Создание HTTP Web-сервисов

С помощью Домино8 можно создавать разнообразные HTTP Web-сервисы, которые обеспечивают доступ сторонних приложений к возможностям инфраструктуры Домино и конкретных проектных решений.

Экземпляр Домино8, запущенный в режиме сервера, предоставляет однопоточный HTTP-прослушиватель (listener), который интегрируется в инфраструктуру HTTP- сервера OCWindows. Сервер Домино8 принимает HTTP- запросы от клиентов, обрабатывает их с использованием проектных процедур и возвращает результаты обработки.

Масштабирование решения достигается пут`м запуска нескольких экземпляров сервера Домино8 и созданием диспетчера-балансировщика нагрузки, который распределяет клиентские запросы между оконечными серверами-исполнителями.

Запуск и работа Домино8 в режиме сервера

Для запуска приложения в режиме сервера следует указать следующие параметры:

- <Проект> – первый (позиционный) параметр, указывает путь к файлам проектного решения
- /SERVER – ключ указывает, что приложение должно выполняться в режиме сервера
- LISTEN – http-адрес, который будет прослушивать этот экземпляр сервера
- DBSERVER – имя сервера oracle
- SCHEME – имя схемы на сервере oracle, в которой хранится БД Домино
- USERNAME – пользователь Домино, от имени которого будет работать этот экземпляр сервера
- USERPWD – пароль пользователя Домино
- TOKEN – уникальный идентификатор экземпляра приложения

Пример строки запуска с проектом RETAIL-STORE:

> C:\Domino\RETAIL-STORE-2010\Bin\Domino8.exe
> C:\Domino\RETAIL-STORE-2010\Project\RETAIL-STORE
> /SERVER
> LISTEN=http://localhost:8081/domino/
> DBSERVER=oracle6

```
SCHEME=TEST
USERNAME=Администратор
TOKEN=2fdb96d1-952f-4ac3-8c49-213afb4886c1
```

Адрес для прослушивания (URI) задаётся в форме http://IP_or_Name:Port/RootPath/. Адрес должен начинаться на http:// и заканчиваться на /. Этот адрес будет корнем, относительно которого будут именоваться описанные в проекте Http-сервисы и методы. Например, если указан URI http://localhost:8081/Domino/Store/ ,и в проекте описан сервис с именем first/svc1 и методами m1 и m2, то эти методы будут доступны по адресам http://localhost:8081/Domino/Store/first/svc1/m1 и http://localhost:8081/Domino/Store/first/svc1/m2.

Имя сервера oracle (DBSERVER) может быть либо ссылкой на файл tnsnames, либо полным путём в соответствии с правилами SQL*Net. Описание БД в проекте при работе в режиме сервера не используется. В примере показан вариант со ссылкой на файл tnsnames. Пример задания полного пути к серверу:

```
DBSERVER=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=192.168.1.33)(PORT=1
```

Уникальный идентификатор приложения (токен) используется для удалённой остановки экземпляра. Рекомендуется использовать достаточно длинное уникальное значение, например GUID.

Запуск в режиме сервера отличается от обычного запуска следующим:

- Файл сертификата не загружается, лицензия не проверяется
- Главное окно приложения не создаётся и не отображается на экране
- Модуль работы с оборудованием не инициализируется
- Действия «после старта», описанные в проекте, выполняются. Действия «После входа пользователя» не выполняются
- Отображение окон сообщений и интерфейсных окон блокируется.

Работа приложения в режиме сервера протоколируется в стандартный файл журнала LOG\Alert_ИмяПродукта.log. Протоколируются поступающие Http-запросы и ответы. Кроме того, в каталогах LOG\WEB\<OsPID> отдельными файлами протоколируются все Http-сессии обмена клиент-сервер. Имя файла равно порядковому номеру сессии. В файл записываются исходный http- запрос со всеми параметрами, данные запроса (content), и данные ответа.

Описание Http-сервисов и методов в проекте

Все http- сервисы и их методы описываются в специальном разделе проекта «HTTP-сервисы».

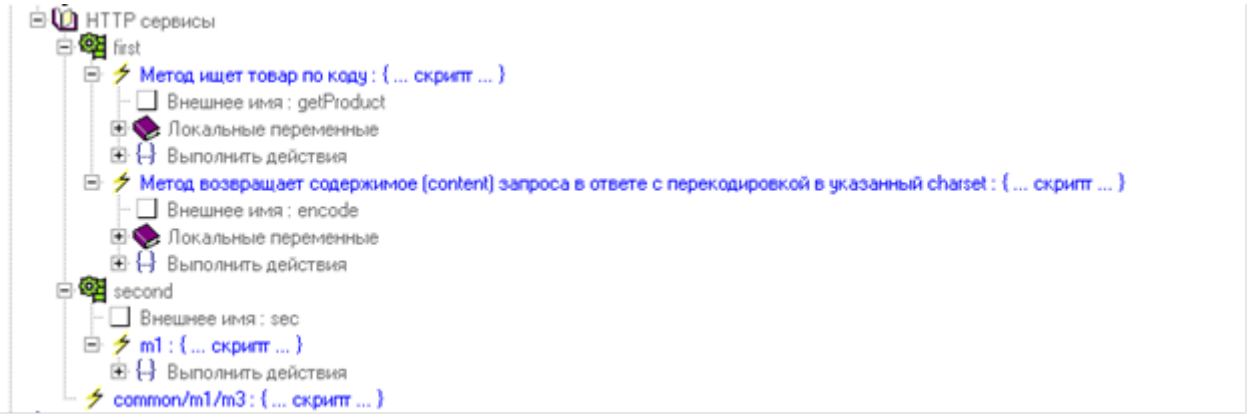
Http- методы – это именованные проектные функции, которые вызываются http-прослушивателем в ответ на входящие запросы. Могут наследоваться от 249[39780357] "{ ... скрипт ... }", либо от любой другой проектной функции. Методы вызываются http-прослушивателем по имени, поэтому имя метода должно быть корректным с точки зрения синтаксиса URI. Если у метода указан атрибут «Внешнее имя», то в качестве URI используется его значение. Собственное имя метода может в этом случае быть любым. Внутри одного сервиса не должно быть двух (активных) методов с одинаковыми именами.

Http-сервис представляет собой именованную папку (раздел), в которой описаны один или несколько методов. Имя сервиса используется в качестве локального пути к его методам, и поэтому оно должно быть корректным локальным URI. Имя сервиса может быть составным – например a/b/c. Если у сервиса указан атрибут «Внешнее имя», то в качестве URI используется его значение. Собственное имя сервиса может в этом случае быть любым удобоваримым именем. Имя сервиса может быть пустым.

Внутри сервиса кроме собственно методов могут быть описаны вложенные сервисы. Путь к методам вложенного сервиса формируется соединением их путей. Например если внутри сервиса «a» описан вложенный сервис «b», то путь к методам «b» будет a/b

Регистр в наименованиях сервисов, методов и параметров игнорируется. Записи first/svc1/m1 и First/Svc1/M1 равнозначны.

Приведённый ниже пример описывает два сервиса - first и second. Сервис first реализует два метода (first/getProduct и first/encode), имена которых определяются атрибутом «Внешнее имя». Сервис second с внешним именем sec реализует один метод (sec/m1). Кроме того, описан метод с именем common/m1/m3, не входящий ни в один из сервисов.



Для Http- методов допустимы подстановки точно так же, как и для обычных проектных функций. С помощью подстановок можно изменять логику методов.

Для того, чтобы отключить (заблокировать) некоторый метод, используется стандартный признак «Использование (в режиме выполнения)». Он может быть как безусловным (укажите «Нет»), так и условным. Этот признак можно задавать как для самого метода, так

и для его подстановки.

Для того, чтобы полностью заместить метод, отключите его с помощью подстановки, указав «Использование (в режиме выполнения)»=Нет. После этого опишите еще один метод с точно таким же именем, как у замещаемого метода.

Реализация Http-методов

Http- методы **–** это именованные проектные функции, которые вызываются http-прослушивателем в ответ на входящие запросы. Методы могут наследоваться от 249[39780357] "{ ... скрипт ... }", либо от любой другой проектной функции.

Для получения параметров из строки http- запроса по их имени используется функция 387[63111218] "HTTPService.GetRequestParam". Для удобочитаемости рекомендуется описать локальные переменные с именами, совпадающими с именами параметров, и присвоить им начальные значения через вызов функции GetRequestParam. Такая нотация позволяет по заголовку метода легко понять, какие он принимает параметры.

Есть так же возможность получить значение множественного параметра, который повторяется в строке запроса несколько раз. Для этого при вызове функции кроме имени нужно указать порядковый номер (индекс) параметра в запросе. Например, GetRequestParam('id', 1) возвращает первое вхождение параметра id в запрос, GetRequestParam('id', 4) - четвёртое.

Параметры http-запроса могут быть без имени (точнее, с пустым именем ''). Знак равенства в этом случае ставить не обязательно. У запроса «имяМетода?aaa&=bbb&ccc» три параметра с пустым именем и значениями aaa, bbb и ccc.

Параметры так же могут быть переданы в содержимом POST- запроса с указанием Content-Type:application/x-www-form-urlencoded.

Полный текст http-запроса с путём, именем метода и всеми параметрами возвращает функция 387[63111215] "HTTPService.GetRequest ". Запрос возвращается в «сыром» (raw) формате, т.е. параметры будут кодированы согласно RFC-1522. Это низкоуровневая возможность, и для написания обычных сервисов она, как правило, не требуется.

Функция 387[67305508] "**HTTPService.GetRequestHttpMethod"** возвращает тип входящего запроса (GET, POST…).

Функция 387[67305509] " **HTTPService.GetRequestRemoteAddress**" возвращает ip-адрес клиента, приславшего запрос.

Для получения параметров заголовка http- запроса по их имени используется функция 387[63111219] "HTTPService.GetRequestHeader". Функция так же относится к низкоуровневым, и применяется редко для реализации специфических возможностей.

Для получения содержимого http POST- запроса используется функция 387[63111216] "**HTTPService.GetRequestContent**

". Содержимое возвращается в виде единой строки. При приёме происходит автоматическое перекодирование данных с использованием атрибута charset параметра заголовка Content-Type. Двоичные типы (application/octet-stream, image/jpeg и т.п.) и multipart- типы не поддерживаются. Если тип содержимого запроса application/x-www-form-urlencoded, то оно автоматически преобразуется в параметры http- запроса: для этого типа функция GetRequestContent будет возвращать пустое значение.

Для получения типа содержимого http POST- запроса (content-type) используется функция 387[63111217] "**HTTPService.GetRequestContentType**". Она возвращает значение параметра Content-Type заголовка http- запроса, Эквивалентно вызову GetRequestHeader('Content-Type').

Рекомендованным форматом для содержимого запросов и ответов является text/xml. Для работы с данными в этом формате есть необходимый проектный функционал. Однако это не является обязательным требованием.

Функция-метод может возвращает содержимое (content) ответа только текстового типа. Бинарные форматы не поддерживаются. По умолчанию текст ответа возвращается в кодировке utf-8, есть возможность явно указать кодировку.

Функция-метод может возвращать содержимое ответа двумя способами: либо как раздел <data> стандартного xml-контейнера ответа, либо формируя его целиком. В первом случае ответ обязательно должен быть корректной xml-структурой без заголовка. Во втором случае нет никаких ограничений на формат ответа кроме того, что он должен быть текстовым.

Стандартный xml-контейнер ответа имеет вид

```
<?xml version="1.0" encoding="Кодировка ответа"?>
<response>
 <cmd>Имя метода</cmd>
 <params>
  <param>
   <name>Имя первого параметра</name>
   <value>Значение первого параметра </value>
  </param>
  ... Остальные параметры ...
 </params>
 <data>
  Содержимое ответа в xml-формате
 </data>
 <result>
  <code>
   Код завершения метода (0 - успешно)
```

```
 </code>
 <msg>
  Текст сообщения об ошибке (описание кода завершения)
 </msg>
</result>
</response>
```

Для того, чтобы вернуть содержимое как раздел <data> стандартного контейнера, необходимо сформировать его как строку в xml-формате (без стандартного заголовка <?xml version=... ?>) и передать функции 387[63111221] "**HTTPService.SetResponseData**". Переданное содержимое не проверяется на корректность.

Если функция-метод возвращает ответ в стандартном xml-контейнере, то параметр Content-Type в заголовке ответа автоматически устанавливается в text/xml;charset=КодировкаОтвета

В разделе <result> стандартного контейнера ответа возвращаются код завершения метода и текстовое описание кода завершения. При успешном завершении метода код равен 0. В случае сбоя (аварийного прерывания) функции-метода в этот раздел записывается внутренний код Домино с расшифровкой. С помощью функции 387[63111223] "**HTTPService.SetResponseResult**" разработчик может явно установить код и текст сообщения, которые будут записаны в раздел <result>. Нужно учитывать, что некоторые коды завершения приводят в возврату специфических кодов состояния HTTP (*HTTP status code*) в ответе клиенту - см.ниже.

Для того, чтобы вернуть содержимое произвольного формата, нужно сформировать его как строку и передать функции 387[63111220] "**HTTPService.SetResponseContent**".

В качестве содержимого функции 387[63111220] "**HTTPService.SetResponseContent**" можно установить значение NULL. В этом случае стандартный xml-ответ, содержащий параметры запроса, код и описание ошибки ответа не формируется.

Для того, чтобы изменить параметр Content-Type в заголовке ответа, нужно использовать функцию 387[63111224] "**HTTPService.SetResponseContentType**". Функции следует передавать только MIME-тип сообщения без ‘;’ и параметров. Например при отправке ответа в json-формате нужно установить 'text/json' или 'application/json'. При отправке ответа система автоматически формирует параметр Content-Type как 'MIME-тип;charset=КодировкаОтвета'

По умолчанию текст ответа возвращается в кодировке utf-8. Для того, чтобы изменить кодировку ответа, нужно использовать функцию 387[63111225] "**HTTPService.SetResponseContentCharset**". Функции нужно передать либо MIME-наименование кодировки, либо номер кодовой страницы. Например, значения ‘windows-1251’ и 1251 равнозначны.

Для прямого изменения параметров заголовка http- ответа используется функция 387[63111222] "**HTTPService.SetResponseHeader**". Функция относится к низкоуровневым, ей следует пользоваться с осторожностью.

Функция-метод может возвращать целое значение. Это значение используется в качестве кода состояния HTTP (*HTTP status code*) в ответе клиенту. Может принимать значение от 200 до 599, остальные значения игнорируются. Если метод не возвращает ничего, то в ответе будет установлен код состояния 200 Ok.

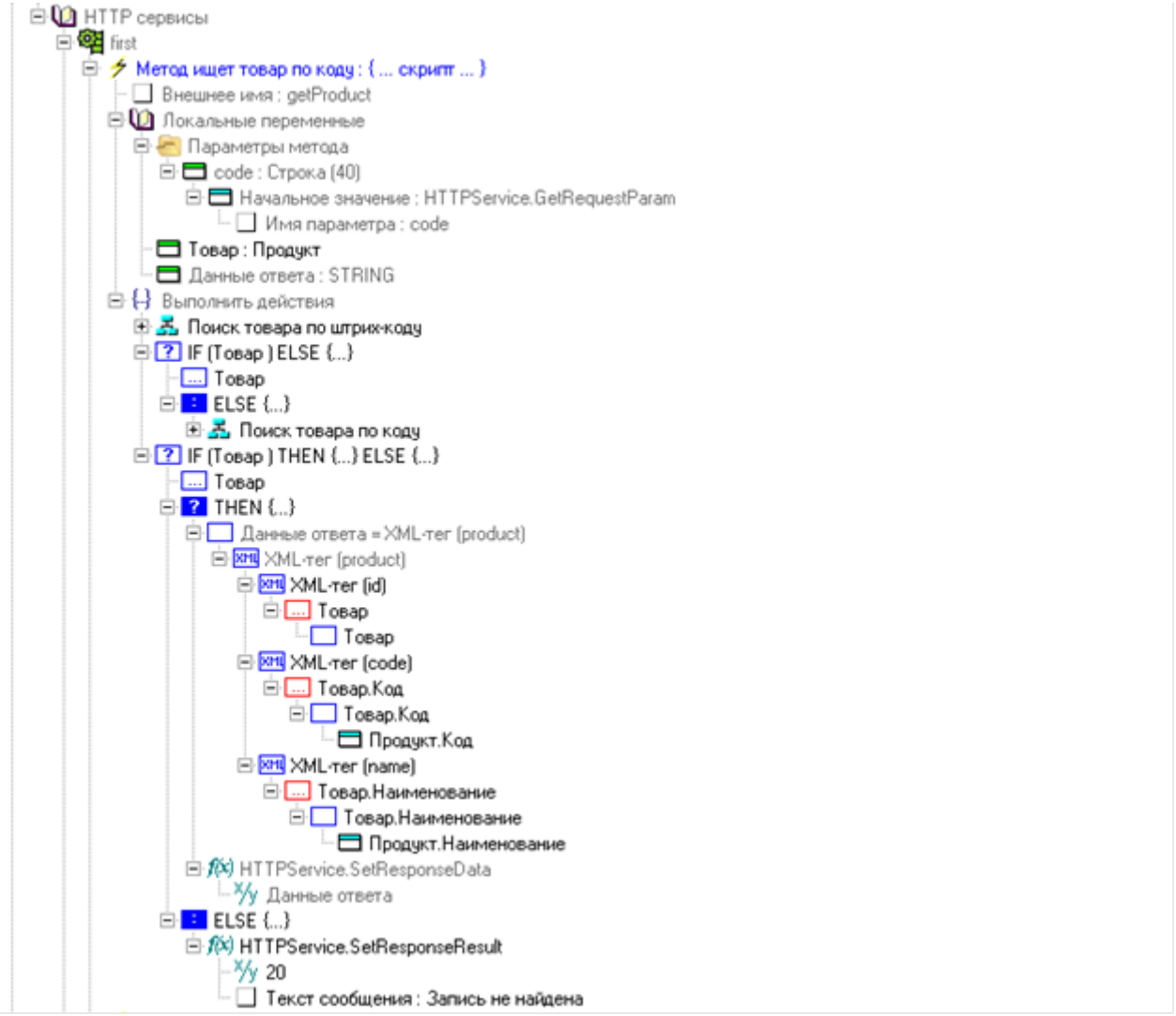
Если метод завершается с ошибками класса 244 «Системная ошибка» или 254 «Ошибка в описании проекта», то в ответе устанавливается код состояния 500 Internal server error.

Если прослушиватель не находит метод по имени, то он возвращает код состояния 404 Not found. при этом в качестве содержимого ответа возвращается стандартный xml-контейнер без раздела <data>, <result><code>-1</code><msg>Неизвестная команда</msg></result>. Нужно помнить, что при поиске метода по имени учитывается признак *«Использование (в режиме выполнения)»*

Пример 1. Поиск товара по коду

Метод first/getProduct ищет товар по продажному коду или коду, и возвращает uid, код и имя найденного продукта в разделе <data> стандартного xml-контейнера ответа. В случае, если продукт не найден, устанавливает код завершения 20, текст сообщения «Запись не найдена».

В примере использованы функции *HTTPService.GetRequestParam* для получения значения параметра из строки запроса, *HTTPService.SetResponseData* для записи результат в раздел <data>, *HTTPService.SetResponseResult* для установки кода завершения «Запись не найдена». Для формирования результата в xml- формате применены встроенные операторы генерации xml (XML-тег)



Пример 2. Перекодировщик произвольных строковых данных

Метод first/encode демонстрирует применение функций низкого уровня. Он принимает на вход текстовые данные в произвольной кодировке, и возвращает их в кодировке, которая указана параметром charset запроса.



Пример вызова метода утилитой curl. Входные данные в кодировке windows-1251 должны

находиться в файле first.encode.in.txt. Выходные данные в кодировке utf-16 будут записаны в файл first.encode.out.txt

```
curl.exe
    -o first.encode.out.txt
    --trace-ascii first.encode.trc
    -H "Content-Type: text/plain;charset=windows-1251"
    -d @first.encode.in.txt
    "http://localhost:8081/domino/first/encode?charset=utf-16"
```

Проектные процедуры и функции

Для взаимодействия метода с контекстом Http-прослушивателя ядро Домино8 предоставляет специальные функции. Этот раздел содержит полный список таких функций.

387[63111215] "HTTPService.GetRequest : STRING"

Функция возвращает полный текст http-запроса с путём, именем метода и всеми параметрами. Запрос возвращается в «сыром» (raw) формате, т.е. параметры будут кодированы согласно RFC-1522. Это низкоуровневая возможность, и для написания обычных сервисов она, как правило, не требуется.

387[63111216] "HTTPService.GetRequestContent : STRING"

Функция возвращает содержимое (content) POST- запроса как строку

387[63111217] "HTTPService.GetRequestContentType : STRING"

Функция возвращает значение параметра Content-Type заголовка запроса.

387[63111218] "HTTPService.GetRequestParam : STRING"

Функция возвращает значение параметра из URI запроса по его имени. Имя параметра задаётся параметром функции «Имя параметра». Если параметр отсутствует, то функция возвращает NULL. Для неуникальных параметров запроса можно параметром «Индекс» указать их номер по порядку начиная с 1. Например, HTTPService.GetRequestParam('id', 5) возвращает значение 5 вхождения параметра id в строку запроса.

387[63111219] "HTTPService.GetRequestHeader : STRING"

Функция возвращает значение параметра из заголовка запроса по его имени. Имя параметра заголовка задаётся параметром функции «Имя параметра». Для неуникальных параметров заголовка можно параметром «Индекс» указать их номерпо порядку начиная с 1 (аналогично GetRequestParam).

387[63111220] "HTTPService.SetResponseContent : void"

Функция устанавливает текстовое содержимое ответа целиком, без ограничения на формат. Её вызов отключает возврат из http-метода содержимого в форме стандартно xml- контейнера. Содержимое передаётся параметром функции «Выражение» как строка произвольного формата. Бинарные типы содержимого не поддерживаются.

387[63111224] "HTTPService.SetResponseContentType : void"

Функция устанавливает MIME- тип содержимого ответа (Content-Type=). Имеет смысл только если http- метод возвращает содержимое произвольного формата функцией HTTPService.SetResponseContent. Если метод возвращает стандартный xml-контейнер ответа, то его тип автоматически устанавливается в text/xml. Значение MIME- типа передаётся параметром функции «Выражение».

387[63111225] "HTTPService.SetResponseContentCharset : void"

Функция устанавливает кодировку (charset=) содержимого ответа. По умолчанию текст ответа возвращается в кодировке utf-8. Для того, чтобы изменить кодировку ответа, нужно передать либо MIME- наименование кодировки, либо номер кодовой страницы. Например, значения ‘windows-1251’ и 1251 равнозначны. Кодировка передаётся параметром функции «Выражение».

387[63111221] "HTTPService.SetResponseData : void"

Функция устанавливает значение раздела <data> стандартного контейнера ответа. Значение передаётся параметром функции «Выражение» как строка в xml-формате (без стандартного заголовка <?xml version=... ?>).

387[63111222] "HTTPService.SetResponseHeader : void"

Функция устанавливает значение параметра в заголовке ответа. Имя параметра заголовка передаётся параметром функции «Имя параметра». Значение параметра - параметром функции «Выражение». Имя параметра следует задавать без двоеточия на конце. Параметры заголовка неуникальны - можно добавить любое число одноименных параметров. Некоторые системные параметры - например, Content-Length, нельзя установить таким образом: параметр не будет записан в заголовок ответа и в AlertLog приложения останется запись об ошибке.
Функция относится к низкоуровневым, ей следует пользоваться с осторожностью.

387[63111223] "HTTPService.SetResponseResult : void"

Функция устанавливает код завершения http-метода и текст сообщения, которые будут возвращены в разделе <result> стандартного контейнера ответа. Код завершения передаётся параметром функции «Выражение» как целое число. Код 0 соответствует успешному завершению. Текст сообщения передаётся параметром функции «Текст сообщения» как строка. Нужно учитывать, что некоторые коды завершения приводят в возврату специфических кодов состояния HTTP (*HTTP status code*) в ответе клиенту.

249[63111299] "UID как строка в RAW-формате"

Функция - уточняющий параметр объекта. Возвращает UID объекта как строку в RAW формате

249[63111298] "Преобразовать UID/UIDSET к строке в RAW-формате"

Функция преобразует значение UID или UIDSET, заданное параметром «Выражение», к строке в RAW-формате

Встроенные методы

Кроме методов, реализованных на уровне проекта, Http-сервер Домино предоставляет встроенные методы.

Получить значение одного или всех атрибутов объекта по его UID (**get**)

Возвращает значение указанного, либо всех атрибутов объекта. UID объекта задаётся параметром id=. Это может быть как UID объекта БД, так и UID проектного элемента. Имя атрибута, значение которого нужно вернуть, задаётся параметром attr=. Если параметр не зада, то возвращаются все атрибуты объекта.

Для объектов БД имя атрибута может быть либо числовым идентификатором (индекс поля в проекте), либо строковым именем, совпадающим с именем поля в таблице БД oracle. Например для товара (product) записи attr=5 и attr=CODE равнозначны.

Для проектных элементов имя атрибута может быть одним из списка (NAME, VALUE, ABBREVIATION, EXT_NAME, TYPE, TYPE_NAME).

Результат возвращается в стандартном xml-контейнере в разделе <data> в. структуре <record>. Каждый атрибут возвращается отдельным тегом, имя тега совпадает с именем атрибута.

Пример запроса проектного элемента и результат (ответ)

```
GET http://host:port/RootPath/get?id=332[1966128]

<response>
 <cmd>get</cmd>
 <params>
  <param>
   <name>id</name><value>42401836[42401793]</value>
  </param>
 </params>
 <data>
  <record>
   <ID>0287002C02870001</ID>
   <NAME>ВОДКА</NAME>
   <TYPE>000000010287002C</TYPE>
   <TYPE_NAME>Кодификатор алкогольной продукции (ГАК)</TYPE_NAME>
   <VALUE/>
   <ABBREVIATION>0318</ABBREVIATION>
  </record>
 </data>
 <result><code>0</code></result>
</response>
```

Пример запроса объекта БД (продукт, атрибут «Наименование») и результат (ответ)

> GET http://host:port/RootPath/get?id=3:0:0:176665&attr=NAME

```
<response>
 <cmd>get</cmd>
 <params>
  <param>
   <name>id</name><value>3:0:0:176665</value>
   <name>attr</name><value>NAME</value>
  </param>
 </params>
 <data>
  <record>
   <ID>8C0000000002B219</ID>
   <NAME>Ранец orange Алмаз дет 600Д</NAME>
  </record>
 </data>
 <result><code>0</code></result>
</response
```

Создать объект в БД (**insert**)

Создаёт новый объект в БД. Категория объекта (таблица) задаётся параметром table=. Возможные значения USER, CLASSIF, PARTNER, PRODUCT, DOCUMENT, LINE. Тип объекта (числовой, по проекту) задаётся параметром type=.

Значения атрибутов для создаваемого объекта передаются в содержимом (content) POST-запроса в xml-формате. Структура контейнера

```
“
<?xml version="1.0" encoding="Кодировка запроса"?>
<request>
```

```
 <data>
  <record>
   <ИмяАтрибута1>Значение атрибута 1</ИмяАтрибута1>
   <ИмяАтрибута2>Значение атрибута 2</ИмяАтрибута2>
   ...
   <ИмяАтрибутаN>Значение атрибута N</ИмяАтрибутаN>
  </record>
 </data>
</request>
```

Имя атрибута должно совпадать с именем поля в таблице БД oracle. Например для товара (product) 305[5]"Код" - <CODE>, 305[6] "Наименование" - <NAME>, 305[14745607] "ЕИ" - <F14745607>

Возвращает стандартный xml-контейнер с кодом завершения и описанием результата. В разделе <data> в структуре <record> возвращается атрибут ID, содержащий UID созданного объекта.

Обновить значения атрибутов объекта БД (**update**)

Изменяет значения атрибутов существующего объекта БД. UID объекта задаётся параметром id=.

Значения атрибутов для изменения передаются в содержимом (content) POST- запроса в xml-формате. Структура контейнера идентична тому, который используется в операции insert

Возвращает стандартный xml-контейнер с кодом завершения и описанием результата

Удалить объект из БД (**delete**)

Удаляет объект из БД, UID которого задаётся параметром id=.

Возвращает стандартный xml-контейнер с кодом завершения и описанием результата

Акцептовать объект (**accept**)

Выполняет стандартный акцепт объекта, UID которого задаётся параметром id=.

Возвращает стандартный xml-контейнер с кодом завершения и описанием результата

Деакцептовать объект (**deaccept**)

Выполняет стандартный деакцепт объекта, UID которого задаётся параметром id=.

Возвращает стандартный xml-контейнер с кодом завершения и описанием результата

> Встроенные методы принимают UID объектов как в стандартном строковом, таки в RAW-формате. Встроенные методы всегда возвращают UID объектов в RAW-формате

> Для блокировки встроенных методов нужно в проекте описать одноименные методы, которые будут возвращать ошибку. Проектные методы проверяются первыми и имеют приоритет над встроенными.

Версия #5

Демонов Сергей создал Mon, Mar 21, 2022 4:09 PM

Демонов Сергей обновил Mon, Jul 3, 2023 7:48 AM