Министерство Цифрового Развития РФ

ФГБОУ ВО «СибГУТИ»

Кафедра вычислительных систем

Расчётно-графическое задание

Вариант №3

Выполнил студент 1 курса группы ИА-232:

Русецкий Артём Сергеевич

Проверил преподаватель:

Авдеев Кирилл Игоревич

Новосибирск, 2022

Содержание

Цели и задачи………….………….……………………………………3

Краткая теория……..………….…………….…………………………3

Описание работы………………….……………………………….…..5

Результат………………………….……….……………………………6

Листинг………………………….…………………………………….11

Цели и задачи

1. Изучить XSLT язык преобразования XML.
2. Преобразовать файл rgz.xml в html страницу используя XSLT – язык преобразования XML-документов.

Краткая теория

**XSLT** *(eXtensible Stylesheet Language Transformations)* — язык преобразования XML-документов. Спецификация XSLT входит в состав XSL и является рекомендацией W3C.  
XSLT имеет множество различных применений, в основном в области веб-программирования и генерации отчётов. Одной из задач, решаемых языком XSLT, является отделение данных от их представления, как часть общей парадигмы MVC (англ. Model-view-controller). Другой стандартной задачей является преобразование XML-документов из одной XML-схемы в другую.

**XML** *(eXtensible Markup Language)* — «расширяемый язык разметки». Спецификация XML описывает XML-документы и частично описывает поведение XML-процессоров (программ, читающих XML-документы и обеспечивающих доступ к их содержимому). XML разрабатывался как язык с простым формальным синтаксисом, удобный для создания и обработки документов как программами, так и человеком, с акцентом на использование в Интернете. Язык называется расширяемым, поскольку он не фиксирует разметку, используемую в документах: разработчик волен создать разметку в соответствии с потребностями к конкретной области, будучи ограниченным лишь синтаксическими правилами языка.

**HTML** *(HyperText Markup Language)* — «язык гипертекстовой разметки» стандартизированный язык гипертекстовой разметки документов для просмотра веб-страниц в браузере. Веб-браузеры получают HTML документ от сервера по протоколам HTTP/HTTPS или открывают с локального диска, далее интерпретируют код в интерфейс, который будет отображаться на экране монитора.

XSL-шаблоны

Задаются элементами xsl:template

Каждый шаблон содержит:

* Адрес узлов (на XPath), на которые действует данный шаблон, записывается в атрибут match элемента xsl:template
* Инструкции по преобразованию Пример: инструкции по преобразованию

Элемент xsl:value-of

* Элемент xsl:value-of вычисляет некоторое значение (обычно, хотя и не всегда, на основе данных исходного документа) и копирует полученный результат в выходной документ. Атрибут select элемента xsl:value-of определяет, какое именно значение вычисляется.

Обработка множественных элементов с помощью xsl:for-each

Элемент xsl:value-of следует использовать только в тех случаях, когда очевидно, значение какого именно узла берется. Если может быть выбрано множество пунктов, тогда будет выбран лишь первый из них.

Существует два способа по очереди обработать множественные элементы:

* xsl:apply-templates с атрибутом select
* использование xsl:for-each

Элемент xsl:for-each обрабатывает по очереди каждый элемент, выбранный атрибутом select. Дополнительные шаблоны в данном случае не требуются.

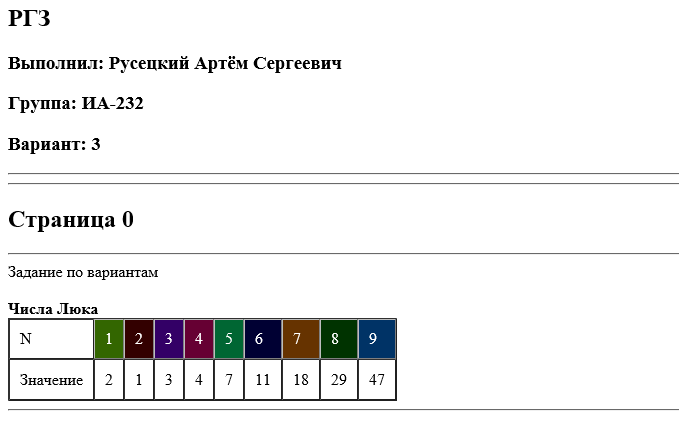
Сортировка выходных элементов

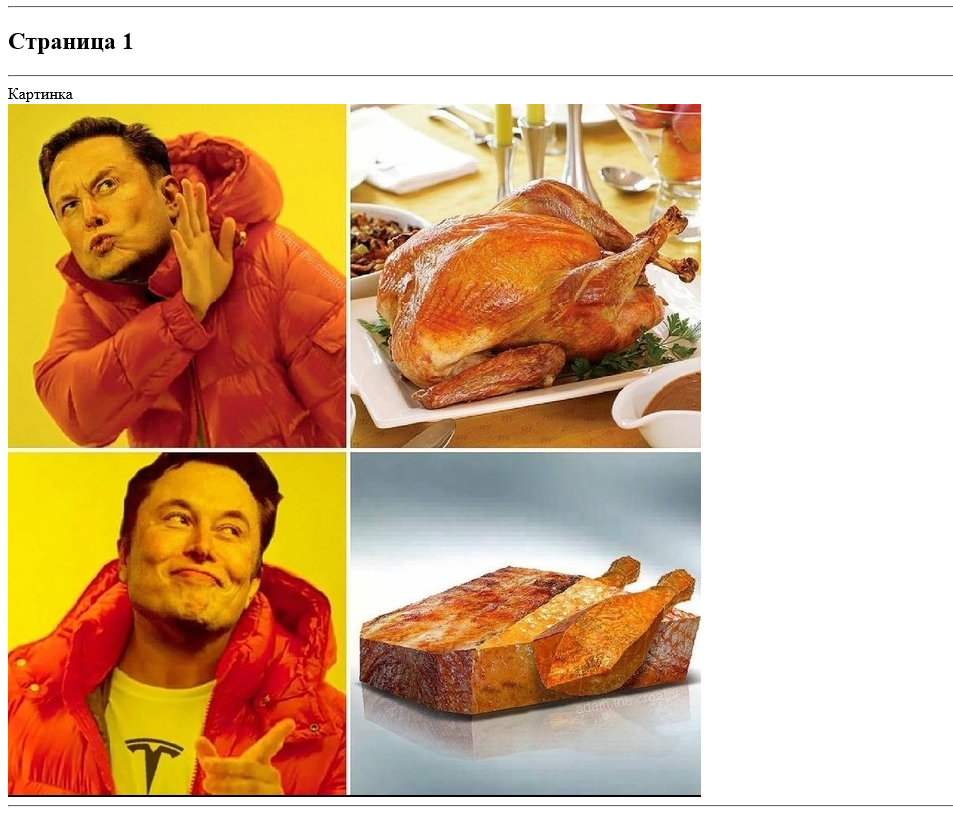
* Элемент xsl:sort сортирует выходные элементы.
* Элемент xsl:sort должен быть дочерним элементом элементов xsl:apply-templates или xsl:for-each.
* Атрибут select элемента xsl:sort задает ключ, используемый для сортировки элементов.

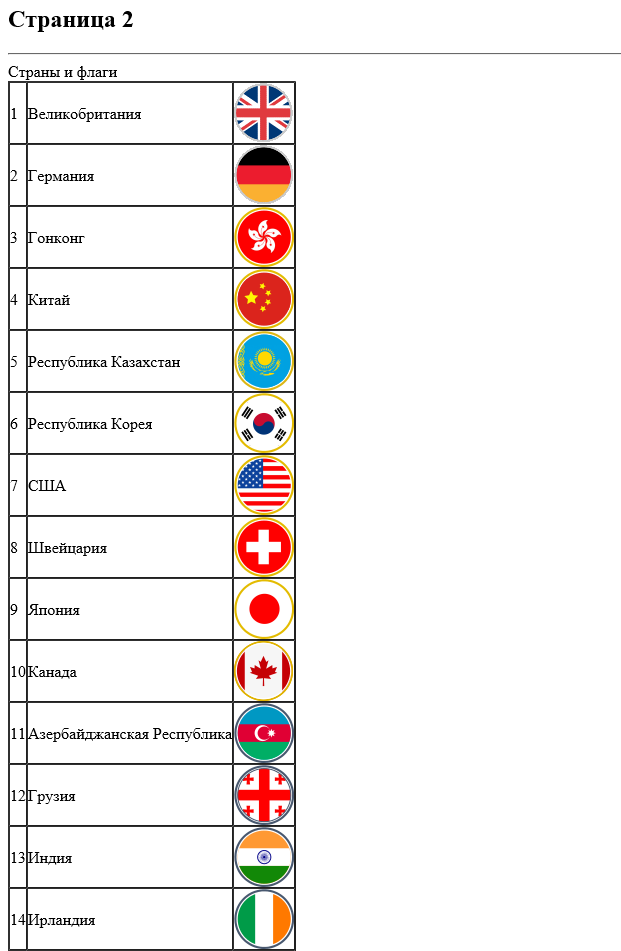
Описание работы

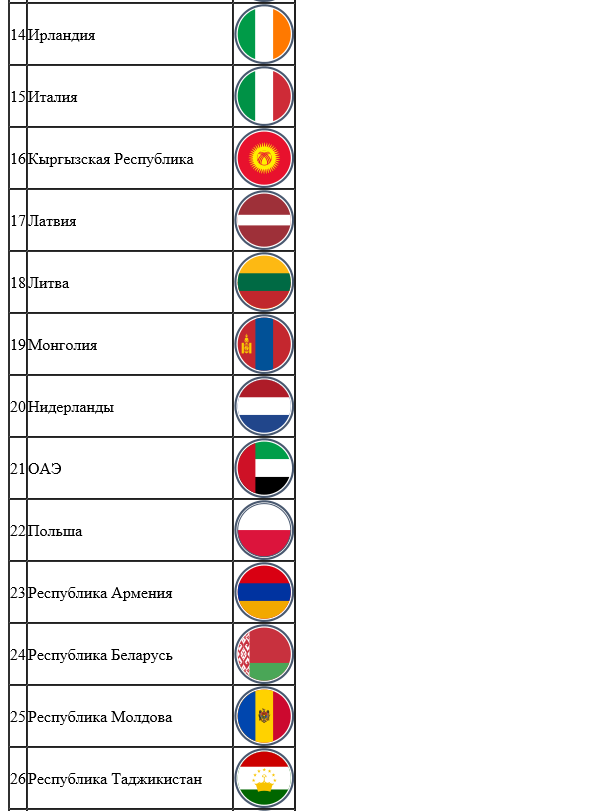
1. Чтобы создать заголовок со словом «Страница», используем xsl:template match=”//page”, затем используем xsl:value-of значение @num (0, 1, 2, 3).
2. Для страницы 0 нужно создать таблицу в соответствии с вариантом. Создаём таблицу и, используя цикл xsl:for-each, берем каждый элемент содержащийся в //sequence[@variant=$variant]/items/item и записываем значение из num в созданную ячейку, это будет N – номер элемента. Те же действия проделываем для //sequence[@variant=$variant]/items/item, при этом записывая их в новую нижнюю строку таблицы из val, это будет значение N в соответствии с номером.
3. Для страницы 1 создаём таблицу без границ. Объявляем переменную rowSize с ссылкой на @row-size (ширина общей “картины”). Далее используя xsl:for-each и xsl:sort мы определяем номера первых картинок в строке. Затем находим оставшиеся картинки для каждой строки в порядке убывания. Размещаем наши изображения, хранящиеся по ссылке в table/item/url с помощью img.
4. Для страницы 2 создаём таблицу. Для этого каждую страну в countries/country сортируем по id (с помощью xsl:for-each и xsl:sort) в порядке возрастания и заносим id, uf\_name и uf\_icon в ячейки строки таблицы.
5. Для страницы 3 создаем цикл, состоящий из таблиц. Таблицы создаются для каждого предмета subjects/subject из студентов и их групп, чьи subjects/subject соответствуют id текущего subject. Заполняем ячейки таблицы при помощи concat(сложение) и substring(взятие определённого отрывка из строки). Затем перебираем студентов, у которых student[subjects[not(subject)]] и заносим их в таблицу студентов «не участвующих студентов». Повторяем все те же действия, описанные выше для заполнения таблицы.

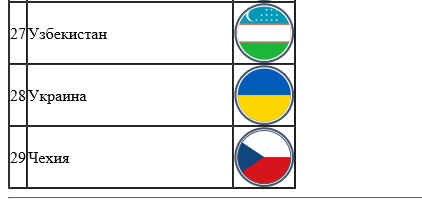
Результат

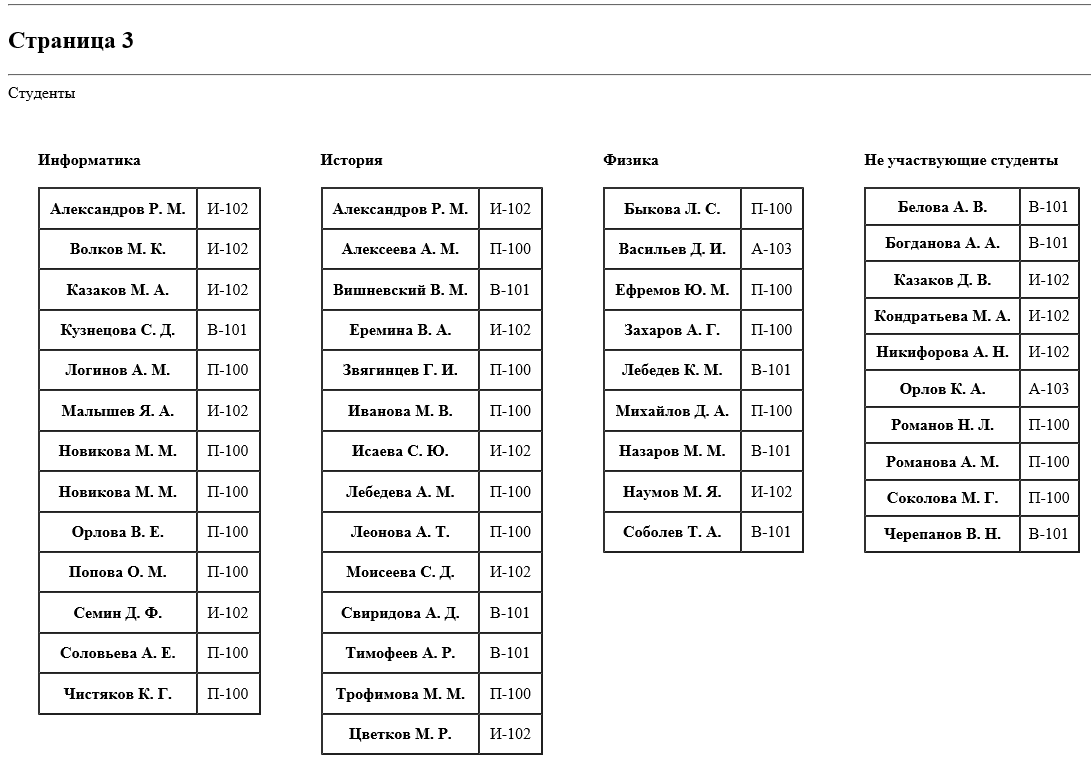












Листинг

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

<xsl:template match="/">

<html>

    <head>

        <title>РГЗ</title>

        <style>

            .fl-t {

                float: left;

                padding: 15px;

                margin: 15px;

            }

        </style>

    </head>

    <body>

        <h2>РГЗ</h2>

        <h3>Выполнил: Русецкий Артём Сергеевич</h3>

        <h3>Группа: ИА-232</h3>

        <h3>Вариант: 3</h3>

        <hr/>

    <xsl:apply-templates />

    </body>

</html>

</xsl:template>

<xsl:template match="//page">

    <hr/>

    <h2>Страница <xsl:value-of select="@num" /></h2>

    <hr/>

    <xsl:apply-templates />

</xsl:template>

<xsl:template match="//page[@num=0]/variants">

    <xsl:variable name="variant" select="3" />

    <br/><br/>

    <b><xsl:value-of select="sequence[@variant=$variant]/name" /></b>

    <table border="1" cellspacing="0" cellpadding="10">

        <tr>

            <td>N</td>

            <xsl:for-each select="//sequence[@variant=$variant]/items/item">

                <td bgcolor="{num/@color}" style="color: white;">

                    <xsl:value-of select="num" />

                </td>

            </xsl:for-each>

        </tr>

        <tr>

            <td>Значение</td>

            <xsl:for-each select="//sequence[@variant=$variant]/items/item">

                <td>

                    <xsl:value-of select="val" />

                </td>

            </xsl:for-each>

        </tr>

    </table>

</xsl:template>

<xsl:template match="//page[@num=1]/table">

    <table cellspacing="0" cellpadding="0">

        <xsl:variable name="rowSize" select="@row-size" />

        <xsl:for-each select="//item[order mod $rowSize = 0]">

            <xsl:sort

                select="order" data-type="number" order="descending"

            />

            <tr>

                <xsl:variable name="rowPosition" select="order" />

                <xsl:for-each select="//item[order &lt;= $rowPosition and order &gt; $rowPosition - $rowSize]">

                    <xsl:sort

                        select="order" data-type="number" order="descending"

                    />

                    <td>

                        <img src="{url}"></img>

                    </td>

                </xsl:for-each>

            </tr>

        </xsl:for-each>

    </table>

</xsl:template>

<xsl:template match="//page[@num=2]/countries">

    <table border="1" cellspacing="0" cellpadding="0">

        <xsl:for-each select="//country">

            <xsl:sort

                select="id" data-type="number" order="ascending"

            />

            <tr>

                <td>

                    <xsl:value-of select="id" />

                </td>

                <td>

                    <xsl:value-of select="uf\_name" />

                </td>

                <td>

                    <img src="{uf\_icon}"></img>

                </td>

            </tr>

        </xsl:for-each>

    </table>

</xsl:template>

<xsl:template match="//page[@num=3]/students">

        <br/><br/>

        <xsl:for-each select="//page[@num=3]/subjects/subject">

            <xsl:variable name="subjectID" select="@id" />

            <xsl:variable name="subjectName" select="name" />

            <div class="fl-t">

                <b><xsl:value-of select="$subjectName" /></b>

                <br/><br/>

                <table border="1" cellspacing="0" cellpadding="10">

                    <xsl:for-each select="//page[@num=3]/students/student[subjects[subject[text()=$subjectID]]]">

                        <xsl:sort select="lname" data-type="text" order="ascending"/>

                        <xsl:variable name="studentGroupID" select="group" />

                        <tr>

                            <th>

                                <xsl:value-of select="concat(lname, ' ')"/>

                                <xsl:value-of select="substring(fname ,1 ,1)"/>.

                                <xsl:value-of select="substring(sname ,1 ,1)"/>.

                            </th>

                            <td>

                                <xsl:value-of select="//page[@num=3]/groups/group[@id=$studentGroupID]/name" />

                            </td>

                        </tr>

                    </xsl:for-each>

                </table>

            </div>

        </xsl:for-each>

        <div class="fl-t">

            <b>Не участвующие студенты</b>

            <br/><br/>

            <table border="1" cellspacing="0" cellpadding="8">

                <xsl:for-each select="//page[@num=3]/students/student[subjects[not(subject)]]">

                    <xsl:sort

                        select="lname" data-type="text" order="ascending"

                    />

                    <xsl:variable name="studentGroupID" select="group" />

                    <tr>

                        <th>

                            <xsl:value-of select="concat(lname, ' ')"/>

                            <xsl:value-of select="substring(fname ,1 ,1)"/>.

                            <xsl:value-of select="substring(sname ,1 ,1)"/>.

                        </th>

                        <td>

                            <xsl:value-of select="//page[@num=3]/groups/group[@id=$studentGroupID]/name" />

                        </td>

                    </tr>

                </xsl:for-each>

            </table>

        </div>

</xsl:template>

<xsl:template match="//page[@num=3]/subjects"></xsl:template>

<xsl:template match="//page[@num=3]/groups"></xsl:template>

</xsl:stylesheet>