Библиотека
Cache Telegram Bot
Руководство разработчика

Данный документ представляет собой руководство разработчика библиотеки Cache Telegram Bot.

В документе представлены следующие разделы.

Общие сведения – содержит описание назначения библиотеки, системные требования.

Описание всех пакетов и классов – содержит описание пакетов, классов библиотеки.

Краткое руководство по работе с библиотекой – содержит перечень действий, которые необходимо выполнить перед началом работы с библиотекой, а также принцип работы с библиотекой.

СОДЕРЖАНИЕ

1 Общие сведения

1.1 Описание

Данная библиотека предназначена для решения задач взаимодействия с Telegram Bot API. Библиотека берет на себя задачи отправки и получения данных, проверки контроля целостности данных, распределения по типу события. Она позволяет использовать готовые оптимальные решения и упрощает разработку приложений.

1.2 Системные требования

Требуется интернет-соединение для взаимодействия с Telegram Bot API.
Intersystems Caché не ниже версии версии 2017.1

2 Описание пакетов и классов

2.1 Serialiser

Данный пакет содержит подпакеты и классы, которые реализуют процессы сериализации и десериализации.
Примеры использования данной библиотеки можно найти по ссылке.

2.2 Src

Данный пакет содержит классы и подпакеты, которые используются для системы контроля версий.

2.3 Telegram

2.3.1 Telegram.Bot

Дополнительную информацию по данному пакету можно найти в Автодокументации.

2.3.1.1 Telegram.Bot.Api

Основной класс. Содержит все методы, предоставляемые Telegram Bot API, а также два новых метода: SetLocalhook и DeleteLocalhook.

Метод SetLocalhook принимает один аргумент – название класса, унаследованного от AbstractHandler. В его основе лежит метод getUpdates.

SetLocalhook отправляет через указанный интервал запрос на получение обновлений. Интервал можно изменить, вызвав в терминале:

```
do ##class(Telegram.Bot.Settings).SetValue("interval", <Время в секундах>)
```

Метод DeleteLocalhook удаляет установленные хуки.

2.3.1.2 Telegram.Bot.AbstractBroker

Класс-родитель для Broker-классов, написанных разработчиком. Содержит в себе метод Update, который вызывается при срабатывании события у бота. Этот класс перехватывает все оповещения, которые присылает Telegram.

2.3.1.3 Telegram.Bot.AbstractHandler

Класс-родитель для Handler-классов, написанных разработчиком. В нем содержатся переопределенные методы, которые вызываются в зависимости от типа обновления или события в Telegram.

2.3.1.4 Telegram.Bot. ResponseException

Класс, возвращаемый при возникновении ошибки во время отправки запроса. Из него можно узнать название метода, в котором произошла ошибка, описание и параметры ошибки.

2.3.1.5 Telegram.Bot.Settings

Данный класс хранит настройки бота.

2.3.2 Telegram.DTO

Подпакет, хранящий транспортные объекты, о которых можно прочитать в документации Telegram Bot API.

2.3.3 Telegram.Examples

Подпакет, содержащий пример использования данной библиотеки.

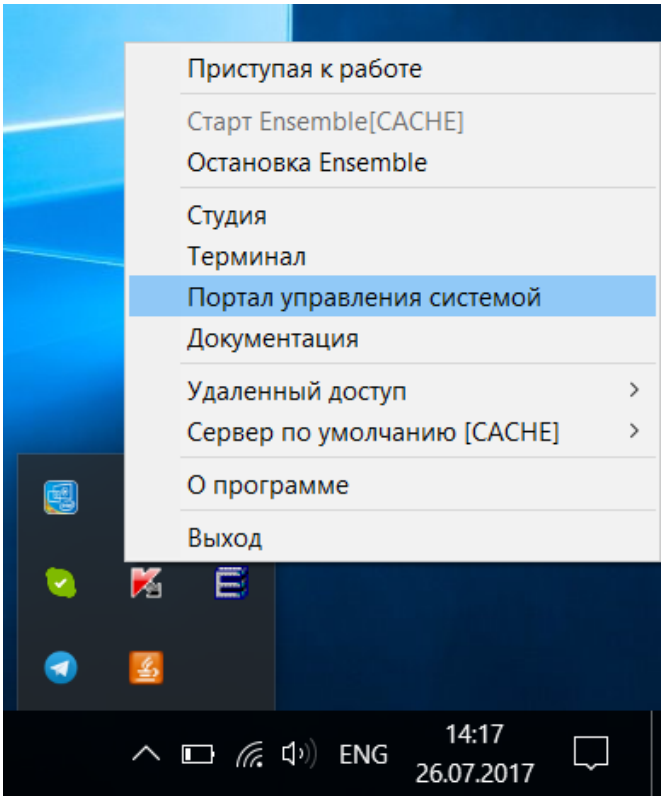
3 Краткое руководство по работе с библиотекой

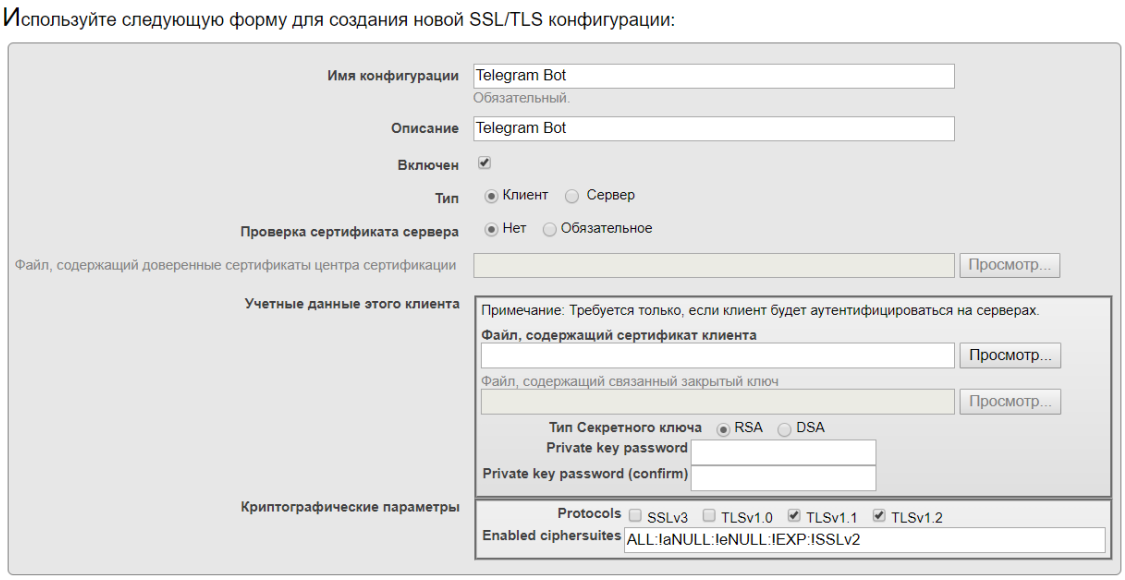
3.1 Подготовка к работе

Прежде чем начать работу с библиотекой, необходимо выполнить следующие действия:

1) Создание SSL/TLS конфигурации.

Перед началом работы с библиотекой необходимо создать SSL конфигурацию. Для этого нужно перейти в портал управления системой и открыть страницу "SSL/TLS Конфигурации". Далее нажать "Создать новую конфигурацию", указать имя и, если существует, файл сертификата. Сохранить.





2) Установка параметров.

В терминале выполнить команду

```
do ##class(Telegram.Bot.Settings).Init(<Имя SSL конфигурации>, <Токен вашего бота>)
```

Данная команда запишет в хранимый класс Telegram.Bot.Settings имя SSL сертификата и Токен Telegram бота.

Чтобы изменить значение параметров настроек, нужно вызвать в терминале

```
do ##class(Telegram.Bot.Settings).SetValue(<Code>, <Value>)
```

и передать в качестве первого аргумента идентификатор параметра, а в качестве второго параметра новое значение. Ознакомиться со списком всех параметров можно в Автодокументации.

3.2 Принцип работы

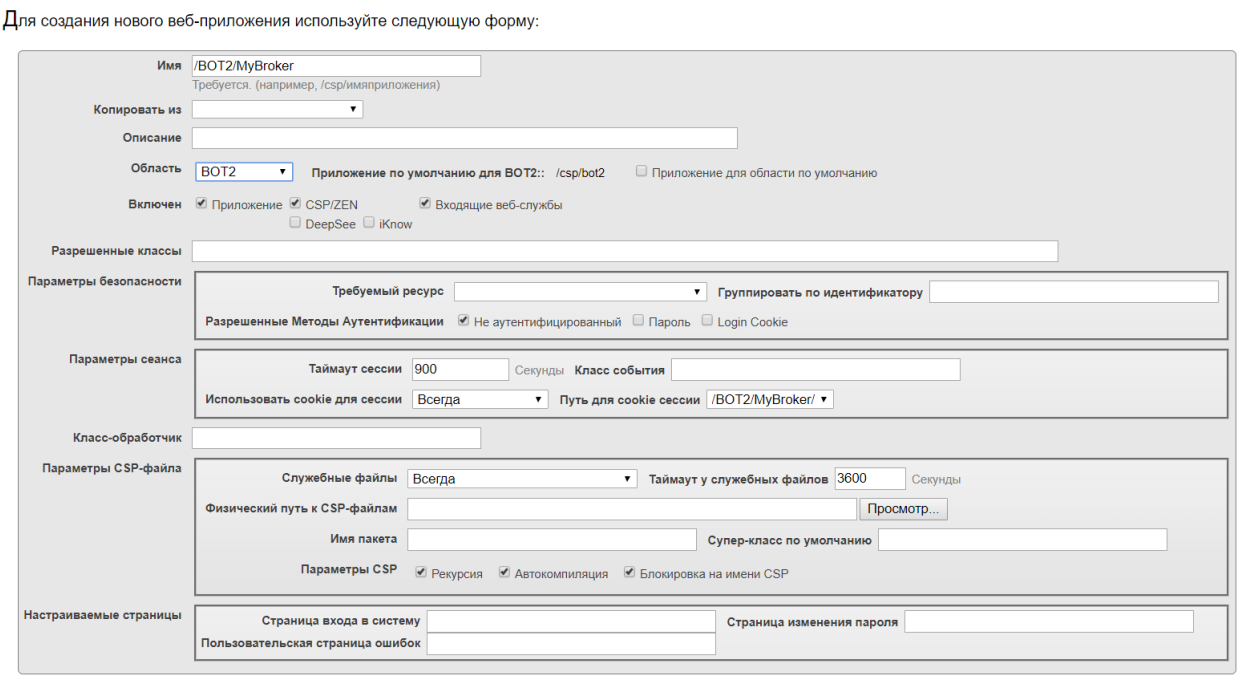
1) Создание наследника класса AbstractHandler.

В классе, унаследованном от Telegram.Bot.AbstractHandler, необходимо переопределить нужные вам класс-методы, такие как OnMessage, OnEditedMessage, OnInlineQuery, OnChosenInlineResult, OnCallbackQuery. Данные методы будут вызываться при получении обновлений. В зависимости от типа обновления вызывается определенный метод.

2) Создание наследника класса Telegram.Bot.AbstractBroker. Класс, унаследованный от Telegram.Bot.AbstractBroker, необходим для установки

Web-хуков. Этот класс является картой путей нашего Web API. В нем нужно переопределить класс-метод GetHandler, который должен возвращать экземпляр класса, описанный в пункте 1.

3) Создание Web-приложения. Данный пункт необходимо выполнить, если вы используете способ получения обновления через Web-хук. В портале управления системой необходимо перейти на страницу "Создать новое веб приложение". Далее нажать "Создать новую конфигурацию". На странице редактирования указать имя, начинающееся с слеша, область, в которой находится наше приложение, и полное название класса, описанного в пункте 2. При использовании Web-хуков, обновления будут отправляться на адрес https://<Адрес сервера>/<Имя веб приложения>/<Токен нашего бота>.



3.3 Примеры работы

Примеры использования библиотеки находятся в подпакете Telegram.Examples.

Он содержит три класса:

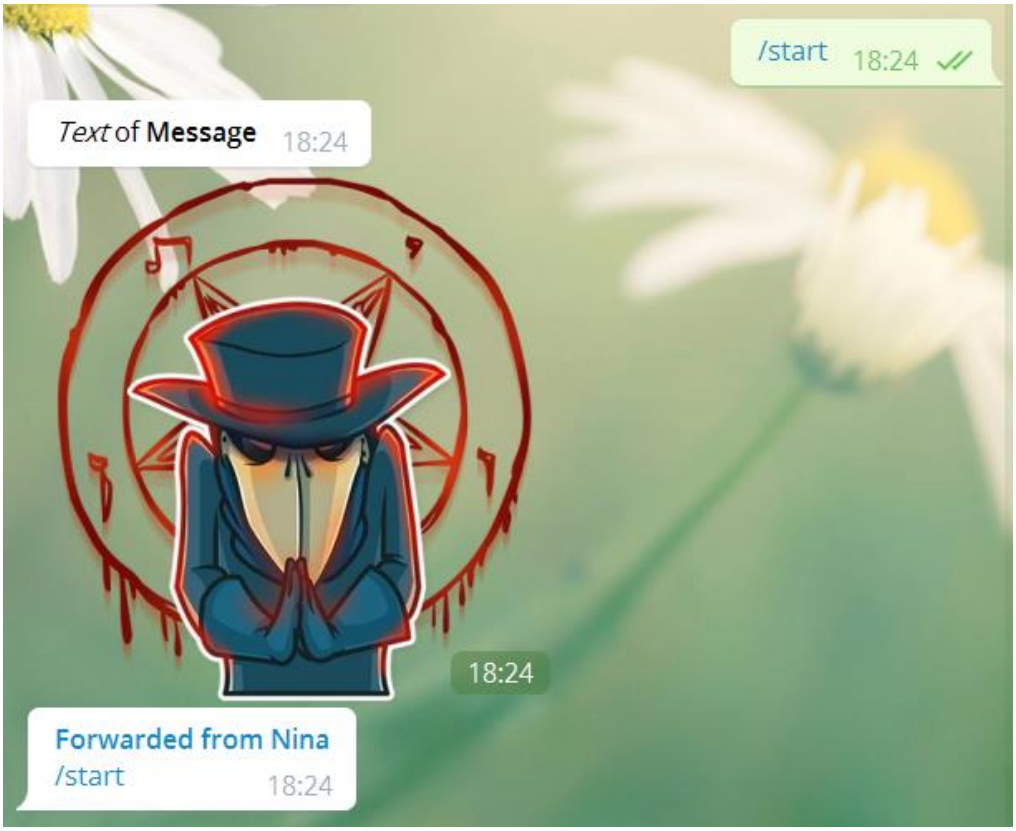
- Telegram.Examples.MyHandler – класс, унаследованный от Telegram.Bot.AbstractHandler. В нём переопределены методы OnCallbackQuery, OnMessage.
- Telegram.Examples.MyBroker – класс, унаследованный Telegram.Bot.AbstractBroker, с переопределённым методом GetHandler.

- Telegram.Examples.MyUserData – класс, используемый для хранения данных о пользователях.

Выполните команду:

```
do ##class(Telegram.Bot.Api).SetLocalhook("Telegram.Examples.MyHandler")
```

Метод OnMessage переопределен так, что пользователь, написавший боту команду /start получит в ответ два сообщения и стикер.



Данные о пользователе, написавшем боту, будут записаны в таблицу Telegram_Examples.MyUserData.