Краткое введение.

Для начала, несколько слов о ReSTfull архитектуре, которая является основным мотивом для того чтобы по-новому взглянуть на разработку web-сайтов.

Wiki: **Re**presentational **S**tate **T**ransfer **(ReST)** is a style of software architecture for distributed hypermedia systems such as the World Wide Web.

Другими словами ReST это стиль web-архитектуры (конечно, ReST можно использовать не только в Web, но в данном случае нас интересует именно эта сфера), который основан на двух главных принципах:

1. Web - это совокупность ресурсов.
2. Каждый ресурс может быть получен при помощи глобального идентификатора (URI)

Какие преимущества дает ReST?
- Единый API, т.е. данные можно запрашивать из любого приложения, написанного на любом языке, на сервере (xslt), на клиенте (ajax, ft sh/ft x) используя одни и те же запросы.
- Возможность кэширования запрашиваемых документов.
- Работа с чистыми данными в xml-формате

Связка ReST+XML позволяет получать чистые данные, но необходим третий компонент, который сериализует данные в конечное представление. И этим компонентом является XSLT, как язык, созданный непосредственно для решения задачи преобразования данных в различные представления.

Быстрый старт.

Чтобы наглядно продемонстрировать, как это все работает, создадим страничку и подключим шаблон.

Чаще всего у нас уже имеется сверстанный макет страницы и нам необходимо на его основе создать шаблон. Для начала нам будет вполне достаточно простой страницы состоящей из главного меню заголовка, и некоторого текста:

```
<html>
	<head>
		<title>Заголовок</title>
	</head>
	<body>
		<div id="main-menu">
			<ul>
				<li><a href="/company/">О компании</a></li>
				<li><a href="/catalogue/">Каталог</a></li>
				<li><a href="/contacts/">Контакты</a></li>
			</ul>
		</div>
		<div id="content">
			<h1>Hello world!</h1>
			<p>Текст на странице...</p>
		</div>
	</body>
</html>
```

Для начала создадим новую страницу, для этого в структуре жмем на «добавить страницу». В редактировании страницы в поле "Название" пишем название страницы "Моя страничка", в поле "Псевдостатический адрес (URL)" пишем mypage, в поле "Заголовок страницы" пишем "Hello world!", в поле "Контент" пишем "Какой-то там текст на странице...", сохраняем, страница готова.

Теперь в папке xsltTpls создадим шаблон с именем mytemplate.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

	<xsl:output method="html" encoding="utf-8" />

	<xsl:template match="/">
		<html>
			<head>
				<title></title>
			</head>
			<body>


				...тестируем шаблон...


			</body>
		</html>
	</xsl:template>

</xsl:stylesheet>
```

XSLT-шаблоны подключаются абсолютно точно также как и TPL-шаблоны, в этом плане ровным счетом ничего не изменилось, за исключением того, что xslt-шаблоны хранятся в директории xsltTpls. Делается это следующим образом. В настройках модуля "Структура" создаем новый шаблон, указывая его название (например: "Основной"), и путь к шаблону относительно папки xsltTpls, в нашем случае mytemplate.xsl. Для того чтобы подключить шаблон к конкретной странице, в редактировании страницы выбираем в выпадающем списке "Шаблон дизайна" название шаблона (в нашем случае: Основной), по которому будет выводиться страница.

Теперь, если зайти по адресу http://www.somedomain.com/mypage мы должны увидеть там текст "...тестируем шаблон...", если видим, значит, наш шаблон удачно подключен.

Сам по себе XSLT ничего не создает, всё, что он может это преобразовывать один формат данных в другой, в данном случае формат UMI Data в HTML. Для того чтобы увидеть нашу страницу http://www.somedomain.com/mypage в формате UMI Data нужно к адресу добавить расширение .xml, т.е. http://www.somedomain.com/mypage.xml

Наша страница должна выглядеть, примерно, следующим образом:

```
<?xml version="1.0" encoding="utf-8"?>
<result
```

```
            module="content"
            method="content"
            domain="www.somedomain.com"
            lang="ru"
            header="Hello world!"
            title="TITLE"
            request-uri="?foo=bar&p=3">

        <meta>
            <keywords>some keywords</keywords>
            <description>some description</description>
        </meta>
        <user id="27227" status="auth" login="userlogin"/>
        <parents/>
        <page id="32832" parentId="0" link="/mypage/">
            <name>Тестовая страница</name>
            <properties>
                <group id="22" name="common">
                    <title>Основные</title>
                    <property id="23" name="h1" type="string">
                        <title>Поле H1</title>
                        <value>Hello world!</value>
                    </property>
                    <property id="26" name="content" type="wysiwyg">
                        <title>Контент</title>
                        <value>Какой-то там текст на
странице...</value>
                    </property>
                </group>
            </properties>
        </page>
</result>
```

Видя перед глазами xml-представление страницы, и базовые элементы и функции XSLT/XPath мы легко можем вывести на странице то, что нам нужно. Добавим на страницу Заголовок и контент страницы:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

    <xsl:output method="html" encoding="utf-8" />

    <xsl:template match="/">
        <html>
            <head>
                <title><xsl:value-of select="/result/@title" /></title>
            </head>
            <body>
                <div id="content">
                    <h1><xsl:value-of select="//property[@name =
'h1']/value" /></h1>
                    <xsl:value-of select="//property[@name =
'content']/value" />
                </div>
            </body>
```

```
        </html>
    </xsl:template>

</xsl:stylesheet>
```

Для вывода меню воспользуемся стандартным макросом %content menu()% подробное описание которого можно посмотреть на help-dev.umi-cms.ru, и протоколом UData, подробное описание которого приводится ниже. Для того чтобы увидеть что нам возвращает данный макрос, введем в адресной строке браузера http://www.somedomain.com/udata:content/menu, в результате браузер должен отобразить следующее:

```
<?xml version="1.0" encoding="utf-8"?>
<udata module="content" method="menu">
        <items>
                <item id="12345" link="/company/">О Компании</item>
                <item id="23456" link="/catalogue/">Каталог</item>
                <item id="34567" link="/contacts/">Контакты</item>
        </items>
</udata>
```

Чтобы вставить меню в наш шаблон, мы воспользуемся стандартной функцией document():

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

    <xsl:output method="html" encoding="utf-8" />

    <xsl:template match="/">
        <html>
            <head>
                <title><xsl:value-of select="/result/@title" /></title>
            </head>
            <body>
                <div id="main-menu">
                    <ul>
                        <xsl:apply-templates
select="document('udata://content/menu')//item" mode="menu" />
                    </ul>
                </div>
                <div id="content">
                    <h1><xsl:value-of select="//property[@name =
'h1']/value" /></h1>
                    <xsl:value-of select="//property[@name =
'content']/value" />
                </div>
            </body>
        </html>
    </xsl:template>

    <xsl:template match="item" mode="menu">
        <li><a href="{@link}"><xsl:value-of select="." /></a></li>
    </xsl:template>
```

```
</xsl:stylesheet>
```

Шаблон готов.

Описание формата UMI Data.

Корневой элемент **result**

Дочерние элементы: meta, user, parents, page
Атрибуты:
@module - название модуля. Указывает, к какому модулю принадлежит текущая страница
@method - название метода, указывает на то, чем является текущая страница в рамках данного модуля
@domain - название домена (www.somedomain.com)
@lang - префикс языка (ru, en, ...)
@header - заголовок H1 (свойство страницы)
@title - TITLE (получается соединением префикса, указываемого в настройках SEO и свойства TITLE текущей страницы)
@request-uri - GET-параметры (?foo=bar&p=3... etc.)

Элемент **meta**

Содержит meta-информацию о странице: ключевые слова и описание
Дочерние элементы: keywords, description

Элемент **keywords**

Содержит ключевые слова текущей страницы, которые задаются в настройках SEO или при редактировании страницы

Элемент **description**

Содержит описание текущей страницы, которое задается в настройках SEO или при редактировании страницы

Элемент **user**

Описывает посетителя сайта
Атрибуты:
@id - идентификатор пользователя
@status [= "auth"] - в случае если пользователь авторизован на сайте
@login - логин пользователя

Элемент **parents**

Описывает родительские страницы
Дочерние элементы: page

Элемент **page**

Описывает текущую страницу
Дочерние элементы: name, properties

Атрибуты:
@id - идентификатор страницы
@parentId - идентификатор родительской страницы
@link - url страницы

Элемент **name**

Название страницы, соответствует свойству "Название" в редактировании

Элемент **properties**

Свойства текущей страницы
Дочерние элементы: group

Элемент **group**

Группа свойств
Дочерние элементы: title, property

Элемент **title** - Название

Элемент **property**

Описывает свойство текущей страницы
Дочерние элементы: title, value
Атрибуты:
@id - идентификатор свойства
@name - имя свойства
@type - тип свойства

Элемент **value** - Значение

Протоколы UData, UPage, UObject, UFS

Данные протоколы используются xslt-шаблоном для получения данных из БД на сервере.

Протокол UData

По этому протоколу можно получить результат работы макросов в xml-формате
На сервере в xslt это можно сделать через функцию document(), например, для вывода новостей
берем макрос *%news lastlist('/news/', default, 3)%* и вызываем его через функцию document():

document('udata://news/lastlist/(/news/)//3')

В строке *udata://news/lastlist/(/news/)//3,* news соответствует модулю, lastlist - методу; а дальше идут параметры; если в качестве параметра передается url, то его необходимо заключить в круглые скобки (/link/), если параметр не используется, то его можно опустить - //

В результате мы получим

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<udata module="news" method="lastlist">
	<items>
		<item id="32821" link="/news/news1/" publish_time="1191933240" lent_id="23857" lent_name="Новости" lent_link="/news/">Новость номер раз</item>
		<item id="32799" link="/news/news2/" publish_time="1191575520" lent_id="23857" lent_name="Новости" lent_link="/news/">Новость номер два</item>
		<item id="32764" link="/news/news3/" publish_time="1191413580" lent_id="23857" lent_name="Новости" lent_link="/news/">Новость номер три</item>
	</items>
	<archive_link>/news/</archive_link>
	<total>10</total>
	<per_page>3</per_page>
</udata>
```

Формат данных для каждого такого запроса может отличаться, в зависимости от характера данных.
Неизменным всегда остается только корневой элемент udata и его атрибуты @module и @method

Протокол UPage

С помощью этого протокола, можно получить любую страницу в xml-формате, указав ее id или url, выглядит это так:

document('upage:///1234') или *document('upage://somesection/somepage/')*

В итоге мы получим страницу в xml-формате

```
<?xml version="1.0" encoding="utf-8"?>
<udata>
	<page id="12345" parentId="0" link="/somepage/">
		<name>Название страницы</name>
		<properties>
			<group id="68" name="common">
				<title>Основные</title>
				<property id="23" name="h1" type="string">
					<title>Поле H1</title>
					<value>Заголовок</value>
				</property>

				...

			</group>

			...

		</properties>
	</page>
</udata>
```

Кроме того, если вам не нужна вся страница, а только одно ее свойство, например h1, вы легко можете получить его указав его в запросе:

upage://somesection/somepage.h1 . В этом случае xml-document будет выглядеть следующим образом:

```
<?xml version="1.0" encoding="utf-8"?>
<udata>
	<property id="23" name="h1" type="string">
		<title>Поле H1</title>
		<value>Новости рынка</value>
	</property>
</udata>
```

Протокол UObject

Простейший формат для получения любого объекта системы по его id. По своей сути и формату он очень похож на upage://, и являясь дополнением к нему, служит для получения тех объектов, которые не являются страницами сайта, например, с помощью данного протокола, можно получить информацию о зарегистрированном пользователе, с помощью запроса uobject://someId, где someId - идентификатор (id) пользователя.

Формат данных отличается от upage только тем, что элемент page здесь заменен на object и у него отсутствуют атрибуты @link и @parentId, а вместо них присутствует атрибут @name - имя объекта

```
<?xml version="1.0" encoding="utf-8"?>
<udata>
	<object id="14" name="user">
		<properties>
			<group id="16" name="idetntify_data">
				<title>Идентификационные данные</title>
				<property id="45" name="login" type="string">
					<title>Логин</title>
					<value>user</value>
				</property>
				<property id="53" name="groups" type="relation">
					<title>Группы пользователей</title>
					<value>
						<item id="15" name="Супервайзеры"/>
					</value>
				</property>

				...

			</group>
		</properties>
	</object>
</udata>
```

Так же как и с upage://, с помощью uobject:// можно получать только одно свойство объекта, например:
uobject://someId.login

```
<?xml version="1.0" encoding="utf-8"?>
<udata>
```

```
        <property id="45" name="login" type="string">
                <title>Логин</title>
                <value>lyxsus</value>
        </property>
</udata>
```

Протокол UFS

Данный протокол предназначен для получения данных о структуре папок и файлов, а также свойствах файлов, расположенных на сайте. Другими словами, это протокол для работы со статичными данными. Это может быть удобным, к примеру, для создания документации или справки из статичных xml-файлов любого формата. С помощью ufs:// легко создать индексный файл, соответствующий структуре созданной в файловой системе.

При запросе ufs://somedirectory, мы получим информацию о всех папках и файлах, расположенных в директории с именем somedirectory, формат данных будет выглядеть следующим образом:

```
<udata>
        <directory name="some-directory-name"/>
        <directory name="other-directory-name"/>
        <file name="some-document-name.xml" ext="xml"/>
        <file name="some-imagefile-name.png" ext="png"/>
        <file name="some-musicfile-name.mp3" ext="mp3"/>
</udata>
```

По умолчанию, мы получаем структуру нулевого уровня вложенности. Если нам необходимо получить структуру с большей глубиной, то при запросе нужно добавить параметр depth со значением > 0, например

ufs://somedirectory?depth=1

```
<udata>
        <directory name="some-directory-name">
                <directory name="inner-directory-name"/>
                <file name="inner-file-name.xml" ext="xml"/>
        </directory>
        <file name="some-file-name.xml" ext="xml"/>
        <file name="some-imagefile-name.png" ext="png"/>
        <file name="some-musicfile-name.mp3" ext="mp3"/>
</udata>
```

Если вам потребуется получить полную структуру, с максимальным уровнем вложенности, вы можете указать параметр depth=-1. Но! Вы должны понимать, что чем больше файловая структура, тем выше будет время генерации и последующей обработки документа. Пользуйтесь этим только тогда, когда это действительно необходимо.

При запросе конкретного файла по протоколу ufs://, если это валидный xml-файл, то его содержимое полностью копируется в элемент file, в случае невалидности содержимое копируется как CDATA.
В случае если запрашиваемый файл является картинкой, в элемент file добавляются свойства этой картинки:

```
<?xml version="1.0" encoding="utf-8"?>
<udata>
```

```
        <file name="pic.jpg" ext="jpg" mimeType="image/jpeg">
                <imageWidth>100</imageWidth>
                <imageHeight>100</imageHeight>
        </file>
</udata>
```

В случае поддержки хостингом дополнительной php-библиотеки EXIF (Exchangeable Image File Format)можно получить все возможные свойства картинки.

То же самое можно получить для всех файлов в директории, указав при запросе дополнительный параметр all, например: ufs://somedirectory?all

```
<udata>
        <file name="pic1.jpg" ext="jpg" mimeType="image/jpeg">
                <imageWidth>100</imageWidth>
                <imageHeight>100</imageHeight>
        </file>
        <file name="pic2.jpg" ext="jpg" mimeType="image/jpeg">
                <imageWidth>200</imageWidth>
                <imageHeight>100</imageHeight>
        </file>
        <file name="pic3.jpg" ext="jpg" mimeType="image/jpeg">
                <imageWidth>300</imageWidth>
                <imageHeight>200</imageHeight>
        </file>
</udata>
```

Получение данных по http. Данные можно получать и через http (необходимое условие ReSTfull-архитектуры для интернет- приложений). Для этого, предположим, вам требуется вставить новости на сайт, на сервере в xslt-шаблоне вы можете сделать это, используя функцию document() следующим образом:

document('udata://news/lastlist/(/news/)//3')

чтобы получить то же самое на клиенте, тот же самый запрос можно сделать по http:

http://udata:news/lastlist/(/news/)//3

Аналогичным образом можно поступить с протоколами upage://, uobject:// и ufs://

Примечание. Для того чтобы протоколы работали через http, необходимо в корне создать пустые файлы scheme.udata.allow, scheme.upage.allow, scheme.uobject.allow, scheme.ufs.allow

Параметры.

В xslt-шаблоне можно использовать параметры, передаваемые GET-запросом.
Например, при запросе
http://www.somedomain.com/somepage?p=3&property[name]=bububu передаются

параметры p и property[name]. Для того чтобы использовать эти параметры в шаблоне необходимо завести параметры с аналогичными именами, т.е.

<xsl:param name="p"/>
<xsl:param name="property.name" />

Примечание. Поскольку в названии параметров нельзя использовать квадратные скобки [...], то параметр property[name] необходимо записывать через точку property.name (param[foo][bar] <=> param.foo.bar)

Определяя параметры можно задать дефолтное значение:

<xsl:param name="p" select="0" />

- Справочник по XSLT

Спецификация по XSLT 1.0 http://www.w3.org/TR/xslt
(перевод: http://www.rol.ru/news/it/helpdesk/xslt01.htm)

Рекомендуемая литература:

из серии must-have:
- XSLT 2.0. Programmer's Reference. Michael Key. (Wrox) - 2004
(https://www.ozon.ru/context/detail/id/3166536/)
- XPath 2.0. Programmer's Reference. Michael Key (Wrox) - 2004
(https://www.ozon.ru/context/detail/id/3166533/)

- XSLT Cookbook, Second Edition. Sal Mangano (O'Reilly) - 2005
(https://www.ozon.ru/context/detail/id/1829874/)

из того, что можно найти на русском:
- XSLT. Справочник программиста. Майкл Кей - 2002
(https://www.ozon.ru/context/detail/id/1139457/)
- Технология XSLT. Алексей Валиков (БХВ-Петербург) - 2002
(https://www.ozon.ru/context/detail/id/964846/)
- XSLT. Библиотека программиста. Стивен Холзнер (Питер) - 2002
(https://www.ozon.ru/context/detail/id/1298917/)
- XML: разработка Web-приложений. Алексей Старыгин (БХВ-Петербург) - 2003
(https://www.ozon.ru/context/detail/id/1318178/)

N.B. Это описание может быть полнее и интереснее, если вы подскажете чего не хватает лично вам, для полного понимания. Все вопросы и пожелания вы можете присылать по адресу help@umi-cms.ru