum">  <xsl:param name="sDate"/>  <xsl:choose>  <xsl:when test="starts-with($sDate, '--')">  <xsl:value-of select="number(substring($sDate, 3, 2))"/>  </xsl:when>  <xsl:otherwise>  <xsl:value-of select="number(substring($sDate, 6, 2))"/>  </xsl:otherwise>  </xsl:choose>  </xsl:template>  <xsl:template name="DateToMonthNameShort">  <xsl:param name="sDate"/>  <xsl:variable name="nMonth">  <xsl:call-template name="DateToMonthNum">  <xsl:with-param name="sDate" select="$sDate"/>  </xsl:call-template>  </xsl:variable>  <xsl:choose>  <xsl:when test="$nMonth = 1">янв</xsl:when>  <xsl:when test="$nMonth = 2">фев</xsl:when>  <xsl:when test="$nMonth = 3">мар</xsl:when>  <xsl:when test="$nMonth = 4">апр</xsl:when>  <xsl:when test="$nMonth = 5">май</xsl:when>  <xsl:when test="$nMonth = 6">июн</xsl:when>  <xsl:when test="$nMonth = 7">июл</xsl:when>  <xsl:when test="$nMonth = 8">авг</xsl:when>  <xsl:when test="$nMonth = 9">сен</xsl:when>  <xsl:when test="$nMonth = 10">окт</xsl:when>  <xsl:when test="$nMonth = 11">ноя</xsl:when>  <xsl:otherwise>дек</xsl:otherwise>  </xsl:choose>  </xsl:template>  <xsl:function name="altova:is-cell-empty" as="xs:boolean">  <xsl:param name="altova:cell" as="element()"/>  <xsl:sequence select="altova:is-node-empty( $altova:cell )"/>  </xsl:function>  <xsl:function name="altova:is-node-empty" as="xs:boolean" xpath-default-namespace="">  <xsl:param name="altova:node" as="element()"/>  <xsl:sequence select="every $altova:child in $altova:node/child::node() satisfies ( ( boolean( $altova:child/self::text() ) and string-length( $altova:child ) = 0 ) or ( ( boolean( $altova:child/self::div ) or boolean( $altova:child/self::span ) or boolean( $altova:child/self::a ) ) and altova:is-node-empty( $altova:child ) ) )"/>  </xsl:function>  <xsl:function name="altova:col-span" as="xs:integer">  <xsl:param name="altova:cell" as="element()"/>  <xsl:sequence select="if ( exists( $altova:cell/@colspan ) ) then xs:integer( $altova:cell/@colspan ) else 1"/>  </xsl:function>  <xsl:template match="@\* | node()" mode="altova:generate-table">  <xsl:param name="altova:generate-cols"/>  <xsl:param name="altova:TableIndexInfo"/>  <xsl:copy>  <xsl:apply-templates select="@\* | node()" mode="#current">  <xsl:with-param name="altova:generate-cols" select="$altova:generate-cols"/>  <xsl:with-param name="altova:TableIndexInfo" select="$altova:TableIndexInfo"/>  </xsl:apply-templates>  </xsl:copy>  </xsl:template>  <xsl:template match="tbody" mode="altova:generate-table" xpath-default-namespace="">  <xsl:param name="altova:generate-cols"/>  <xsl:param name="altova:TableIndexInfo"/>  <xsl:choose>  <xsl:when test="empty(tr)">  <xsl:copy>  <tr>  <td/>  </tr>  </xsl:copy>  </xsl:when>  <xsl:otherwise>  <xsl:copy>  <xsl:apply-templates select="@\* | node()" mode="#current">  <xsl:with-param name="altova:generate-cols" select="$altova:generate-cols"/>  <xsl:with-param name="altova:TableIndexInfo" select="$altova:TableIndexInfo"/>  </xsl:apply-templates>  </xsl:copy>  </xsl:otherwise>  </xsl:choose>  </xsl:template>  <xsl:template match="th | td" mode="altova:generate-table" xpath-default-namespace="">  <xsl:choose>  <xsl:when test="altova:is-cell-empty( . )">  <xsl:copy>  <xsl:apply-templates select="@\*" mode="#current"/>  <xsl:text>&#160;</xsl:text>  </xsl:copy>  </xsl:when>  <xsl:otherwise>  <xsl:copy>  <xsl:apply-templates select="@\* | node()" mode="#current"/>  </xsl:copy>  </xsl:otherwise>  </xsl:choose>  </xsl:template>  <xsl:function name="altova:CountLeadingTrueValues">  <xsl:param name="seqBools"/>  <xsl:sequence select="if (not($seqBools[1] = true())) then 0 else 1 + altova:CountLeadingTrueValues($seqBools[position() gt 1])"/>  </xsl:function>  <xsl:function name="altova:GetCellFromRow" as="node()?" xpath-default-namespace="">  <xsl:param name="nodeTableRow" as="node()"/>  <xsl:param name="nCell" as="xs:integer"/>  <xsl:sequence select="$nodeTableRow/(th | td)[$nCell]"/>  </xsl:function>  <xsl:function name="altova:GetCellsFromRow" as="node()\*" xpath-default-namespace="">  <xsl:param name="nodeTableRow" as="node()"/>  <xsl:sequence select="$nodeTableRow/(th | td)"/>  </xsl:function>  <xsl:function name="altova:GetRowsFromTable" as="node()\*" xpath-default-namespace="">  <xsl:param name="nodeTable" as="node()"/>  <xsl:sequence select="$nodeTable/(thead | tbody | tfoot)/tr"/>  </xsl:function>  <xsl:function name="altova:GetRowspanFromCell" as="xs:integer" xpath-default-namespace="">  <xsl:param name="altova:nodeCell" as="node()"/>  <xsl:variable name="altova:sRowSpan" select="$altova:nodeCell/@rowspan" as="xs:string?"/>  <xsl:variable name="altova:nRowSpan" select="if ($altova:sRowSpan) then xs:integer($altova:sRowSpan) else 1" as="xs:integer"/>  <xsl:sequence select="$altova:nRowSpan"/>  </xsl:function>  <!-- In HTML and FO, the table section order is header, footer, body. This function determines, for a given cell, the number its row would have if the section order were header, body, footer -->  <xsl:function name="altova:GetGridRowNumForCell" xpath-default-namespace="">  <xsl:param name="altova:nodeTableCell" as="node()"/>  <xsl:variable name="altova:nodeTableRow" select="$altova:nodeTableCell/.." as="node()"/>  <xsl:variable name="altova:nodeTableSection" select="$altova:nodeTableRow/.." as="node()"/>  <xsl:variable name="altova:sTableSection" select="fn:local-name($altova:nodeTableSection)" as="xs:string"/>  <xsl:variable name="altova:nodeTable" select="$altova:nodeTableSection/.." as="node()"/>  <xsl:variable name="altova:nRowNumInSection" select="count($altova:nodeTableRow/preceding-sibling::tr) + 1" as="xs:integer"/>  <xsl:choose>  <xsl:when test="$altova:sTableSection eq 'tbody'">  <xsl:sequence select="count($altova:nodeTable/thead/tr) + $altova:nRowNumInSection"/>  </xsl:when>  <xsl:when test="$altova:sTableSection eq 'thead'">  <xsl:sequence select="$altova:nRowNumInSection"/>  </xsl:when>  <xsl:when test="$altova:sTableSection eq 'tfoot'">  <xsl:sequence select="count($altova:nodeTable/thead/tr) + count($altova:nodeTable/tbody/tr) + $altova:nRowNumInSection"/>  </xsl:when>  <xsl:otherwise>  <xsl:message select="'Internal Error'" terminate="yes"/>  </xsl:otherwise>  </xsl:choose>  </xsl:function>  <xsl:function name="altova:BuildTableIndexInfo" as="node()" xpath-default-namespace="">  <xsl:param name="altova:nodeTable" as="node()"/>  <altova:ColumnIndices>  <xsl:variable name="altova:nodeAllRows" select="altova:GetRowsFromTable($altova:nodeTable/table)" as="node()\*"/>  <xsl:if test="$altova:nodeAllRows">  <xsl:variable name="altova:seqCellsInFirstRow" select="altova:GetCellsFromRow($altova:nodeAllRows[1])" as="node()\*"/>  <xsl:variable name="altova:nColumnsInTable" select="sum(for $nodeCell in $altova:seqCellsInFirstRow return altova:col-span($nodeCell))" as="xs:integer"/>  <xsl:variable name="altova:seqActiveRowSpans" select="for $Cell in 1 to $altova:nColumnsInTable return 0" as="xs:integer\*"/>  <!--xsl:sequence select="altova:BuildTableIndexInfo\_Recursive($altova:nodeAllRows, 1, $altova:seqActiveRowSpans)"/-->  <xsl:call-template name="altova:BuildTableIndexInfo\_Recursive">  <xsl:with-param name="altova:nodeTableRows" select="$altova:nodeAllRows"/>  <xsl:with-param name="altova:nRow" select="1"/>  <xsl:with-param name="altova:seqActiveRowSpans" select="$altova:seqActiveRowSpans"/>  </xsl:call-template>  </xsl:if>  </altova:ColumnIndices>  </xsl:function>  <!--xsl:template name="altova:BuildTableIndexInfo\_Recursive" as="node()\*" xpath-default-namespace=""-->  <xsl:template name="altova:BuildTableIndexInfo\_Recursive" xpath-default-namespace="">  <xsl:param name="altova:nodeTableRows" as="node()\*"/>  <xsl:param name="altova:nRow" as="xs:integer"/>  <xsl:param name="altova:seqActiveRowSpans" as="xs:integer\*"/>  <xsl:variable name="altova:nodeRow" select="$altova:nodeTableRows[$altova:nRow]" as="node()?"/>  <xsl:choose>  <xsl:when test="empty($altova:nodeRow)">  <xsl:sequence select="()"/>  </xsl:when>  <xsl:when test="empty(altova:GetCellFromRow($altova:nodeRow, 1))">  <xsl:sequence select="()"/>  </xsl:when>  <xsl:otherwise>  <xsl:variable name="altova:nodeCell1" select="altova:GetCellFromRow($altova:nodeRow, 1)" as="node()?"/>  <xsl:variable name="altova:nColSpan" select="altova:col-span($altova:nodeCell1)" as="xs:integer"/>  <altova:Row>  <!--xsl:variable name="altova:seqColumnIndicesOfCurrentRow" select="altova:BuildTableIndexInfo\_SingleRow\_Recursive($altova:seqActiveRowSpans, $altova:nodeRow, 1, 1, $altova:nColSpan, true(), 0, 0)" as="node()\*"/>  <xsl:sequence select="$altova:seqColumnIndicesOfCurrentRow"/-->  <xsl:call-template name="altova:BuildTableIndexInfo\_SingleRow\_Recursive">  <xsl:with-param name="altova:seqActiveRowSpans" select="$altova:seqActiveRowSpans"/>  <xsl:with-param name="altova:nodeRow" select="$altova:nodeRow"/>  <xsl:with-param name="altova:nColumn" select="1"/>  <xsl:with-param name="altova:nCellInCurrentRow" select="1"/>  <xsl:with-param name="altova:nColSpanInCellRemaining" select="$altova:nColSpan"/>  <xsl:with-param name="altova:bColSpanBegins" select="true()"/>  <xsl:with-param name="altova:nCurrentSum" select="0"/>  <xsl:with-param name="altova:nRowSpansToAdd" select="0"/>  </xsl:call-template>  </altova:Row>  <xsl:variable name="altova:seqActiveRowSpans\_New" select="altova:BuildTableIndexInfo\_ActiveRowSpans\_Recursive($altova:seqActiveRowSpans, $altova:nodeRow, 1, 1, $altova:nColSpan)" as="xs:integer\*"/>  <!--xsl:sequence select="altova:BuildTableIndexInfo\_Recursive($altova:nodeTableRows, $altova:nRow + 1, $altova:seqActiveRowSpans\_New)"/-->  <xsl:call-template name="altova:BuildTableIndexInfo\_Recursive">  <xsl:with-param name="altova:nodeTableRows" select="$altova:nodeTableRows"/>  <xsl:with-param name="altova:nRow" select="$altova:nRow + 1"/>  <xsl:with-param name="altova:seqActiveRowSpans" select="$altova:seqActiveRowSpans\_New"/>  </xsl:call-template>  </xsl:otherwise>  </xsl:choose>  </xsl:template>  <!--xsl:template name="altova:BuildTableIndexInfo\_SingleRow\_Recursive" as="node()\*" xpath-default-namespace=""-->  <xsl:template name="altova:BuildTableIndexInfo\_SingleRow\_Recursive" xpath-default-namespace="">  <xsl:param name="altova:seqActiveRowSpans" as="xs:integer\*"/>  <xsl:param name="altova:nodeRow" as="node()"/>  <xsl:param name="altova:nColumn" as="xs:integer"/>  <xsl:param name="altova:nCellInCurrentRow" as="xs:integer"/>  <xsl:param name="altova:nColSpanInCellRemaining" as="xs:integer"/>  <xsl:param name="altova:bColSpanBegins" as="xs:boolean"/>  <!-- Also true if it's just a single cell -->  <xsl:param name="altova:nCurrentSum" as="xs:integer"/>  <xsl:param name="altova:nRowSpansToAdd" as="xs:integer"/>  <xsl:choose>  <xsl:when test="$altova:nColumn gt count($altova:seqActiveRowSpans)">  <xsl:sequence select="()"/>  </xsl:when>  <!-- If the cell is not under a rowspan -->  <xsl:when test="$altova:seqActiveRowSpans[$altova:nColumn] eq 0">  <!-- If the cell is not under a colspan -->  <xsl:if test="$altova:bColSpanBegins eq true()">  <!-- A non-spanned cell starts in this column -->  <altova:ColumnIndex>  <xsl:sequence select="$altova:nCurrentSum + 1"/>  </altova:ColumnIndex>  </xsl:if>  <xsl:variable name="altova:nCellInCurrentRow\_New" select="if ($altova:nColSpanInCellRemaining gt 1) then $altova:nCellInCurrentRow else $altova:nCellInCurrentRow + 1" as="xs:integer"/>  <xsl:variable name="altova:nColSpanInCell\_New" select="if ($altova:nColSpanInCellRemaining gt 1) then $altova:nColSpanInCellRemaining - 1 else if (empty(altova:GetCellFromRow($altova:nodeRow, $altova:nCellInCurrentRow\_New))) then 1 else altova:col-span(altova:GetCellFromRow($altova:nodeRow, $altova:nCellInCurrentRow\_New))" as="xs:integer"/>  <xsl:variable name="altova:bColSpanBegins\_New" select="$altova:nCellInCurrentRow ne $altova:nCellInCurrentRow\_New" as="xs:boolean"/>  <!--xsl:sequence select="altova:BuildTableIndexInfo\_SingleRow\_Recursive($altova:seqActiveRowSpans, $altova:nodeRow, $altova:nColumn + 1, $altova:nCellInCurrentRow\_New, $altova:nColSpanInCell\_New, $altova:bColSpanBegins\_New, $altova:nCurrentSum + 1, $altova:nRowSpansToAdd)"/-->  <xsl:call-template name="altova:BuildTableIndexInfo\_SingleRow\_Recursive">  <xsl:with-param name="altova:seqActiveRowSpans" select="$altova:seqActiveRowSpans"/>  <xsl:with-param name="altova:nodeRow" select="$altova:nodeRow"/>  <xsl:with-param name="altova:nColumn" select="$altova:nColumn + 1"/>  <xsl:with-param name="altova:nCellInCurrentRow" select="$altova:nCellInCurrentRow\_New"/>  <xsl:with-param name="altova:nColSpanInCellRemaining" select="$altova:nColSpanInCell\_New"/>  <xsl:with-param name="altova:bColSpanBegins" select="$altova:bColSpanBegins\_New"/>  <xsl:with-param name="altova:nCurrentSum" select="$altova:nCurrentSum + 1"/>  <xsl:with-param name="altova:nRowSpansToAdd" select="$altova:nRowSpansToAdd"/>  </xsl:call-template>  </xsl:when>  <!-- The cell is under a rowspan -->  <xsl:otherwise>  <!--xsl:sequence select="altova:BuildTableIndexInfo\_SingleRow\_Recursive($altova:seqActiveRowSpans, $altova:nodeRow, $altova:nColumn + 1, $altova:nCellInCurrentRow, $altova:nColSpanInCellRemaining, $altova:bColSpanBegins, $altova:nCurrentSum + 1, $altova:nRowSpansToAdd + 1)"/-->  <xsl:call-template name="altova:BuildTableIndexInfo\_SingleRow\_Recursive">  <xsl:with-param name="altova:seqActiveRowSpans" select="$altova:seqActiveRowSpans"/>  <xsl:with-param name="altova:nodeRow" select="$altova:nodeRow"/>  <xsl:with-param name="altova:nColumn" select="$altova:nColumn + 1"/>  <xsl:with-param name="altova:nCellInCurrentRow" select="$altova:nCellInCurrentRow"/>  <xsl:with-param name="altova:nColSpanInCellRemaining" select="$altova:nColSpanInCellRemaining"/>  <xsl:with-param name="altova:bColSpanBegins" select="$altova:bColSpanBegins"/>  <xsl:with-param name="altova:nCurrentSum" select="$altova:nCurrentSum + 1"/>  <xsl:with-param name="altova:nRowSpansToAdd" select="$altova:nRowSpansToAdd + 1"/>  </xsl:call-template>  </xsl:otherwise>  </xsl:choose>  </xsl:template>  <xsl:function name="altova:BuildTableIndexInfo\_ActiveRowSpans\_Recursive" as="xs:integer\*" xpath-default-namespace="">  <xsl:param name="altova:seqRowSpans" as="xs:integer\*"/>  <xsl:param name="altova:nodeCurrentRow" as="node()"/>  <xsl:param name="altova:nColumn" as="xs:integer"/>  <xsl:param name="altova:nCellInCurrentRow" as="xs:integer"/>  <xsl:param name="altova:nColSpanInCellRemaining" as="xs:integer"/>  <xsl:choose>  <xsl:when test="$altova:nColumn gt count($altova:seqRowSpans)">  <xsl:sequence select="()"/>  </xsl:when>  <xsl:otherwise>  <xsl:variable name="altova:nRowSpanForCurrentColumn" select="altova:BuildTableIndexInfo\_ActiveRowSpans\_SingleColumn($altova:seqRowSpans, $altova:nodeCurrentRow, $altova:nColumn, $altova:nCellInCurrentRow)" as="xs:integer?"/>  <xsl:sequence select="$altova:nRowSpanForCurrentColumn"/>  <xsl:variable name="altova:nCellInCurrentRow\_New" select="if ($altova:seqRowSpans[$altova:nColumn] eq 0 and $altova:nColSpanInCellRemaining eq 1) then $altova:nCellInCurrentRow + 1 else $altova:nCellInCurrentRow" as="xs:integer"/>  <xsl:variable name="altova:nColSpanInCell\_New" select="if ($altova:seqRowSpans[$altova:nColumn] eq 0 and $altova:nCellInCurrentRow eq $altova:nCellInCurrentRow\_New) then $altova:nColSpanInCellRemaining - 1 else if (empty(altova:GetCellFromRow($altova:nodeCurrentRow, $altova:nCellInCurrentRow\_New))) then 1 else altova:col-span(altova:GetCellFromRow($altova:nodeCurrentRow, $altova:nCellInCurrentRow\_New))" as="xs:integer"/>  <xsl:sequence select="altova:BuildTableIndexInfo\_ActiveRowSpans\_Recursive($altova:seqRowSpans, $altova:nodeCurrentRow, $altova:nColumn + 1, $altova:nCellInCurrentRow\_New, $altova:nColSpanInCell\_New)"/>  </xsl:otherwise>  </xsl:choose>  </xsl:function>  <xsl:function name="altova:BuildTableIndexInfo\_ActiveRowSpans\_SingleColumn" as="xs:integer?" xpath-default-namespace="">  <xsl:param name="altova:seqRowSpans" as="xs:integer\*"/>  <xsl:param name="altova:nodeCurrentRow" as="node()"/>  <xsl:param name="altova:nColumn" as="xs:integer"/>  <xsl:param name="altova:nCellInCurrentRow" as="xs:integer"/>  <xsl:choose>  <xsl:when test="$altova:seqRowSpans[$altova:nColumn] gt 0">  <xsl:sequence select="$altova:seqRowSpans[$altova:nColumn] - 1"/>  </xsl:when>  <xsl:otherwise>  <xsl:variable name="altova:nodeCell" select="altova:GetCellFromRow($altova:nodeCurrentRow, $altova:nCellInCurrentRow)" as="node()"/>  <xsl:sequence select="altova:GetRowspanFromCell($altova:nodeCell) - 1"/>  </xsl:otherwise>  </xsl:choose>  </xsl:function>  <xsl:function name="altova:reluri-to-content-id" as="xs:string">  <xsl:param name="input"/>  <xsl:choose>  <xsl:when test="$altova:bGeneratingMime and not(contains($input, '://') or starts-with($input, '/') or contains($input, ':/') or contains($input, ':\'))">  <xsl:sequence select="string-join(('cid:', encode-for-uri($input), '@', $altova:execution-guid, '.Altova.StyleVision'), '')"/>  </xsl:when>  <xsl:otherwise>  <xsl:sequence select="$input"/>  </xsl:otherwise>  </xsl:choose>  </xsl:function>  <xsl:function name="altova:GetChartYValuesForSingleSeries">  <xsl:param name="seqCategoryLeafPos" as="node()\*"/>  <xsl:param name="nodeSeriesLeafPos" as="node()"/>  <xsl:param name="bValuesInCategory" as="xs:boolean"/>  <xsl:for-each select="$seqCategoryLeafPos">  <xsl:element name="altova:Value">  <xsl:value-of select="altova:GetChartYValueForSingleSeriesPos($nodeSeriesLeafPos, ., $bValuesInCategory)"/>  </xsl:element>  </xsl:for-each>  </xsl:function>  <xsl:function name="altova:GetChartYValueForSingleSeriesPos">  <xsl:param name="nodeSeriesLeafPos" as="node()"/>  <xsl:param name="nodeCategoryLeafPos" as="node()"/>  <xsl:param name="bValuesInCategory" as="xs:boolean"/>  <xsl:variable name="altova:seqCategoryContextIds" select="$nodeCategoryLeafPos/altova:Context/@altova:ContextId" as="xs:string\*"/>  <xsl:variable name="altova:seqSeriesContextIds" select="$nodeSeriesLeafPos/altova:Context/@altova:ContextId" as="xs:string\*"/>  <xsl:variable name="altova:sCommonContextId" select="for $i in $altova:seqCategoryContextIds return if (some $j in $altova:seqSeriesContextIds satisfies $i eq $j) then $i else ()" as="xs:string\*"/>  <xsl:choose>  <xsl:when test="count($altova:sCommonContextId) gt 1">  <xsl:message select="concat('Found several values instead of a single one (contexts: ', string-join($altova:sCommonContextId, ', '), ').')" terminate="yes"/>  </xsl:when>  <xsl:when test="count($altova:sCommonContextId) lt 1">  <xsl:message select="concat('XBRL Chart: Info: No value found for position labeled &quot;', $nodeCategoryLeafPos/@altova:sLabel, '&quot;')" terminate="no"/>  <xsl:sequence select="'altova:no-value'"/>  </xsl:when>  <xsl:when test="$bValuesInCategory">  <xsl:sequence select="xs:string($nodeCategoryLeafPos/altova:Context[@altova:ContextId eq $altova:sCommonContextId]/@altova:Value)"/>  </xsl:when>  <xsl:otherwise>  <xsl:sequence select="xs:string($nodeSeriesLeafPos/altova:Context[@altova:ContextId eq $altova:sCommonContextId]/@altova:Value)"/>  </xsl:otherwise>  </xsl:choose>  </xsl:function>  <xsl:function name="altova:GetChartLabelForPos" as="xs:string">  <xsl:param name="nodeParam" as="node()"/>  <xsl:value-of select="string-join($nodeParam/ancestor-or-self::altova:Pos/@altova:sLabel, ' ')"/>  </xsl:function>  <xsl:function name="altova:convert-length-to-pixel" as="xs:decimal">  <xsl:param name="altova:length"/>  <xsl:variable name="normLength" select="normalize-space($altova:length)"/>  <xsl:choose>  <xsl:when test="ends-with($normLength, 'px')">  <xsl:value-of select="substring-before($normLength, 'px')"/>  </xsl:when>  <xsl:when test="ends-with($normLength, 'in')">  <xsl:value-of select="xs:decimal(substring-before($normLength, 'in')) \* $altova:nPxPerIn"/>  </xsl:when>  <xsl:when test="ends-with($normLength, 'cm')">  <xsl:value-of select="xs:decimal(substring-before($normLength, 'cm')) \* $altova:nPxPerIn div 2.54"/>  </xsl:when>  <xsl:when test="ends-with($normLength, 'mm')">  <xsl:value-of select="xs:decimal(substring-before($normLength, 'mm')) \* $altova:nPxPerIn div 25.4"/>  </xsl:when>  <xsl:when test="ends-with($normLength, 'pt')">  <xsl:value-of select="xs:decimal(substring-before($normLength, 'pt')) \* $altova:nPxPerIn div 72.0"/>  </xsl:when>  <xsl:when test="ends-with($normLength, 'pc')">  <xsl:value-of select="xs:decimal(substring-before($normLength, 'pc')) \* $altova:nPxPerIn div 6.0"/>  </xsl:when>  <xsl:otherwise>  <xsl:value-of select="$normLength"/>  </xsl:otherwise>  </xsl:choose>  </xsl:function>  <xsl:function name="altova:convert-length-to-mm" as="xs:decimal">  <xsl:param name="altova:length"/>  <xsl:variable name="normLength" select="normalize-space($altova:length)"/>  <xsl:choose>  <xsl:when test="ends-with($normLength, 'px')">  <xsl:value-of select="xs:decimal(substring-before($normLength, 'px')) div $altova:nPxPerIn \* 25.4"/>  </xsl:when>  <xsl:when test="ends-with($normLength, 'in')">  <xsl:value-of select="xs:decimal(substring-before($normLength, 'in')) \* 25.4"/>  </xsl:when>  <xsl:when test="ends-with($normLength, 'cm')">  <xsl:value-of select="xs:decimal(substring-before($normLength, 'cm')) \* 10"/>  </xsl:when>  <xsl:when test="ends-with($normLength, 'mm')">  <xsl:value-of select="substring-before($normLength, 'mm') "/>  </xsl:when>  <xsl:when test="ends-with($normLength, 'pt')">  <xsl:value-of select="xs:decimal(substring-before($normLength, 'pt')) \* 25.4 div 72.0"/>  </xsl:when>  <xsl:when test="ends-with($normLength, 'pc')">  <xsl:value-of select="xs:decimal(substring-before($normLength, 'pc')) \* 25.4 div 6.0"/>  </xsl:when>  <xsl:otherwise>  <xsl:value-of select="number($normLength) div $altova:nPxPerIn \* 25.4"/>  </xsl:otherwise>  </xsl:choose>  </xsl:function>  </xsl:stylesheet> |