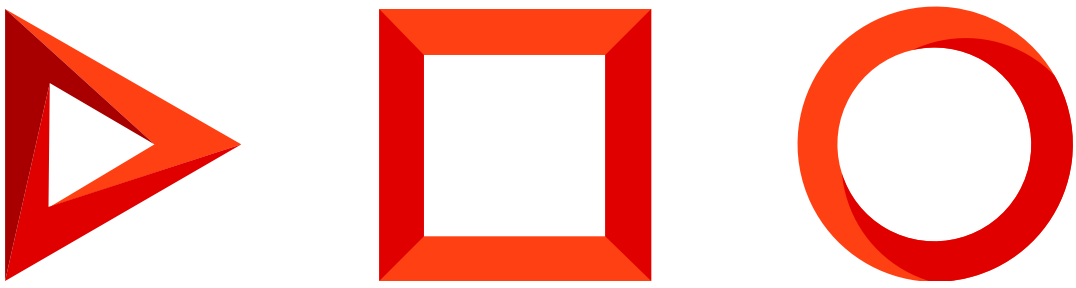
Creatio Academy

Back-end разработка

Пользовательские веб-сервисы

Версия 8.0



Эта документация предоставляется с ограничениями на использование и защищена законами об интеллектуальной собственности. За исключением случаев, прямо разрешенных в вашем лицензионном соглашении или разрешенных законом, вы не можете использовать, копировать, воспроизводить, переводить, транслировать, изменять, лицензировать, передавать, распространять, демонстрировать, выполнять, публиковать или отображать любую часть в любой форме или посредством любые значения. Обратный инжиниринг, дизассемблирование или декомпиляция этой документации, если это не требуется по закону для взаимодействия, запрещены.

Информация, содержащаяся в данном документе, может быть изменена без предварительного уведомления и не может гарантировать отсутствие ошибок. Если вы обнаружите какие-либо ошибки, сообщите нам о них в письменной форме.

Содержание

Пользовательские веб-сервисы

Средний

Веб-сервис — идентифицируемая уникальным веб-адресом (URL-адресом) программная система, которая обеспечивает взаимодействие между приложениями. **Назначение** веб-сервиса — настройка интеграции между Creatio и внешними приложениями и системами.

На основе пользовательской бизнес-логики Creatio сгенерирует и отправит запрос веб-сервису, получит ответ и предоставит необходимые данные. Эти данные можно использовать для создания или обновления записей в Creatio, а также для реализации пользовательской бизнес-логики или автоматизации.

Виды веб-сервисов в Creatio:

- **Внешние REST и SOAP-сервисы**, с которыми можно настроить интеграцию low-code инструментами. Подробнее читайте в блоке статей Веб-сервисы документации для пользователя.
- **Системные веб-сервисы**.
  - Системные веб-сервисы с аутентификацией на основе cookies.
  - Системные веб-сервисы с анонимной аутентификацией.
- **Пользовательские веб-сервисы**.
  - Пользовательские веб-сервисы с аутентификацией на основе cookies.
  - Пользовательские веб-сервисы с анонимной аутентификацией.

Системные веб-сервисы, разработанные на .NET Framework, реализованы на основе технологии WCF и управляются на уровне IIS. Системные веб-сервисы, разработанные на .NET Core, реализованы на основе технологии ASP.NET Core Web API.

Виды аутентификации, которые поддерживаются веб-сервисами в Creatio, описаны в статье Аутентификация. Рекомендуемым способом аутентификации является аутентификация на основе открытого протокола авторизации OAuth 2.0, которая описана в статье Настроить авторизацию интегрированных приложений по протоколу OAuth 2.0.

Примеры **системных веб-сервисов с аутентификацией на основе cookies**, предоставляемых Creatio:

- `odata` — выполнение запросов от внешних приложений к серверу баз данных Creatio по протоколу OData 4. Описание использования протокола OData 4 в Creatio содержится в статье OData.
- `EntityDataService.svc` — выполнение запросов от внешних приложений к серверу баз данных Creatio по протоколу OData 3. Описание использования протокола OData 3 в Creatio содержится в статье OData.
- `ProcessEngineService.svc` — запуск бизнес-процессов Creatio из внешних приложений. Описание веб-сервиса содержится в статье Сервис запуска бизнес-процессов.

Примеры **системных веб-сервисов с анонимной аутентификацией**, предоставляемых Creatio:

- `AuthService.svc` — выполнение запроса на аутентификацию в приложении Creatio. Описание веб-сервиса содержится в статье Аутентификация.

В этой статье рассмотрены пользовательские веб-сервисы. Системные веб-сервисы описаны в разделе Интеграция и внешний API.

Разработать пользовательский веб-сервис

Пользовательский веб-сервис — RESTful-сервис, реализованный на базе технологии WCF (для .NET Framework) или ASP.NET Core Web API (для .NET Core). **Отличие** пользовательского веб-сервиса от системного — возможность реализации специфических интеграционных задач.

В зависимости от платформы, на которой развернуто приложение, разработка пользовательского веб-сервиса имеет свои особенности. Ниже рассмотрены особенности разработки пользовательского веб-сервиса для платформ .NET Framework и .NET Core.

Разработать пользовательский веб-сервис с аутентификацией на основе cookies

1. Создайте схему [*Исходный код*] ([*Source code*]). Для создания схемы воспользуйтесь статьей Разработка конфигурационных элементов.
2. Создайте класс сервиса.
   a. В дизайнере схем добавьте пространство имен `Terrasoft.Configuration` или любое вложенное в него пространство имен. Название пространства имен может быть любым.
   b. С помощью директивы `using` добавьте пространства имен, типы данных которых будут задействованы в классе.
   c. Используйте пространство имен `Terrasoft.Web.Http.Abstractions`, которое позволит пользовательскому веб-сервису работать на платформах .NET Framework и .NET Core. Если при разработке веб-сервиса было использовано пространство имен `System.Web` и необходимо запустить веб-сервис на платформе .NET Core, то выполните адаптацию веб-сервиса .
   d. Добавьте название класса, которое соответствует названию схемы (свойство [*Код*] ([*Code*])).
   e. В качестве родительского класса укажите класс `Terrasoft.Nui.ServiceModel.WebService.BaseService`.
   f. Для класса добавьте атрибуты `[ServiceContract]` и `[AspNetCompatibilityRequirements]` с необходимыми параметрами. Атрибут `[ServiceContract]` описан в официальной документации Microsoft. Атрибут `[AspNetCompatibilityRequirements]` описан в официальной документации Microsoft.
3. Реализуйте методы класса, которые соответствуют конечным точкам веб-сервиса.

   Для методов добавьте атрибуты `[OperationContract]` и `[WebInvoke]` с необходимыми параметрами. Атрибут `[OperationContract]` описан в официальной документации Microsoft. Атрибут `[WebInvoke]` описан в официальной документации Microsoft.
4. Реализуйте дополнительные классы, экземпляры которых будут принимать или возвращать методы веб-сервиса (опционально). Выполняется при необходимости передачи данных сложных типов (экземпляры объектов, коллекции, массивы и т. д.).

   Для класса добавьте атрибут `[DataContract]`, а для полей класса — атрибут `[DataMember]`. Атрибут

`[DataContract]` описан в официальной документации Microsoft. Атрибут `[DataMember]` описан в официальной документации Microsoft.

5. Опубликуйте схему исходного кода.

В результате пользовательский веб-сервис с аутентификацией на основе cookies будет доступен для вызова из программного кода конфигурационных схем, а также из внешних приложений.

Разработать пользовательский веб-сервис с анонимной аутентификацией

Пользовательские веб-сервисы с анонимной аутентификацией — веб-сервисы, которые не требуют предварительной аутентификации пользователя, т. е. доступны для анонимного использования.

Важно. При реализации пользовательских веб-сервисов не рекомендуется использовать анонимную аутентификацию, поскольку это может снижать безопасность и производительность.

Разработать пользовательский веб-сервис с анонимной аутентификацией для платформы .NET Framework

1. Выполните шаги 1-5 инструкции Разработать пользовательский веб-сервис с аутентификацией на основе cookies.
2. При создании класса сервиса добавьте системное подключение `SystemUserConnection`.
3. При создании метода класса укажите пользователя, от имени которого будет выполняться обработка http-запроса. Для этого после получения `SystemUserConnection` вызовите метод `SessionHelper.SpecifyWebOperationIdentity` пространства имен `Terrasoft.Web.Common`. Этот метод обеспечивает работоспособность бизнес-процессов при работе с сущностью (`Entity`) базы данных из пользовательского веб-сервиса с анонимной аутентификацией.

```
Terrasoft.Web.Common.SessionHelper.SpecifyWebOperationIdentity(HttpContextAccessor.GetInstanc
```

4. Зарегистрируйте пользовательский веб-сервис с анонимной аутентификацией:
   a. Перейдите в каталог `..\Terrasoft.WebApp\ServiceModel`.
   b. Создайте файл с названием веб-сервиса и расширением *.svc. Добавьте в него запись.

Шаблон регистрации пользовательского веб-сервиса с анонимной аутентификацией

```
<% @ServiceHost
    Service = "Service, ServiceNamespace"
    Factory = "Factory, FactoryNamespace"
    Debug = "Debug"
    Language = "Language"
    CodeBehind = "CodeBehind"
%>
```

Пример регистрации пользовательского веб-сервиса с анонимной аутентификацией

```
<% @ServiceHost
    Service = "Terrasoft.Configuration.UsrAnonymousConfigurationServiceNamespace.UsrAnonym
    Debug = "true"
    Language = "C#"
%>
```

Атрибут `Service` должен содержать полное имя класса веб-сервиса с указанием пространства имен.

WCF-директива `@ServiceHost` описана в официальной документации Microsoft.

5. Настройте пользовательский веб-сервис с анонимной аутентификацией для работы по протоколам http и https:

   a. Откройте файл `..\Terrasoft.WebApp\ServiceModel\http\services.config` и добавьте в него запись.

Пример изменений файла `..\Terrasoft.WebApp\ServiceModel\http\services.config`

```
<services>
    ...
    <service name="Terrasoft.Configuration.[Custom namespace].[Service name]">
        <endpoint name="[Service name]EndPoint"
            address=""
            binding="webHttpBinding"
            behaviorConfiguration="RestServiceBehavior"
            bindingNamespace="http://Terrasoft.WebApp.ServiceModel"
            contract="Terrasoft.Configuration.[Custom namespace].[Service name]" />
    </service>
</services>
```

`<services>` — элемент, который содержит перечень конфигураций всех веб-сервисов приложения (вложенные элементы `<service>`).

`name` — атрибут, который содержит название типа (класса или интерфейса), реализующего контракт веб-сервиса.

`<endpoint>` — вложенный элемент, который содержит адрес, привязку и интерфейс, определяющий контракт веб-сервиса, указанного в атрибуте `name` элемента `<service>`.

Описание элементов конфигурирования веб-сервиса доступно в официальной документации Microsoft.

   b. Аналогичную запись добавьте в файл `..\Terrasoft.WebApp\ServiceModel\https\services.config`.

6. Настройте доступ к пользовательскому веб-сервису с анонимной аутентификацией для всех пользователей:

   a. Откройте файл `..\Terrasoft.WebApp\Web.config`.

   b. Добавьте элемент `<location>`, определяющий относительный путь и права доступа к веб-сервису.

   Пример изменений файла `..\Terrasoft.WebApp\Web.config`

   ```
   <configuration>
       ...
       <location path="ServiceModel/[Service name].svc">
           <system.web>
               <authorization>
                   <allow users="*" />
               </authorization>
           </system.web>
       </location>
       ...
   </configuration>
   ```

   c. В атрибут `value` ключа `AllowedLocations` элемента `<appSettings>` добавьте относительный путь к веб-сервису.

   Пример изменений файла `..\Terrasoft.WebApp\Web.config`

   ```
   <configuration>
       ...
       <appSettings>
       ...
           <add key="AllowedLocations" value="[Предыдущие значения];ServiceModel/[Service name
       ...
       </appSettings>
       ...
   </configuration>
   ```

7. Перезапустите приложение в IIS.

В результате пользовательский веб-сервис с анонимной аутентификацией будет доступен для вызова из программного кода конфигурационных схем, а также из внешних приложений. К веб-сервису можно обращаться, как с предварительной аутентификацией, так и без нее.

Разработать пользовательский веб-сервис с анонимной аутентификацией для платформы .NET Core

1. Выполните шаги 1-5 инструкции Разработать пользовательский веб-сервис с аутентификацией на основе cookies.

2. Настройте доступ к пользовательскому веб-сервису с анонимной аутентификацией для всех пользователей:

Пример изменений файла `..\Terrasoft.WebHost\appsettings.json`

```
"Terrasoft.Configuration.[Service name]": [
    "/ServiceModel/[Service name].svc"
]
```

   a. Откройте конфигурационный файл `..\Terrasoft.WebHost\appsettings.json`.

   b. Добавьте информацию о веб-сервисе в блок `AnonymousRoutes` файла.

3. Перезапустите приложение.

В результате пользовательский веб-сервис с анонимной аутентификацией будет доступен для вызова из программного кода конфигурационных схем, а также из внешних приложений. К веб-сервису можно обращаться, как с предварительным вводом логина и пароля, так и без их использования.

Важно. После обновления приложения необходимо выполнить повторную настройку веб-сервиса, поскольку при обновлении приложения все конфигурационные файлы заменяются новыми.

Вызвать пользовательский веб-сервис

Пользовательский веб-сервис можно вызвать из браузера и из front-end части.

Вызвать пользовательский веб-сервис из браузера

Вызвать из браузера пользовательский веб-сервис с аутентификацией на основе cookies

Чтобы вызвать из браузера пользовательский веб-сервис с аутентификацией на основе cookies для платформы **.NET Framework**:

1. Для получения аутентификационных cookie используйте системный веб-сервис `AuthService.svc`.
2. Для вызова пользовательского веб-сервиса используйте строку запроса:

Шаблон адреса пользовательского веб-сервиса с аутентификацией на основе cookies

```
[Адрес приложения Creatio/0/rest/[Название пользовательского веб-сервиса]/[Конечная точка пол
```

Пример адреса пользовательского веб-сервиса с аутентификацией на основе cookies

```
http://mycreatio.com/0/rest/UsrCustomConfigurationService/GetContactIdByName?Name=User1
```

Для платформы **.NET Core** процедура вызова пользовательского веб-сервиса с аутентификацией на основе cookies аналогична. Различие — не нужно использовать приставку `/0`.

Вызвать из браузера пользовательский веб-сервис с анонимной аутентификацией

Чтобы вызвать из браузера пользовательский веб-сервис с анонимной аутентификацией для платформы **.NET Framework**, используйте строку запроса:

Шаблон адреса пользовательского веб-сервиса с анонимной аутентификацией

```
[Адрес приложения Creatio]/0/ServiceModel/[Название пользовательского веб-сервиса]/[Конечная точ
```

Пример адреса пользовательского веб-сервиса с анонимной аутентификацией

```
http://mycreatio.com/0/ServiceModel/UsrCustomConfigurationService.svc/GetContactIdByName?Name=Us
```

Для платформы **.NET Core** процедура вызова пользовательского веб-сервиса с анонимной аутентификацией аналогична. Различие — не нужно использовать приставку `/0`.

Вызвать пользовательский веб-сервис из front-end части

1. В модуль страницы, из которой вызывается веб-сервис, в качестве зависимости подключите модуль `ServiceHelper`. Этот модуль предоставляет удобный интерфейс для выполнения запросов к серверу через провайдер запросов `Terrasoft.AjaxProvider`, реализованный в клиентском ядре.
2. Вызовите пользовательский веб-сервис из модуля `ServiceHelper`.

   Способы вызова пользовательского веб-сервиса:

   - Вызовите метод `callService(serviceName, serviceMethodName, callback, serviceData, scope)`.
   - Вызовите метод `callService(config)`, где `config` — конфигурационный объект со свойствами:

     `serviceName` — имя пользовательского веб-сервиса.

     `methodName` — имя вызываемого метода пользовательского сервиса.

     `callback` — функция обратного вызова, в которой выполняется обработка ответа от веб-сервиса.

     `data` — объект с проинициализированными входящими параметрами для метода веб-сервиса.

     `scope` — контекст выполнения запроса.

Важно. Модуль `ServiceHelper` работает только с `POST`-запросами. Поэтому к методам

пользовательского веб-сервиса необходимо добавить атрибут `[WebInvoke]` с параметром `Method = "POST"`.

Перенести существующий пользовательский веб-сервис на платформу .NET Core

Пользовательский веб-сервис, который был разработан на платформе .NET Framework и получает контекст без наследования базового класса `Terrasoft.Web.Common.BaseService`, можно перенести на платформу .NET Core. Для переноса необходимо **выполнить адаптацию пользовательского веб-сервиса**.

Свойство `HttpContextAccessor` класса `Terrasoft.Web.Common.BaseService` обеспечивает унифицированный доступ к контексту (`HttpContext`) в .NET Framework и .NET Core. Свойства `UserConnection` и `AppConnection` позволяют получить объект пользовательского подключения и объект подключения на уровне приложения. Это позволяет отказаться от использования свойства `HttpContext.Current` библиотеки `System.Web`.

Пример использования свойств родительского класса `Terrasoft.Web.Common.BaseService`

```
namespace Terrasoft.Configuration.UsrCustomNamespace
{
    using Terrasoft.Web.Common;

    [ServiceContract]
    [AspNetCompatibilityRequirements(RequirementsMode = AspNetCompatibilityRequirementsMode.Requ
    public class UsrCustomConfigurationService: BaseService
    {
        /* Метод веб-сервиса. */
        [OperationContract]
        [WebInvoke(Method = "GET", RequestFormat = WebMessageFormat.Json, BodyStyle = WebMessage
        ResponseFormat = WebMessageFormat.Json)]
        public void SomeMethod() {
            ...
            /* UserConnection — свойство BaseService. */
            var currentUser = UserConnection.CurrentUser;
            /* AppConnection — свойство BaseService. */
            var sdkHelpUrl = AppConnection.SdkHelpUrl;
            /* HttpContextAccessor — свойство BaseService. */
            var httpContext = HttpContextAccessor.GetInstance();
            ...
        }
    }
}
```

Для веб-сервиса, который был разработан без наследования класса `Terrasoft.Web.Common.BaseService`

реализованы следующие **способы получения контекста**:

- Через `IHttpContextAccessor`, зарегистрированный в `DI` (`ClassFactory`).

  Этот способ позволяет покрывать код тестами и отображает явные зависимости класса. Подробнее об использовании фабрики классов можно узнать из статьи Принцип замещения классов.

- Через статическое свойство `HttpContext.Current`.

  В исходный код с помощью директивы `using` необходимо добавить пространство имен `Terrasoft.Web.Http.Abstractions`. Статическое свойство `HttpContext.Current` реализует унифицированный доступ к `HttpContext`. Для адаптации кода веб-сервиса для платформы .NET Core замените пространство имен `System.Web` на `Terrasoft.Web.Http.Abstractions`.

Важно. В конфигурации запрещено использовать конкретную реализацию доступа к контексту запроса из .NET Framework (библиотека `System.Web`) или .NET Core (библиотека `Microsoft.AspNetCore.Http`).

Пример адаптации веб-сервиса к платформе .NET Core

```
namespace Terrasoft.Configuration.UsrCustomNamespace
{
    /* Использовать вместо System.Web. */
    using Terrasoft.Web.Http.Abstractions;

    [ServiceContract]
    [AspNetCompatibilityRequirements(RequirementsMode = AspNetCompatibilityRequirementsMode.Requ
    public class UsrCustomConfigurationService
    {
        /* Метод веб-сервиса. */
        [OperationContract]
        [WebInvoke(Method = "GET", RequestFormat = WebMessageFormat.Json, BodyStyle = WebMessage
        ResponseFormat = WebMessageFormat.Json)]
        public void SomeMethod() {
            ...
            var httpContext = HttpContext.Current;
            ...
        }
    }
}
```

Разработать пользовательский веб-сервис с аутентификацией на основе cookies

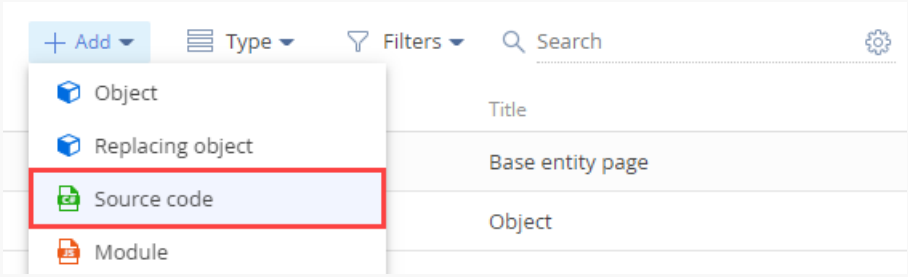
Средний

Пример. Создать пользовательский веб-сервис с аутентификацией на основе cookies, который возвращает идентификатор контакта по указанному имени.

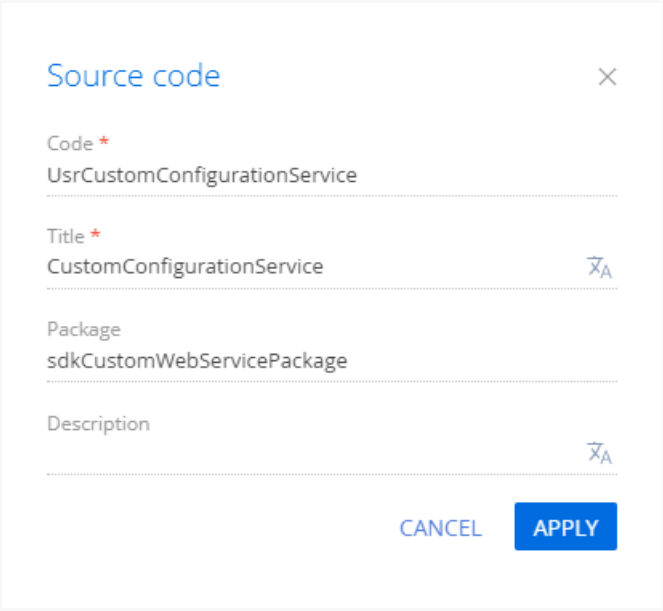
- Если контакт найден, то вернуть идентификатор контакта.
- Если найденных контактов несколько, то вернуть идентификатор первого найденного контакта.
- Если контакт найден, то вернуть пустую строку.

1. Создать схему [*Исходный код*]

1. Перейдите в раздел [*Конфигурация*] ([*Configuration*]) и выберите пользовательский пакет, в который будет добавлена схема.
2. На панели инструментов реестра раздела нажмите [*Добавить*] —> [*Исходный код*] ([*Add*] —> [*Source code*]).



3. В дизайнере схем заполните свойства схемы:
   - [*Код*] ([*Code*]) — "`UsrCustomConfigurationService`".
   - [*Заголовок*] ([*Title*]) — "`CustomConfigurationService`".



Для применения заданных свойств нажмите [*Применить*] ([*Apply*]).

2. Создать класс сервиса

1. В дизайнере схем добавьте пространство имен, вложенное в `Terrasoft.Configuration`. Название может быть любым, например, `UsrCustomConfigurationServiceNamespace`.
2. С помощью директивы `using` добавьте пространства имен, типы данных которых будут задействованы в классе.
3. Добавьте название класса, которое соответствует названию схемы (свойство [*Код*] ([*Code*])).
4. В качестве родительского класса укажите класс `Terrasoft.Nui.ServiceModel.WebService.BaseService`.
5. Для класса добавьте атрибуты `[ServiceContract]` и `[AspNetCompatibilityRequirements(RequirementsMode = AspNetCompatibilityRequirementsMode.Required)]`.

3. Реализовать метод класса

В дизайнере схем добавьте в класс метод `public string GetContactIdByName(string Name)`, который реализует конечную точку пользовательского веб-сервиса. С помощью `EntitySchemaQuery` метод отправит запрос к базе данных. В зависимости от значения параметра `Name`, отправленного в строке запроса, тело ответа на запрос будет содержать:

- Идентификатор контакта (типа строка) — если контакт найден.
- Идентификатор первого найденного контакта (типа строка) — если найдено несколько контактов.
- Пустую строку — если контакт не найден.

Исходный код пользовательского веб-сервиса `UsrCustomConfigurationService` представлен ниже.

`UsrCustomConfigurationService`

```
namespace Terrasoft.Configuration.UsrCustomConfigurationServiceNamespace
{
    using System;
    using System.ServiceModel;
    using System.ServiceModel.Web;
    using System.ServiceModel.Activation;
    using Terrasoft.Core;
    using Terrasoft.Web.Common;
    using Terrasoft.Core.Entities;

    [ServiceContract]
    [AspNetCompatibilityRequirements(RequirementsMode = AspNetCompatibilityRequirementsMode.Requ
    public class UsrCustomConfigurationService: BaseService
    {

        /* Метод, возвращающий идентификатор контакта по имени контакта. */
        [OperationContract]
        [WebInvoke(Method = "GET", RequestFormat = WebMessageFormat.Json, BodyStyle = WebMessage
        ResponseFormat = WebMessageFormat.Json)]
        public string GetContactIdByName(string Name) {
            /* Результат по умолчанию. */
            var result = "";
            /* Экземпляр EntitySchemaQuery, обращающийся в таблицу Contact базы данных. */
            var esq = new EntitySchemaQuery(UserConnection.EntitySchemaManager, "Contact");
            /* Добавление колонок в запрос. */
            var colId = esq.AddColumn("Id");
            var colName = esq.AddColumn("Name");
            /* Фильтрация данных запроса. */
            var esqFilter = esq.CreateFilterWithParameters(FilterComparisonType.Equal, "Name", N
            esq.Filters.Add(esqFilter);
            /* Получение результата запроса. */
            var entities = esq.GetEntityCollection(UserConnection);
            /* Если данные получены. */
            if (entities.Count > 0)
            {
                /* Возвратить значение колонки "Id" первой записи результата запроса. */
                result = entities[0].GetColumnValue(colId.Name).ToString();
                /* Также можно использовать такой вариант:
                result = entities[0].GetTypedColumnValue<string>(colId.Name); */
            }
            /* Возвратить результат. */
            return result;
        }
    }
}
```

На панели инструментов дизайнера нажмите [*Сохранить*] ([*Save*]), а затем [*Опубликовать*] ([*Publish*]).

Результат выполнения примера

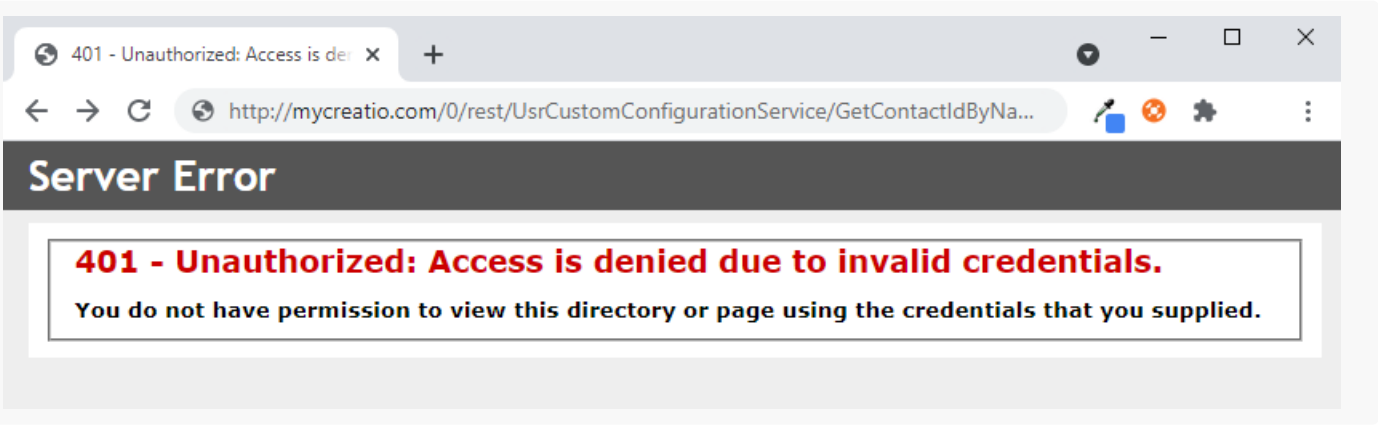
В результате выполнения примера в Creatio появится пользовательский веб-сервис `UsrCustomConfigurationService` типа REST с конечной точкой `GetContactIdByName`.

Из браузера обратитесь к конечной точке `GetContactIdByName` веб-сервиса и в параметре `Name` передайте имя контакта.

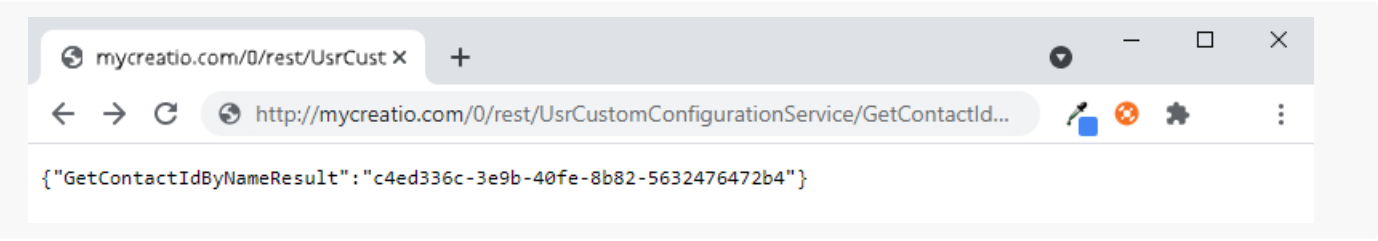
Строка запроса с именем существующего контакта

```
http://mycreatio.com/0/rest/UsrCustomConfigurationService/GetContactIdByName?Name=Andrew%20Baker
```

При обращении к веб-сервису без предварительной авторизации возникнет ошибка.



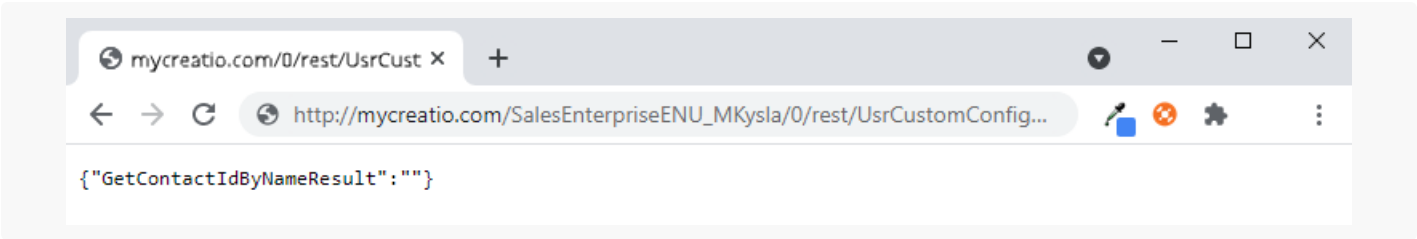
Авторизуйтесь в приложении и выполните запрос еще раз. Если контакт, указанный в параметре `Name`, найден в базе данных, то в свойстве `GetContactIdByNameResult` будет возвращено значение идентификатора контакта.



Если контакт, указанный в параметре `Name`, не найден в базе данных, то в свойстве `GetContactIdByNameResult` будет возвращена пустая строка.

Строка запроса с именем несуществующего контакта

```
http://mycreatio.com/0/rest/UsrCustomConfigurationService/GetContactIdByName?Name=Andrew%20Bake
```



Разработать пользовательский веб-сервис с анонимной аутентификацией

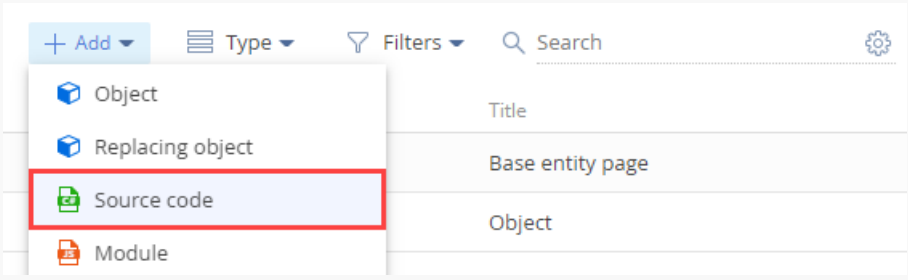
Средний

Пример. Создать пользовательский веб-сервис с анонимной аутентификацией, который возвращает идентификатор контакта по указанному имени.

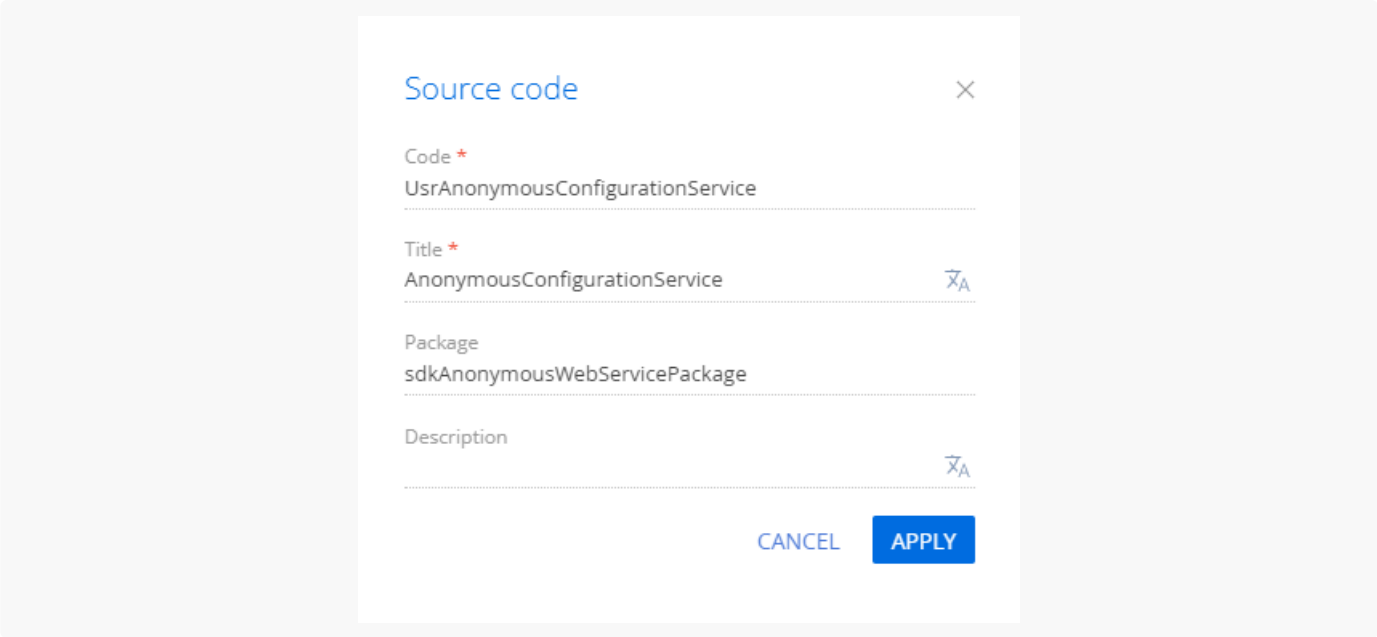
- Если контакт найден, то вернуть идентификатор контакта.
- Если найденных контактов несколько, то вернуть идентификатор первого найденного контакта.
- Если контакт найден, то вернуть пустую строку.

1. Создать схему [*Исходный код*]

1. Перейдите в раздел [*Конфигурация*] ([*Configuration*]) и выберите пользовательский пакет, в который будет добавлена схема.
2. На панели инструментов реестра раздела нажмите [*Добавить*] —> [*Исходный код*] ([*Add*] —> [*Source code*]).



3. В дизайнере схем заполните свойства схемы:
   - [*Код*] ([*Code*]) — "`UsrAnonymousConfigurationService`".
   - [*Заголовок*] ([*Title*]) — "`AnonymousConfigurationService`".



Для применения заданных свойств нажмите [*Применить*] ([*Apply*]).

2. Создать класс сервиса

1. В дизайнере схем добавьте пространство имен, вложенное в `Terrasoft.Configuration`. Название может быть любым, например, `UsrAnonymousConfigurationServiceNamespace`.
2. С помощью директивы `using` добавьте пространства имен, типы данных которых будут задействованы в классе.
3. Добавьте название класса, которое соответствует названию схемы (свойство [*Код*] ([*Code*])).
4. В качестве родительского класса укажите класс `Terrasoft.Nui.ServiceModel.WebService.BaseService`.
5. Для класса добавьте атрибуты `[ServiceContract]` и `[AspNetCompatibilityRequirements(RequirementsMode = AspNetCompatibilityRequirementsMode.Required)]`.
6. Для анонимного доступа к пользовательскому веб-сервису добавьте системное подключение `SystemUserConnection`.

3. Реализовать метод класса

В дизайнере схем добавьте в класс метод `public string GetContactIdByName(string Name)`, который реализует конечную точку пользовательского веб-сервиса. С помощью `EntitySchemaQuery` метод отправит запрос к базе данных. В зависимости от значения параметра `Name`, отправленного в строке запроса, тело ответа на запрос будет содержать:

- Идентификатор контакта (типа строка) — если контакт найден.
- Идентификатор первого найденного контакта (типа строка) — если найдено несколько контактов.
- Пустую строку — если контакт не найден.

Укажите пользователя, от имени которого будет выполняться обработка данного http-запроса. Для

этого после получения `SystemUserConnection` вызовите метод `SessionHelper.SpecifyWebOperationIdentity` пространства имен `Terrasoft.Web.Common`. Этот метод обеспечивает работоспособность бизнес-процессов при работе с сущностью (`Entity`) базы данных из пользовательского веб-сервиса с анонимной аутентификацией.

```
Terrasoft.Web.Common.SessionHelper.SpecifyWebOperationIdentity(HttpContextAccessor.GetInstance()
```

Исходный код пользовательского веб-сервиса `UsrAnonymousConfigurationService` представлен ниже.

`UsrAnonymousConfigurationService`

```
/* Пользовательское пространство имен. */
namespace Terrasoft.Configuration.UsrAnonymousConfigurationServiceNamespace
{
    using System;
    using System.ServiceModel;
    using System.ServiceModel.Web;
    using System.ServiceModel.Activation;
    using Terrasoft.Core;
    using Terrasoft.Web.Common;
    using Terrasoft.Core.Entities;

    [ServiceContract]
    [AspNetCompatibilityRequirements(RequirementsMode = AspNetCompatibilityRequirementsMode.Requ
    public class UsrAnonymousConfigurationService: BaseService
    {
        /* Ссылка на экземпляр UserConnection, требуемый для обращения к базе данных. */
        private SystemUserConnection _systemUserConnection;
        private SystemUserConnection SystemUserConnection {
            get {
                return _systemUserConnection ?? (_systemUserConnection = (SystemUserConnection)A
            }
        }

        /* Метод, возвращающий идентификатор контакта по имени контакта. */
        [OperationContract]
        [WebInvoke(Method = "GET", RequestFormat = WebMessageFormat.Json, BodyStyle = WebMessage
        ResponseFormat = WebMessageFormat.Json)]
        public string GetContactIdByName(string Name){
            /* Указывается пользователь, от имени которого выполняется обработка данного http-за
            SessionHelper.SpecifyWebOperationIdentity(HttpContextAccessor.GetInstance(), SystemU
            /* Результат по умолчанию. */
            var result = "";
            /* Экземпляр EntitySchemaQuery, обращающийся в таблицу Contact базы данных. */
            var esq = new EntitySchemaQuery(SystemUserConnection.EntitySchemaManager, "Contact")
            /* Добавление колонок в запрос. */
```

```
            var colId = esq.AddColumn("Id");
            var colName = esq.AddColumn("Name");
            //* Фильтрация данных запроса. */
            var esqFilter = esq.CreateFilterWithParameters(FilterComparisonType.Equal, "Name", N
            esq.Filters.Add(esqFilter);
            /* Получение результата запроса. */
            var entities = esq.GetEntityCollection(SystemUserConnection);
            /* Если данные получены. */
            if (entities.Count > 0)
            {
                /* Возвратить значение колонки "Id" первой записи результата запроса. */
                result = entities[0].GetColumnValue(colId.Name).ToString();
                /* Также можно использовать такой вариант:
                result = entities[0].GetTypedColumnValue<string>(colId.Name); */
            }
            /* Возвратить результат. */
            return result;
        }
    }
}
```

На панели инструментов дизайнера нажмите [*Сохранить*] ([*Save*]), а затем [*Опубликовать*] ([*Publish*]).

4. Зарегистрировать пользовательский веб-сервис с анонимной аутентификацией

1. Перейдите в каталог `..\Terrasoft.WebApp\ServiceModel`.
2. Создайте файл `UsrAnonymousConfigurationService.svc` и добавьте в него запись.

```
<% @ServiceHost
    Service = "Terrasoft.Configuration.UsrAnonymousConfigurationServiceNamespace.UsrAnonymous
    Debug = "true"
    Language = "C#"
%>
```

Атрибут `Service` содержит полное имя класса веб-сервиса с указанием пространства имен.

5. Настроить пользовательский веб-сервис с анонимной аутентификацией для работы по протоколам http и https

1. Откройте файл `..\Terrasoft.WebApp\ServiceModel\http\services.config` и добавьте в него запись.

Файл `..\Terrasoft.WebApp\ServiceModel\http\services.config`

```
<services>
    ...
    <service name="Terrasoft.Configuration.UsrAnonymousConfigurationServiceNamespace.UsrAnony
        <endpoint name="[Service name]EndPoint"
            address=""
            binding="webHttpBinding"
            behaviorConfiguration="RestServiceBehavior"
            bindingNamespace="http://Terrasoft.WebApp.ServiceModel"
            contract="Terrasoft.Configuration.UsrAnonymousConfigurationServiceNamespace.UsrAn
    </service>
</services>
```

2. Аналогичную запись добавьте в файл `..\Terrasoft.WebApp\ServiceModel\https\services.config`.

6. Настроить доступ к пользовательскому веб-сервису с анонимной аутентификацией для всех пользователей

1. Откройте файл `..\Terrasoft.WebApp\Web.config`.
2. Добавьте элемент `<location>`, определяющий относительный путь и права доступа к веб-сервису.

Файл `..\Terrasoft.WebApp\Web.config`

```
<configuration>
    ...
    <location path="ServiceModel/UsrAnonymousConfigurationService.svc">
        <system.web>
            <authorization>
                <allow users="*" />
            </authorization>
        </system.web>
    </location>
    ...
</configuration>
```

3. В атрибут `value` ключа `AllowedLocations` элемента `<appSettings>` добавьте относительный путь к веб-сервису.

Файл `..\Terrasoft.WebApp\Web.config`

```
<configuration>
    ...
    <appSettings>
    ...
        <add key="AllowedLocations" value="[Предыдущие значения];ServiceModel/UsrAnonymousCon
```

```
    ...
    </appSettings>
    ...
</configuration>
```

7. Перезапустить приложение в IIS

Для применения изменений перезапустите приложение в IIS.

Результат выполнения примера

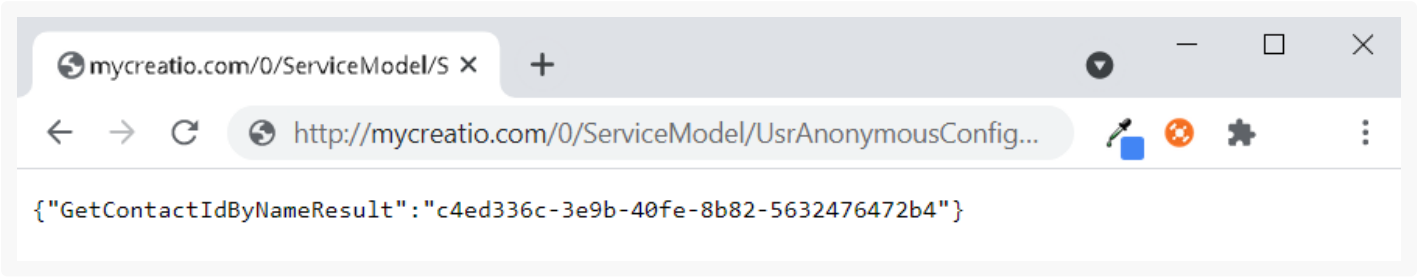
В результате выполнения примера в Creatio появится пользовательский веб-сервис `UsrAnonymousConfigurationService` с конечной точкой `GetContactIdByName`. К веб-сервису можно обращаться из браузера, как с предварительным вводом логина и пароля, так и без их использования.

Из браузера обратитесь к конечной точке `GetContactIdByName` веб-сервиса и в параметре `Name` передайте имя контакта.

Строка запроса с именем существующего контакта

```
http://mycreatio.com/0/ServiceModel/UsrAnonymousConfigurationService/GetContactIdByName?Name=Anc
```

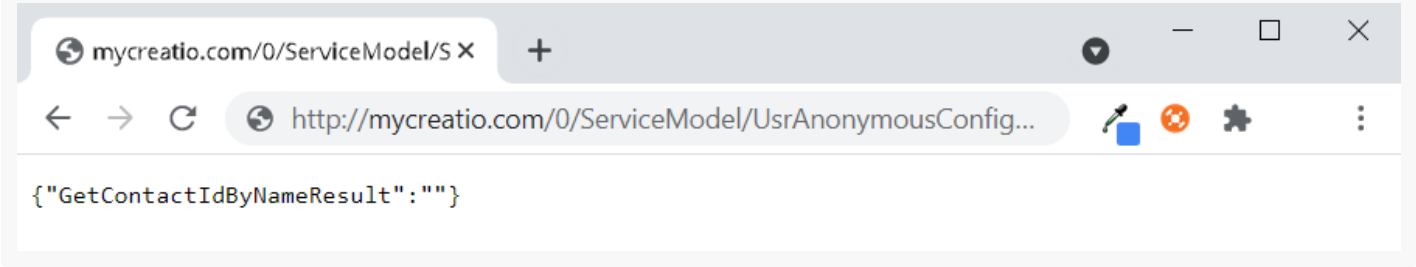
Если контакт, указанный в параметре `Name`, найден в базе данных, то в свойстве `GetContactIdByNameResult` будет возвращено значение идентификатора контакта.



Если контакт, указанный в параметре `Name`, не найден в базе данных, то в свойстве `GetContactIdByNameResult` будет возвращена пустая строка.

Строка запроса с именем несуществующего контакта

```
http://mycreatio.com/0/ServiceModel/UsrAnonymousConfigurationService/GetContactIdByName?Name=Anc
```



Вызвать пользовательский веб-сервис из front-end части

Средний

Пример. На страницу добавления нового контакта добавить кнопку, по клику на которую будет вызываться пользовательский веб-сервис. В информационном окне страницы отобразить результат, возвращаемый методом веб-сервиса.

1. Создать пользовательский веб-сервис

В примере используется пользовательский веб-сервис `UsrCustomConfigurationService`, разработка которого описана в статье Разработать пользовательский веб-сервис с аутентификацией на основе cookies.

В веб-сервисе `UsrCustomConfigurationService` измените параметр `Method` атрибута `WebInvoke` на `POST`.

Исходный код пользовательского веб-сервиса, используемого в примере, представлен ниже.

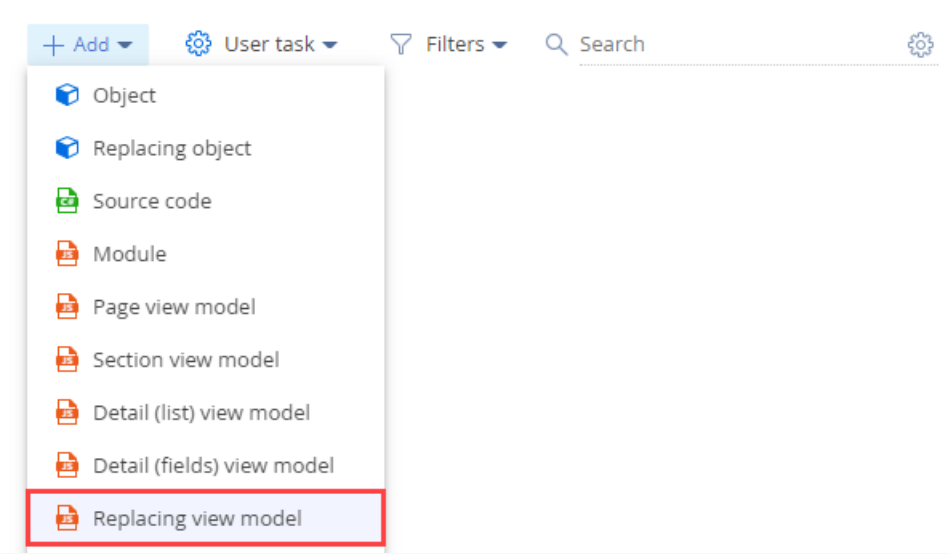
UsrCustomConfigurationService

```
namespace Terrasoft.Configuration.UsrCustomConfigurationServiceNamespace
{
    using System;
    using System.ServiceModel;
    using System.ServiceModel.Web;
    using System.ServiceModel.Activation;
    using Terrasoft.Core;
    using Terrasoft.Web.Common;
    using Terrasoft.Core.Entities;

    [ServiceContract]
    [AspNetCompatibilityRequirements(RequirementsMode = AspNetCompatibilityRequirementsMode.Requ
    public class UsrCustomConfigurationService: BaseService
    {

        /* Метод, возвращающий идентификатор контакта по имени контакта. */
```

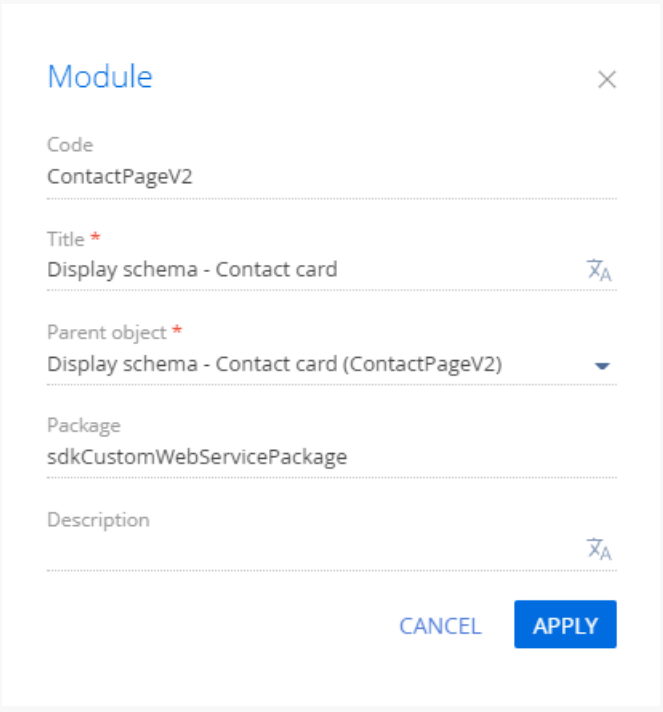
```
        [OperationContract]
        [WebInvoke(Method = "POST", RequestFormat = WebMessageFormat.Json, BodyStyle = WebMessag
        ResponseFormat = WebMessageFormat.Json)]
        public string GetContactIdByName(string Name) {
            /* Результат по умолчанию. */
            var result = "";
            /* Экземпляр EntitySchemaQuery, обращающийся в таблицу Contact базы данных. */
            var esq = new EntitySchemaQuery(UserConnection.EntitySchemaManager, "Contact");
            /* Добавление колонок в запрос. */
            var colId = esq.AddColumn("Id");
            var colName = esq.AddColumn("Name");
            /* Фильтрация данных запроса. */
            var esqFilter = esq.CreateFilterWithParameters(FilterComparisonType.Equal, "Name", N
            esq.Filters.Add(esqFilter);
            /* Получение результата запроса. */
            var entities = esq.GetEntityCollection(UserConnection);
            /* Если данные получены. */
            if (entities.Count > 0)
            {
                /* Возвратить значение колонки "Id" первой записи результата запроса. */
                result = entities[0].GetColumnValue(colId.Name).ToString();
                /* Также можно использовать такой вариант:
                result = entities[0].GetTypedColumnValue<string>(colId.Name); */
            }
            /* Возвратить результат. */
            return result;
        }
    }
}
```

2. Создать замещающую страницу записи контакта

1. Перейдите в раздел [*Конфигурация*] ([*Configuration*]) и выберите пользовательский пакет, в который будет добавлена схема.
2. На панели инструментов реестра раздела нажмите [*Добавить*] —> [*Замещающая модель представления*] ([*Add*] —> [*Replacing view model*]).



3. В свойстве [*Родительский объект*] ([*Parent object*]) выберите схему модели представления [*Схема отображения карточки контакта*] ([*Display schema — Contact card*]) пакета `ContactPageV2`, которую необходимо заместить. После подтверждения выбранного родительского объекта остальные свойства будут заполнены автоматически.



4. В объявлении модуля страницы записи в качестве зависимости подключите модуль `ServiceHelper`. Зависимости модуля описаны в статье Функция define().

3. Добавить кнопку на страницу записи контакта

1. В блоке [*Локализуемые строки*] ([*Localizable strings*]) панели свойств нажмите кнопку + и заполните **свойства локализуемой строки**:

- [*Код*] ([*Code*]) — "GetServiceInfoButtonCaption".
- [*Значение*] ([*Value*]) — "Вызвать сервис" ("Call service").

2. Добавьте обработчик кнопки.

   Для вызова веб-сервиса воспользуйтесь методом `callService()` модуля `ServiceHelper`. **Параметры функции** `callService()`, которые необходимо передать:

   - `UsrCustomConfigurationService` — имя класса пользовательского веб-сервиса.
   - `GetContactIdByName` — имя вызываемого метода пользовательского веб-сервиса.
   - Функцию обратного вызова, в которой выполните обработку результатов сервиса.
   - `serviceData` — объект с проинициализированными входящими параметрами для метода пользовательского веб-сервиса.
   - Контекст выполнения.

   Исходный код схемы замещающей модели представления `ContactPageV2` представлен ниже.

ContactPageV2

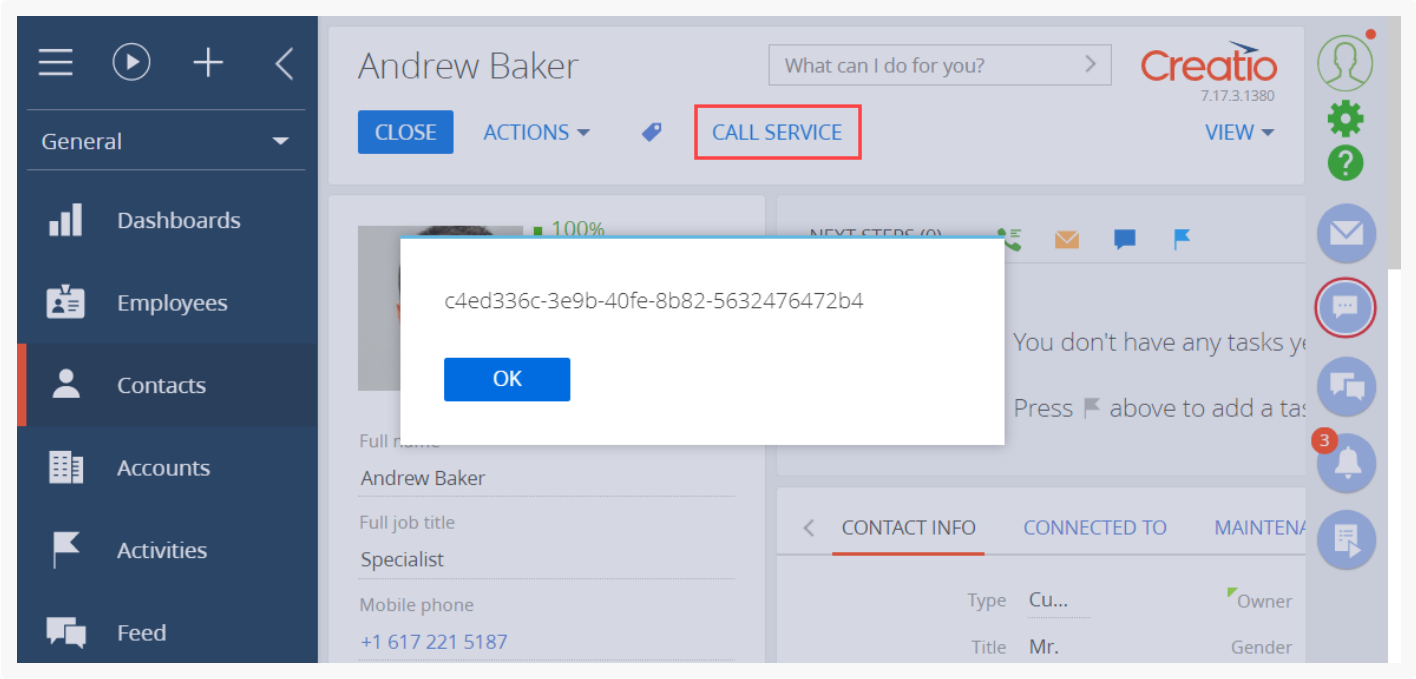
```
define("ContactPageV2", ["ServiceHelper"],
function(ServiceHelper) {
    return {
        /* Название схемы объекта страницы записи. */
        entitySchemaName: "Contact",
        details: /**SCHEMA_DETAILS*/{}/**SCHEMA_DETAILS*/,
        /* Методы модели представления страницы записи. */
        methods: {
            /* Проверяет, заполнено ли поле [ФИО] страницы. */
            isContactNameSet: function() {
                return this.get("Name") ? true : false;
            },
            /* Метод-обработчик нажатия кнопки. */
            onGetServiceInfoClick: function() {
                var name = this.get("Name");
                /* Объект, инициализирующий входящие параметры для метода сервиса. */
                var serviceData = {
                    /* Название свойства совпадает с именем входящего параметра метода сервис
                    Name: name
                };
                /*/ Вызов веб-сервиса и обработка результатов. */
                ServiceHelper.callService("UsrCustomConfigurationService", "GetContactIdByNam
                    function(response) {
                        var result = response.GetContactIdByNameResult;
                        this.showInformationDialog(result);
                    }, serviceData, this);
            }
        },
```

```
        diff: /**SCHEMA_DIFF*/[
            /* Метаданные для добавления на страницу пользовательской кнопки. */
            {
                /* Выполняется операция добавления элемента на страницу. */
                "operation": "insert",
                /* Имя родительского элемента управления, в который добавляется кнопка. */
                "parentName": "LeftContainer",
                /* Кнопка добавляется в коллекцию элементов управления родительского элемента
                "propertyName": "items",
                /* Имя добавляемой кнопки. */
                "name": "GetServiceInfoButton",
                /* Дополнительные свойства поля. */
                "values": {
                    /* Тип добавляемого элемента - кнопка. */
                    itemType: Terrasoft.ViewItemType.BUTTON,
                    /* Привязка заголовка кнопки к локализуемой строке схемы. */
                    caption: {bindTo: "Resources.Strings.GetServiceInfoButtonCaption"},
                    /* Привязка метода-обработчика нажатия кнопки. */
                    click: {bindTo: "onGetServiceInfoClick"},
                    /* Привязка свойства доступности кнопки. */
                    enabled: {bindTo: "isContactNameSet"},
                    /* Настройка расположения поля. */
                    "layout": {"column": 1, "row": 6, "colSpan": 2, "rowSpan": 1}
                }
            }
        ]/**SCHEMA_DIFF*/
    };
});
```

3. На панели инструментов дизайнера нажмите [*Сохранить*] ([*Save*]).

Результат выполнения примера

В результате выполнения примера после обновления страницы приложения на странице контакта появится кнопка [*Вызвать сервис*] ([*Call service*]). При нажатии на эту кнопку вызывается метод `GetContactIdByName` пользовательского веб-сервиса `UsrCustomConfigurationService`, который возвращает значение идентификатора текущего контакта.



Вызвать пользовательский веб-сервис с помощью Postman

Средний

Для интеграции внешних приложений с пользовательскими веб-сервисами Creatio необходимо выполнять HTTP-запросы к этим сервисам. Для понимания принципа формирования запросов удобно использовать такие инструменты редактирования и отладки, как Postman или Fiddler.

Postman — это набор инструментов для тестирования запросов. **Назначение** Postman — тестирование отправки запросов с клиента на сервер и получения ответа от сервера. В этой статье рассмотрен пример вызова пользовательского веб-сервиса с аутентификацией на основе cookies с помощью Postman.

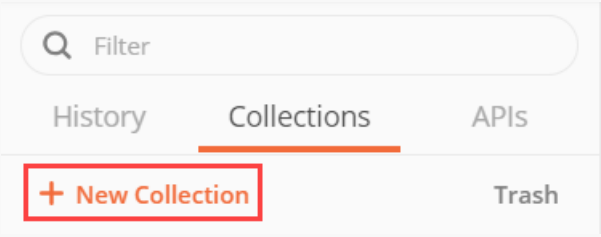
Пример. Вызвать пользовательский веб-сервис с аутентификацией на основе cookies с помощью Postman.

В примере используется пользовательский веб-сервис `UsrCustomConfigurationService`, разработка которого описана в статье Разработать пользовательский веб-сервис с аутентификацией на основе cookies.

Поскольку в примере используется пользовательский веб-сервис с аутентификацией на основе cookies, то предварительно необходимо выполнить аутентификацию в приложении, выполнив запрос к системному веб-сервису `AuthService.svc`. Описание аутентификации содержится в статье Аутентификация.

1. Создать коллекцию запросов

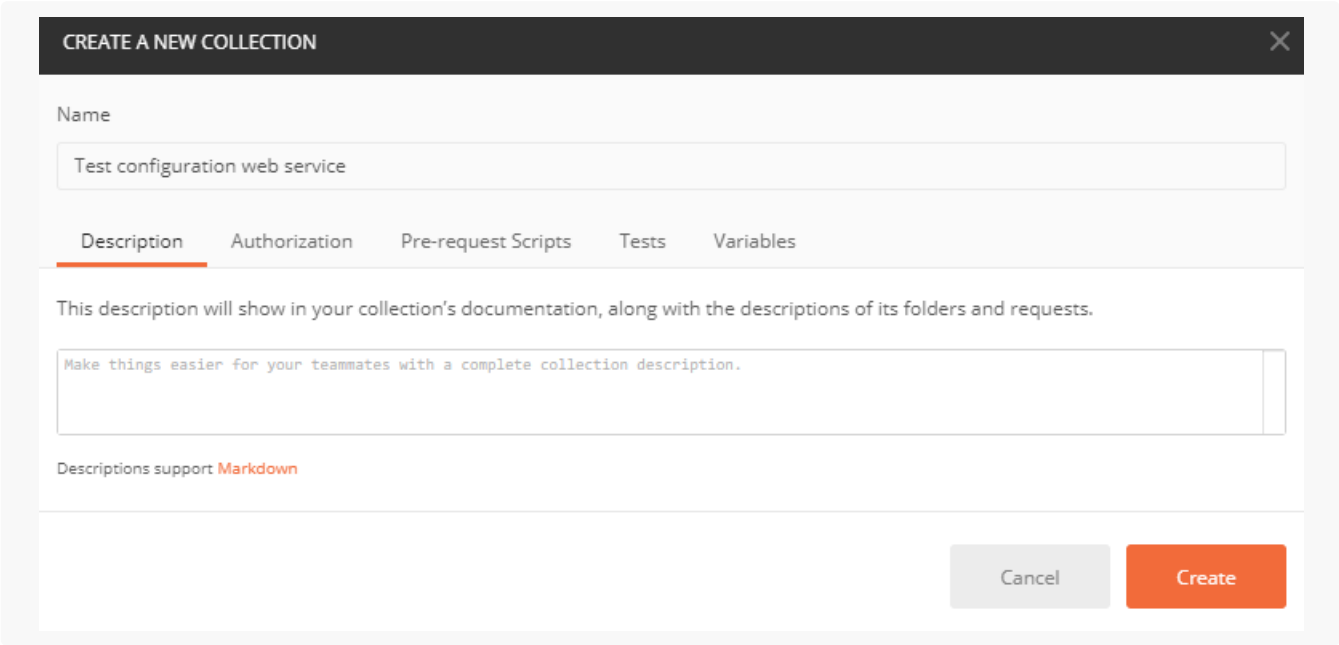
1. На панели работы с запросами в Postman перейдите на вкладку [*Collections*] и нажмите [*+ New Collection*].



2. Заполните **поля коллекции запросов**:

   - [*Name*] — "`Test configuration web service`".

Окно создания коллекции запросов



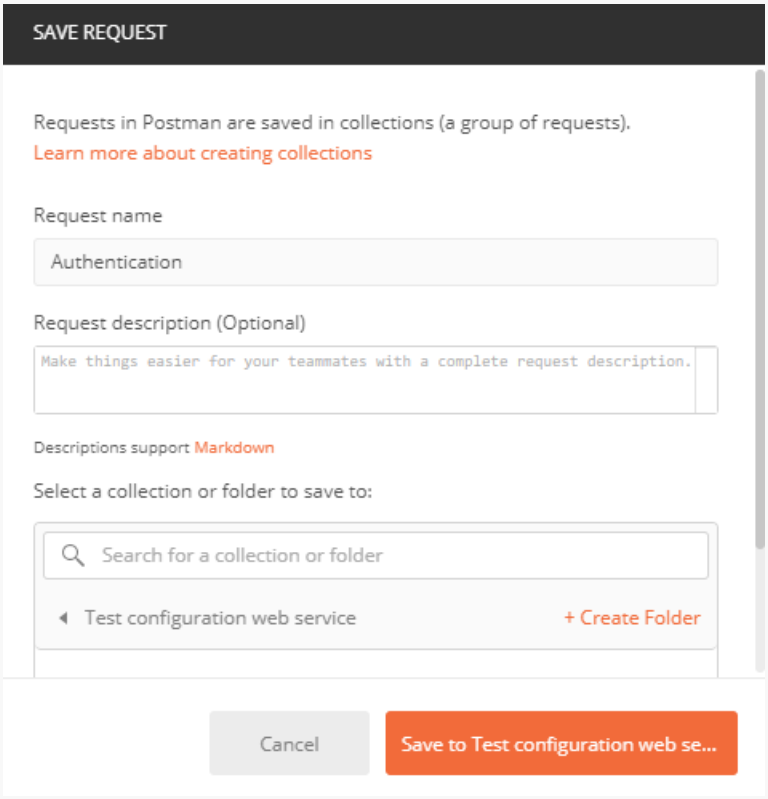
3. Чтобы создать коллекцию запросов, нажмите [*Create*].

2. Настроить аутентификационный запрос

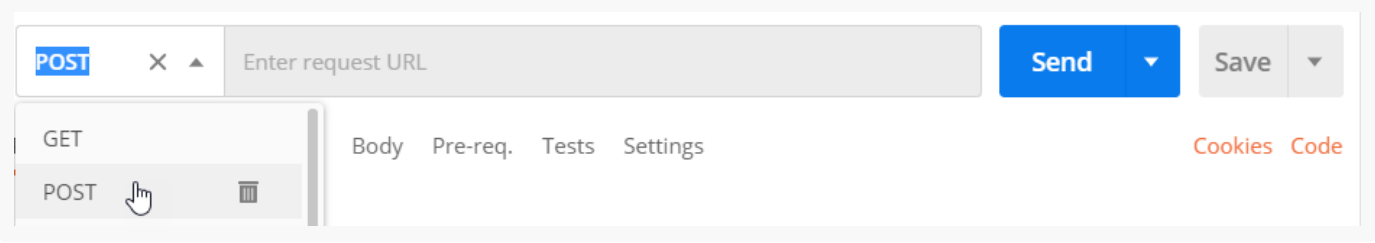
1. На панели работы с запросами в Postman правой кнопкой мыши кликните по имени коллекции `Test configuration web service` —> [*Add request*].
2. Заполните **поля запроса**:

   - [*Request name*] — "`Authentication`".

Окно создания запроса



3. Чтобы добавить запрос в коллекцию, нажмите [*Save to Test configuration web service*].
4. В выпадающем списке панели инструментов рабочей области Postman выберите метод запроса `POST`.



5. В поле запроса панели инструментов рабочей области Postman введите строку запроса к сервису аутентификации.

Шаблон адреса сервиса AuthService.svc

```
[Адрес приложения Creatio]/ServiceModel/AuthService.svc/Login
```

Пример адреса сервиса AuthService.svc

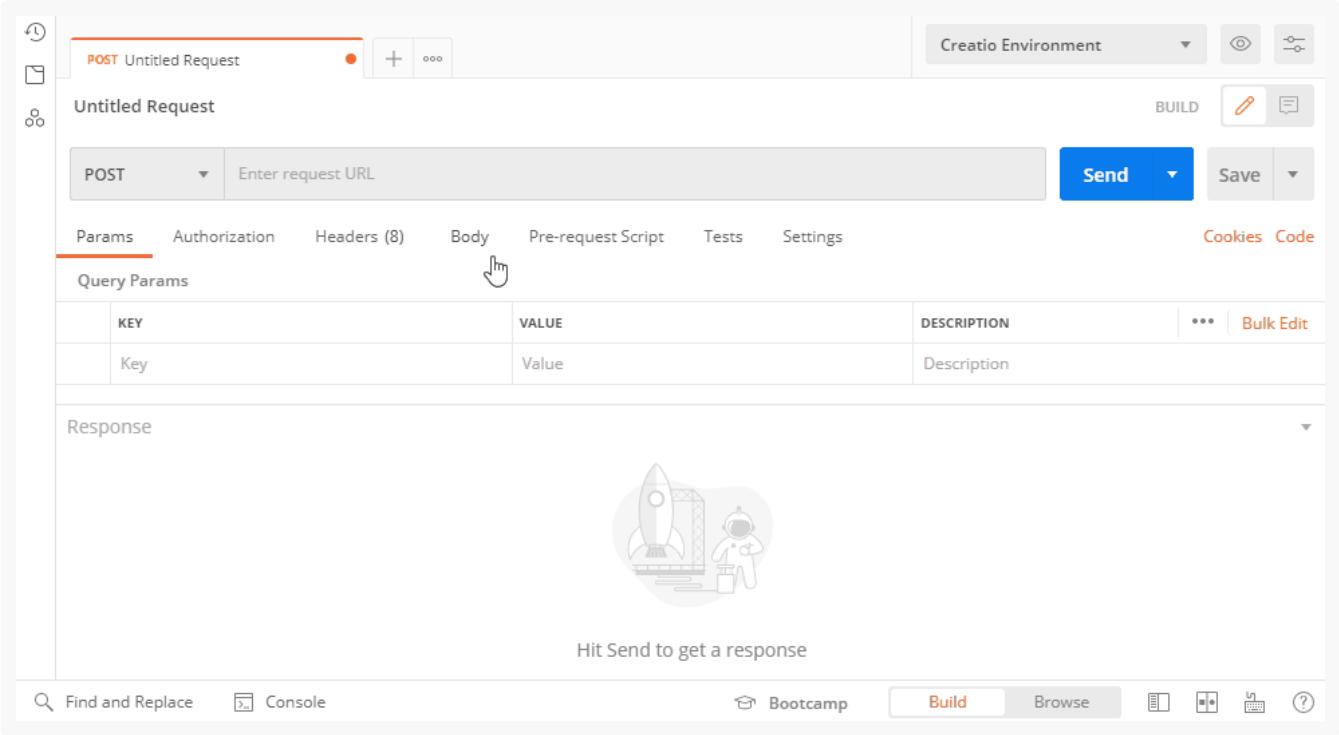
```
http://mycreatio.com/creatio/ServiceModel/AuthService.svc/Login
```

6. Установите **формат данных запроса**:

a. Перейдите на вкладку [*Body*].

b. Установите опцию "`raw`".

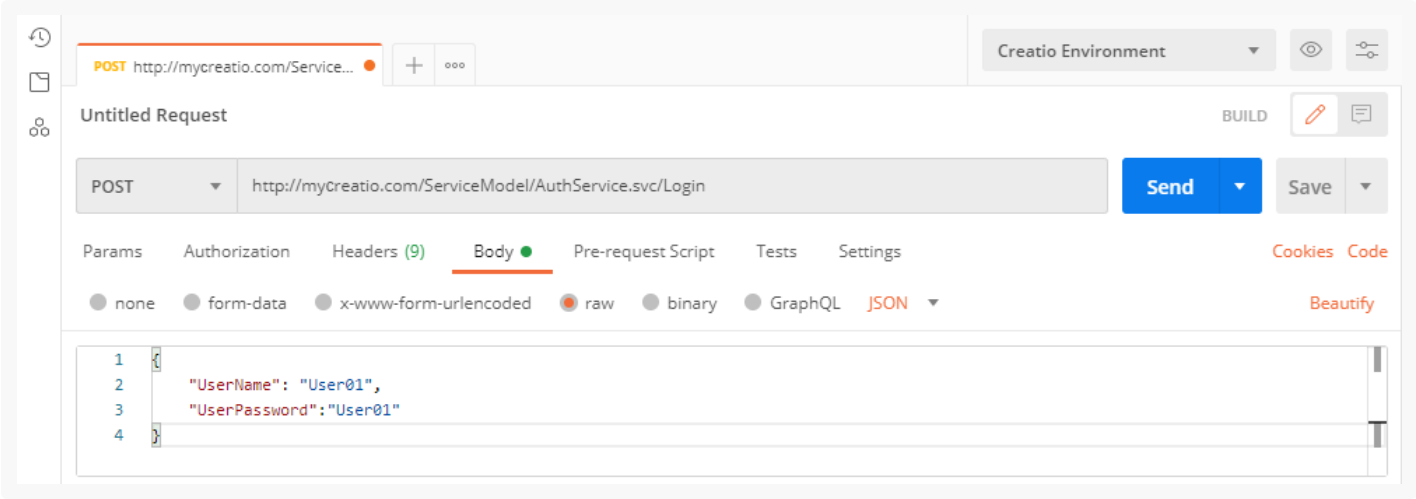
c. Выберите тип "`JSON`".



7. В рабочей области Postman перейдите на вкладку [*Body*] и заполните тело `POST`-запроса — JSON-объект, который содержит аутентификационные данные (логин и пароль).

Тело `POST`-запроса

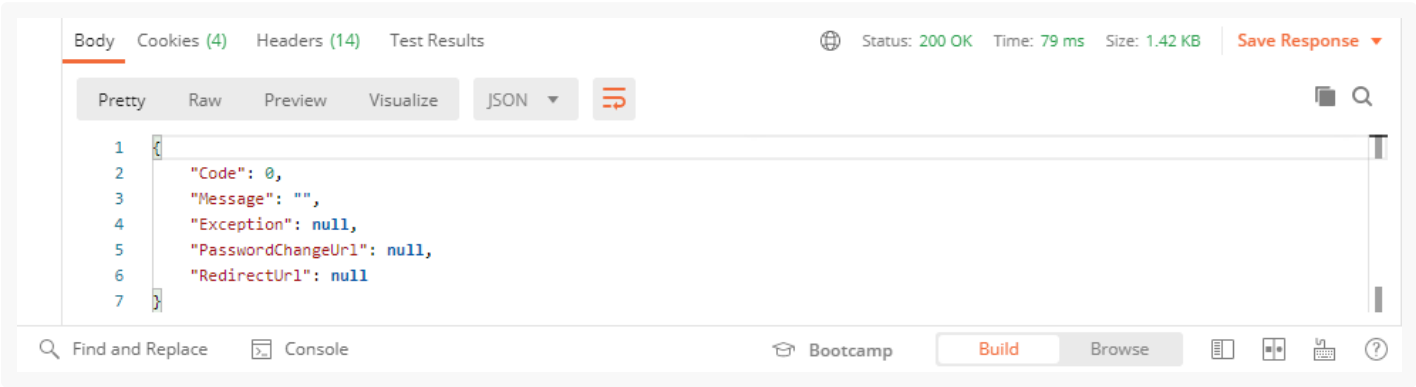
```
{
    "UserName": "User01",
    "UserPassword":"User01"
}
```



3. Выполнить аутентификационный запрос

Чтобы **выполнить запрос в Postman**, на панели инструментов рабочей области нажмите [*Send*].

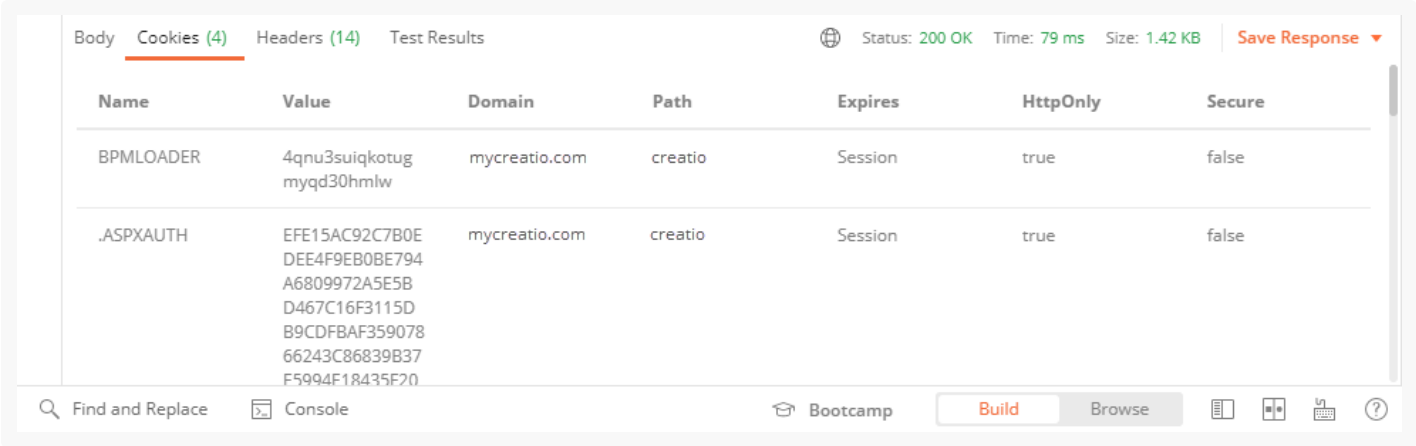
В результате выполнения запроса будет получен ответ, который содержит JSON-объект. Тело ответа отображается на вкладке `Body` в Postman.



Признаки **успешного выполнения запроса**:

- Получен код состояния ● **200 OK**.
- Параметр `Code` тела ответа содержит значение "`0`".

Ответ на запрос также содержит cookie `BPMLOADER`, `.ASPXAUTH`, `BPMCSRF` и `UserName`, которые отображаются на вкладке `Cookies`, а также продублированы на вкладке `Headers` в Postman.



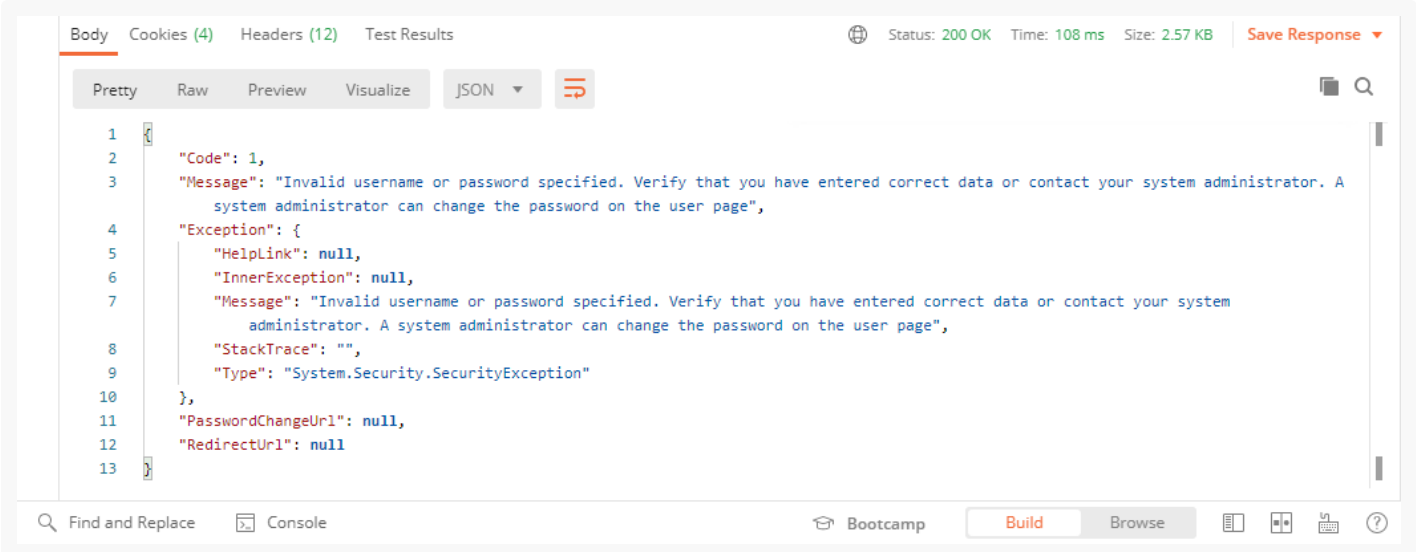
Эти cookie необходимо использовать в дальнейших запросах к сервисам Creatio, которые используют аутентификацию на основе cookies.

При включенной защите от CSRF-атак использование cookie `BPMCSRF` является обязательным. Если cookie `BPMCSRF` не будет использован, сервер вернет код состояния **403 Forbidden**. Для демо-сайтов Creatio использование cookie `BPMCSRF` необязательно, поскольку защита от CSRF-атак по умолчанию отключена.

Запрос выполняется неуспешно, если была допущена ошибка при построении запроса или тела запроса.

Признаки **неуспешного выполнения запроса**:

- Параметр `Code` тела ответа содержит значение "1".
- Параметр `Message` тела ответа содержит описание причины неуспешной аутентификации.



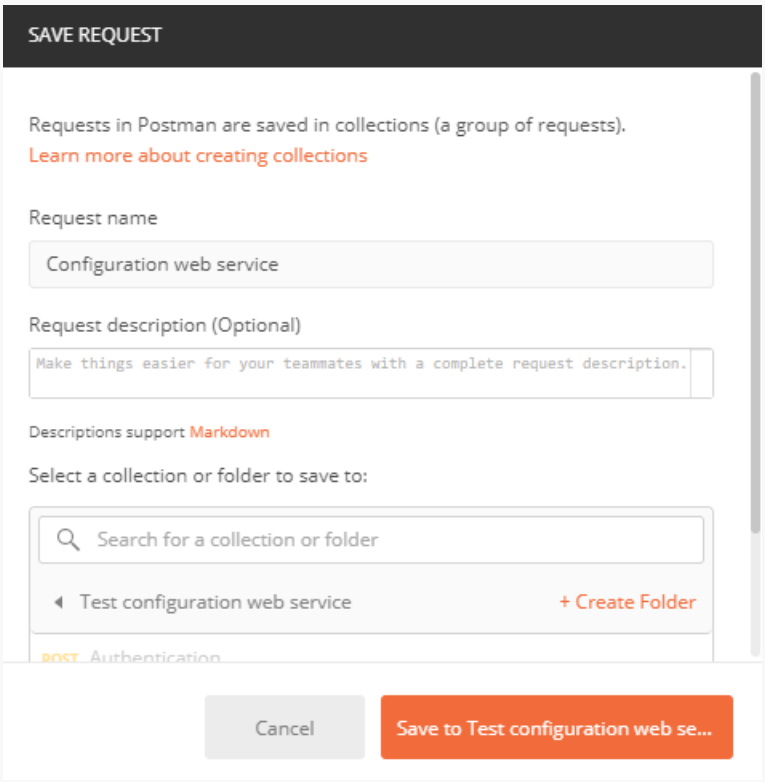
4. Настроить запрос к пользовательскому веб-сервису с аутентификацией на основе cookies

Пользовательский веб-сервис `UsrCustomConfigurationService` работает с запросами только по методу `GET`.

Чтобы **настроить запрос к пользовательскому веб-сервису с аутентификацией на основе cookies**:

1. На панели работы с запросами в Postman правой кнопкой мыши кликните по имени коллекции `Test configuration web service` —> [*Add request*].
2. Заполните **поля запроса**:
   - [*Request name*] — "`Configuration web service`".

Окно создания запроса



3. Чтобы добавить запрос в коллекцию, нажмите [*Save to Test configuration web service*].
4. По умолчанию в Postman выбран метод `GET`. В поле запроса панели инструментов рабочей области Postman введите строку запроса к пользовательскому веб-сервису `UsrCustomConfigurationService`.

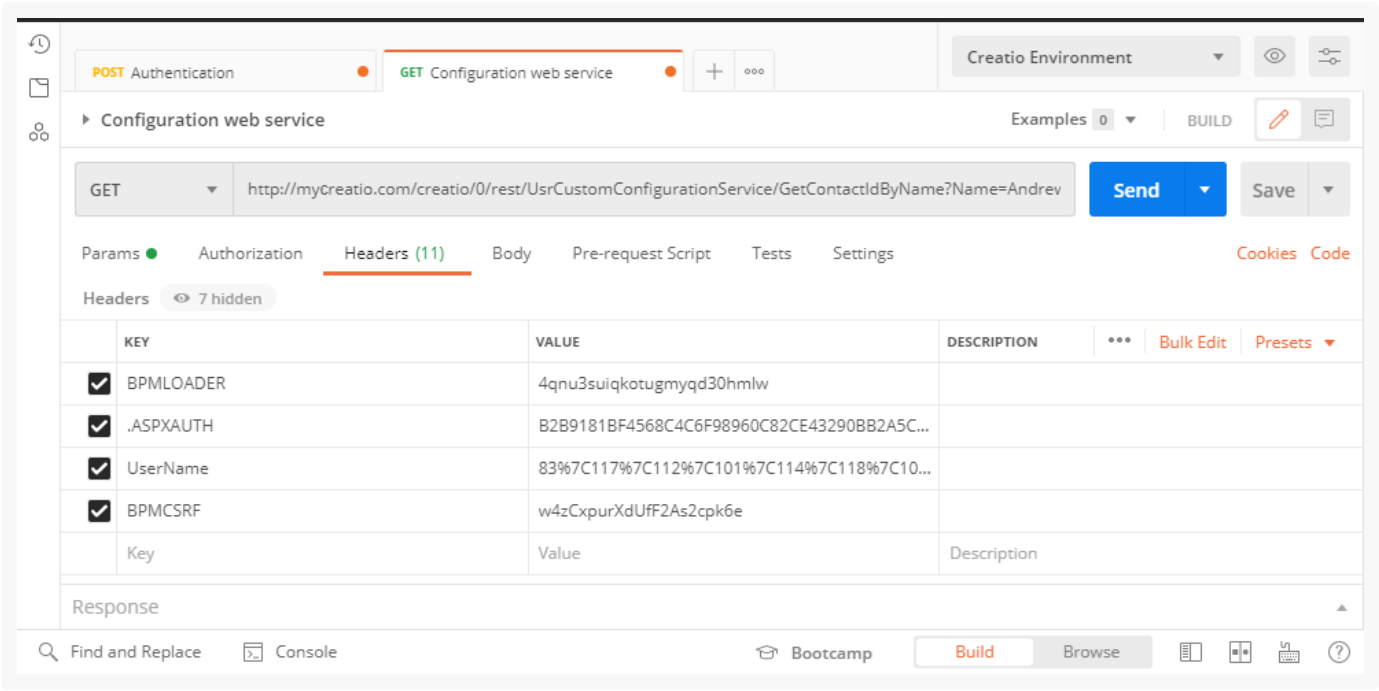
Шаблон адреса пользовательского веб-сервиса

```
[Адрес приложения Creatio]/0/rest/UsrCustomConfigurationService/GetContactIdByName?Name=[Имя
```

Пример адреса пользовательского веб-сервиса

```
http://mycreatio.com/creatio/0/rest/UsrCustomConfigurationService/GetContactIdByName?Name=And
```

5. В рабочей области Postman перейдите на вкладку [*Headers*] и в заголовки запроса к пользовательскому веб-сервису добавьте cookie, полученные в ответ на авторизационный запрос. В поле [*Key*] добавьте имя cookie, а в поле [*Value*] скопируйте соответствующее значение cookie.



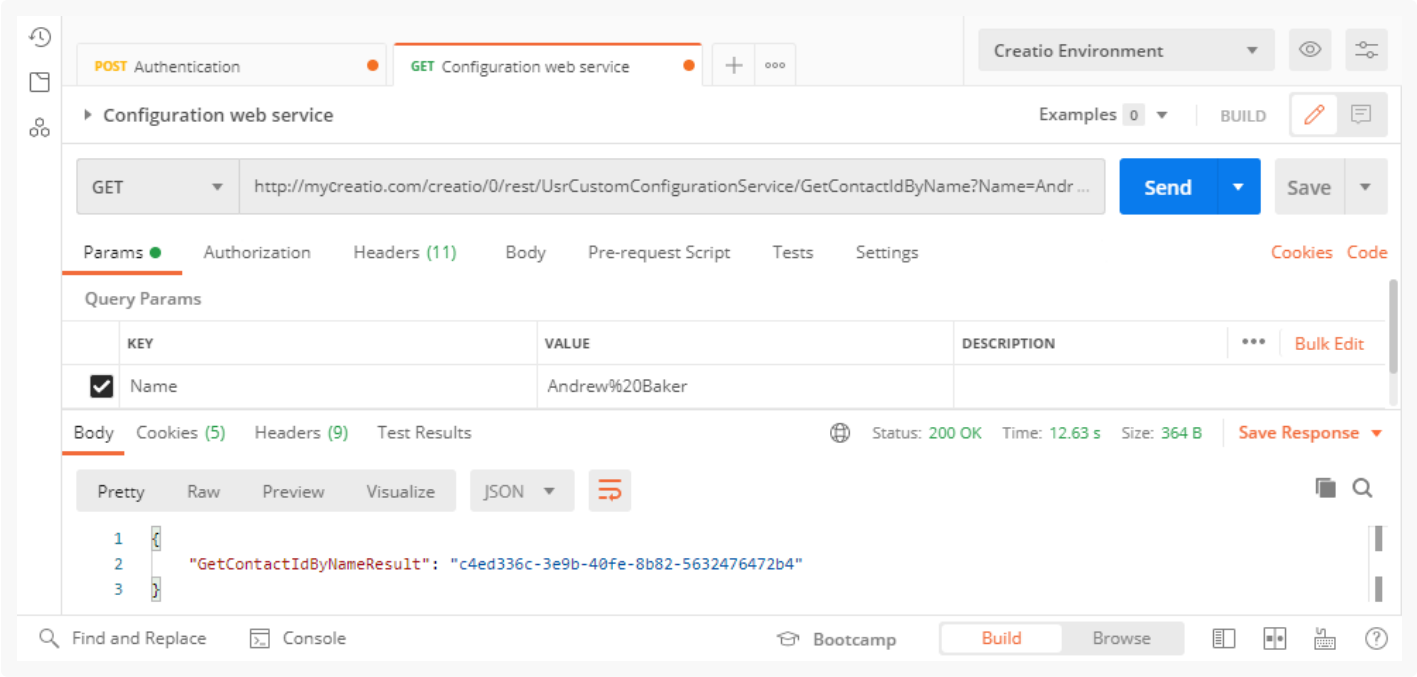
5. Выполнить запрос к пользовательскому веб-сервису с аутентификацией на основе cookies

Чтобы выполнить запрос в Postman, на панели инструментов рабочей области нажмите [*Send*].

Результат выполнения примера

В результате выполнения запроса будет получен ответ, который содержит JSON-объект. Тело ответа отображается на вкладке `Body` в Postman.

Если контакт, указанный в параметре `Name`, найден в базе данных, то в свойстве `GetContactIdByNameResult` будет возвращено значение идентификатора контакта.



Если контакт, указанный в параметре `Name`, не найден в базе данных, то в свойстве `GetContactIdByNameResult` будет возвращена пустая строка.