Универсальная форма в виде ссылки (для любого мессенджера)

Инструкция для разработчика бота.

Актуальный исходный код веб-формы реализующий все три вида форм смотрите по ссылке: go-to-the-mars.html

Пример работы веб-формы приведенной выше смотрите в Telegram боте https://t.me/metabot_test_form_bot

Форма открывается в отдельной странице браузера. Как обычная страница web-браузера.

Для Telegram ссылку можно отправить в виде inline-кнопки, но это все равно будет не "Web App приложение", а обычная страница открытая в браузере.

Если в мессенджере открытие ссылок во внешнем браузере выключено то форма откроется в браузере мессенджера, но все равно работа с данным видом формы будет отличаться от формы в виде Web App для inline/keyboard кнопки. При этом пользователю все равно доступна возможность позволяющая открыть ссылку во внешнем браузере. После закрытия формы нет гарантий, что пользователь вернется в браузер (например для PC), поэтому желательно прислать пользователю уведомление в бота, чтобы он нажал на него и вернулся в бота.

Если вы планируете использовать формы только в Telegram, то можете перейти к разделу описывающему форму на основе Web App, открывающейся с помощью inline-кнопки.

Краткое описание принципа работы бота:

1. В бот отправляется ссылка на форму (или inline-кнопка в виде ссылки). При нажатии на ссылку (или inline-кнопку в виде ссылки) открывается внешний браузер (или браузер встроенный в Telegram).

2. После заполнения формы данные должны быть отправлены на сторонний бэк, а из бэка данные должны быть отправлены в Metabot API. Для понимания смотрите исходный код веб-формы.
3. Данные пришедшие с формы сохраняются в JS Internal API Endpoint, который сохраняет данные и вызывает скрипт бота для продолжения беседы, этот скрипт работает с сохраненными данными формы, запрашивает подтверждения, что все введено верно, а также предлагает заполнить форму повторно.

Принцип работы, по шагам:

1. Бот генерирует ссылку с уникальным хэш-кодом, привязанным к лиду.
2. Бот отправляет эту ссылку в мессенджер в виде обычного сообщения.
3. Пользователь бота нажимает на ссылку, открывается браузер с формой.
4. Пользователь бота заполняет форму и нажимает кнопку для отправки формы.
5. Форма отправляет данные на backend сайта, где размещена форма.
6. Backend сайта отправляет данные формы в API Metabot, в данные должен быть включен уникальный хэш лида.
7. Metabot принимает и сохраняет данные заполненной формы.
8. Если все ок, то страница с формой в браузере должна быть закрыта, это делается с помощью простого JS кода (подробности рассматриваются на отдельной странице документации с описанием создания HTML-формы).

Таблицы для работы с формами

Чтобы рассматриваемый пример бота с формами работал, вам необходимо создать две кастомные таблицы.

Таблица: Хэш-коды лидов

> Таблица необходима для хранения соответствий между лидом и хэшом для лида.

При каждом вызове формы выполняется поиск хэша по лиду, если хэш не найден, то формируется новый. После отправки формы, запись с хэш-кодом удаляется, чтобы ссылка на каждый новый опрос содержала уникальный хэш. Поиск предыдущего хэша необходим, если есть формы, ожидающие заполнения, для случаев, если лид закрыл форму, а затем спустя время время решил ее заполнить, нажав на туже кнопку вызова формы. При этом в таблицу заложено поле (expire_at) - дата истечения хэш-кода, вы можете использовать это поле для реализации времени жизни хэш-кода, но в таком случае в веб-форме необходимо предусмотреть уведомление о том, что форма истекла, или закрывать ее автоматом после истечения, или формировать в боте новую ссылку, если лид пытается отправить форму, хэш которой истек. Это требование не обязательно, может реализовываться по мере необходимости и зависит от конкретной задачи.

Структура таблицы:

ID	АКТИВНА	СИХР. СТРУКТУРЫ	ОТОБРАЖАТЬ В МЕНЮ	НАИМЕНОВАНИЕ	ЗАГОЛОВОК В ИНТЕРФЕЙСЕ
69	Да	Да	Да	lead_form_hash	Leads Hashes for Forms

Структура таблицы (JSON словарь):

АКТИВНО	СИХР. СТРУКТУРЫ	НАИМЕНОВАНИЕ	ЗАГОЛОВОК	ТИП	ОБЯЗАТЕЛЬНОЕ	ЗНАЧЕНИЕ ПО УМОЛЧАНИЮ	РАЗМЕР	ТОЧНОСТЬ	ПОДСКАЗКА	ВСПЛЫВАЮЩАЯ ПОДСКАЗКА
Да	Да	id	ID	AUTOINCREMENT	Да	NULL				
Да	Да	form	Form	TEXT	Да					
Да	Да	lead_id	Lead ID	BIG_INT	Да	NULL				
Да	Да	hash	Hash	TEXT	Да					
Да	Да	expire_at	Expire At	DATETIME		NULL				
Да	Да	created_at	Created At	CREATED_AT	Да	NOW				

Таблица: Данные форм

Таблица необходима для хранения данных заполненных форм.

Данная таблица реализована как универсальная, все данные пришедшие с формы сохраняются в текстовом поле в виде JSON, поэтому она может работать с любой формой, но работать с строкой в виде JSON в JS неудобно, тк для доступа к данным приходится выполнять JSON.parse(), вы можете нормализовать данные, реализуя для каждой формы бота отдельную таблицу с необходимыми полями для хранения, например такими как: Имя, Фамилия, Адрес и т.д., чтобы в дальнейшем было проще работать с данными и выполнять поиск по таблице. Также, в вашем случае таблица для хранения данных может и не потребоваться, если всплывающая форма очень простая и предназначена, например, для заполнения одного поля, например, адреса с авто комплитом от Dadata или для выбора выпадающего списка или даты в календаре. Т.е. форма может реализовывать даже один компонент для заполнения которого необходим HTML + JS.

Структура таблицы:

ID	АКТИВНА	СИХР. СТРУКТУРЫ	ОТОБРАЖАТЬ В МЕНЮ	НАИМЕНОВАНИЕ	ЗАГОЛОВОК В ИНТЕРФЕЙСЕ
70	Да	Да	Да	form_results	Form Results

Структура таблицы (JSON словарь):

АКТИВНО	СИХР. СТРУКТУРЫ	НАИМЕНОВАНИЕ	ЗАГОЛОВОК	ТИП	ОБЯЗАТЕЛЬНОЕ	ЗНАЧЕНИЕ ПО УМОЛЧАНИЮ	РАЗМЕР	ТОЧНОСТЬ	ПОДСКАЗКА	ВСПЛЫВАЮЩАЯ ПОДСКАЗКА
Да	Да	id	ID	AUTOINCREMENT	Да	NULL				
Да	Да	form	Form	TEXT	Да					
Да	Да	lead_id	Lead ID	BIG_INT	Да	NULL				
Да	Да	data	Data	TEXT						
Да	Да	created_at	Created At	CREATED_AT	Да	NOW				
Да	Да	updated_at	Updated At	UPDATED_AT		NULL				

Чтобы посмотреть исходный код внутреннего API, сначала перейдите в бота по ссылке на любой из скриптов.

Нажмите в предупреждающем сообщении на ссылку для смены активного бота, а затем перейдите на страницу с внутренним API в настройках бота или по ссылке.

Если у вас нет доступа к указанному скрипту, то запросите шаблон бота в технической поддерже Metabot.

Скрипт для формирования и отправки ссылки в мессенджер

Пример скрипта:

ID36182 Форма в виде ссылки

Тип: Стандарт **Раздел:** (корень)

КОМАНДЫ

Эти команды бот выполняет до того как покажет меню.

КОМАНДА	СОДЕРЖИМОЕ

Выполнить JavaScript	`// СКРИПТ ФОРМИРОВАНИЯ ХЭШ-КОДА И ОТПРАВКИ ССЫЛКИ НА ФОРМУ`
	`// PS: Здесь же можно отправить ссылку в виде inline кнопки (не inline Web App, а обычную кнопку с ссылкой)`
	`// реализовав это в виде  команды JS Callback`
	`// но такой вариант будет работать только в Telegram`
	`// поэтому чтобы данный вариант формы был универсальным отправляем ее просто в виде ссылки`

Отправить текст	Перейдите по ссылке ниже и заполните форму {{ &$formUrl }}
Выполнить скрипт	Главное меню

Скрипт отправки ссылки состоит из следующих команд бота:

- **Выполнить JavaScript**:

```
let md5 = require('Common.Utils.Md5')

let expireAt = new Date()
expireAt.setSeconds(expireAt.getSeconds() + 300)

// Todo: Можно вынести в сниппет
let hashItems = table.find('lead_form_hash', [], [
  ['form', '=', 'mars'],
  ['lead_id', '=', leadId],
])

let leadHash
if (hashItems.length > 0) {
  leadHash = hashItems[0].hash
} else {
  const salt = 'YourSuperSecretString' + (new Date()).getTime()
  leadHash = md5(salt + leadId)

  table.createItem('lead_form_hash', {
    'form': 'mars',
    'lead_id': leadId,
    'hash': leadHash,
    'expireAt': expireAt
  })
}

// Вы можете вынести эту ссылку в атрибуты бота
let url = 'https://app.metabot24.com/testWidgets/go-to-the-mars.html?q=' + leadHash

memory.setAttr('formUrl', url)
```

PS: Здесь же можно отправить ссылку в виде inline кнопки (не inline Web App, а обычную кнопку с ссылкой)
реализовав это в виде команды JS Callback
но такой вариант будет работать только в Telegram
поэтому чтобы данный вариант формы был универсальным отправляем ее просто в виде ссылки

- Команда **Отправить текст**:

```
Перейдите по ссылке ниже и заполните форму


{{ &$formUrl }}
```

- Команда **Выполнить скрипт** — команда необходима для вывода меню.

Внутреннее api для приема данных с формы и сохранения их в таблицу

Это точка API, куда приходит запрос после заполнения формы.

ВНУТРЕННЕЕ API. КОНЕЧНЫЕ ТОЧКИ

ID	АКТИВНА	НАИМЕНОВАНИЕ	АЛИАС	МЕТОД	ФОРМАТ
94	Да	submit-form	submit-form	POST	JSON

JavaScript для вычисления API ответа (Response Body):

```
let requestParams = request.json
let formLeadId = null

if (!request.json || !request.json.q) {
  return {"result": false, "message": "Invalid Lead Hash"}
}

// Todo: Можно вынести в сниппет
let hashItems = table.find('lead_form_hash', [], [
  ['form', '=', 'mars'],
  ['hash', '=', request.json.q],
])

let found = 0

if (hashItems.length > 0) {
  formLeadId = hashItems[0].lead_id
```

```
  hashItems.forEach(hashItem => hashItem.delete())
}

if (formLeadId > 0) {
  // Удаляем предыдущие ответы
  // Todo: Можно вынести в сниппет
  oldResults = table.find('form_results', [], [
    ['form', '=', 'mars'],
    ['lead_id', '=', formLeadId],
  ])
  if (oldResults.length > 0) {
    oldResults.forEach(oldResult => oldResult.delete())
  }

  // Предварительно сохраняем данные в таблице
  table.createItem('form_results', {
    "form": "mars",
    "lead_id": formLeadId,
    "data": request.string
  })

  bot.scheduleScriptByCode('after_submit', formLeadId)

  //Для передачи данных в скрипт без предварительного сохранения в таблицу
  //bot.scheduleScriptByCode('after_submit', leadId, null, {"script_request_params":
requestParams})
  return {"result": true}
} else {
  return {"result": false, "message": "Lead by hash is not found"}
}
```

Скрипт выполняемый после вызова API

Данный скрипт нужен чтобы пользователь продолжил диалог с ботом, а также демонстрирует как использовать данные сохраненные в таблицу результатов и вывести их ввиде текста.

Пример скрипта.

Скрипт отправки ссылки состоит из следующих команд бота и пунктов меню:

- Команда **Выполнить JavaScript**:

```
const menuHelper = require('Common.TelegramComponents.MenuHelper')

// ------------------------------------------------------------
// Удаляем кнопку с формой для Inline формы, если она выведена

if (menuHelper.hasLastTelegramMessageId()) {
  // Удаляем кнопку с ссылкой на форму
  menuHelper.removeInlineKeyboard()

  // Сбрасываем ID последнего сообщения
  menuHelper.clearLastTelegramMessageId()
}
// ------------------------------------------------------------

// ------------------------------------------------------------
// Для Inline кнопки с формой
// Удаляем кнопку с формой , если она выведена
// Удаляем здесь - тк после заполнения формы в JS-Callback мы уже не попадаем
// Тк прием данных с формы осуществляет через Internal Endpoiint
// Для keyboard кнопки ссылку на форму с кнопкой здесь удалять не нужно,
// тк для keyboard кнопки мы возвращаемся в JS-Callbak после заполнении формы
if (menuHelper.hasLastTelegramMessageId()) {
  // Удаляем кнопку
  menuHelper.removeInlineKeyboard()

  // Сбрасываем ID последнего сообщения
  menuHelper.clearLastTelegramMessageId()
}
// ------------------------------------------------------------

// Если получаем данные напрямую, без таблицы form_results
//let data = request.json

// Если получаем данные из таблицы form_results
// Todo: Можно вынести в сниппет
let data = table.find('form_results', [], [
  ['form', '=', 'mars'],
  ['lead_id', '=', leadId]
```

```
])

if (data.length > 0 && typeof(data[0].data) === 'string' && data[0].data.length > 0) {
  data = JSON.parse(data[0].data)
} else {
  data = {}
  bot.sendText('Ошибка! Данные не найдены')
}

// Для определения какую кнопку показывать в меню ниже
// См условие аунктов меню текущего скрипта
memory.setAttr('mode', '')
if (typeof(data.mode) === 'string' && data.mode.length > 0) {
  memory.setAttr('mode', data.mode)
}



/*
// Сообщение о результате отработки формы, отправляется в бота
// Но проблема что система распознает сообщение как вебхук от лида
// Поэтому использовать можно только в JS-Callback и игнорировать вебхук вручную
// Или через регулярку маршрута, запуская отдельны скрипт
//
// Данное сообщение также автоматом закрывает web-view
// Но пробелм с закрытием нет, это выполняется в JS итак с помощью
// кода window.Telegram.WebApp.sendData(JSON.stringify(formData));
// или при выполнении window.Telegram.WebApp.close()
//
// Обычно используется для игры в web_view для вывода очков после победы или проигрыша
if (data.tg_query_id && data.tg_query_id.length > 0) {
  //https://core.telegram.org/bots/api#sentwebappmessage
  //https://core.telegram.org/bots/api#inlinequeryresultarticle
  bot.sendPayload('answerWebAppQuery', {
    "web_app_query_id": data.tg_query_id,
    "result": {
      "type": "article",
      "id": "1", // Unique identifier for this result, 1-64 Bytes
      "title": "Форма заполнена",
      //"description": "Description",
      "input_message_content": {"message_text": "Спасибо! Данные сохранены!"}
```

```
    }
  })
}
*/

memory.setJsonAttr('data', data)
memory.setAttr('is_qualified', data.is_qualified ? 'Да' : 'Нет')
memory.setAttr('has_experience', data.has_experience ? 'Да' : 'Нет')
```

- Команда **Отправить текст**:

```
Ваши данные:
Имя: {{ &$$data.name }}
Email: {{ &$$data.email }}
Возраст: {{ &$$data.age }}
Профессия: {{ &$$data.specialization}}
Ваш адрес проживания: {{ &$$data.address}}
Прошел курсы в Центре подготовки космонавтов: {{ &$is_qualified }}
Я уже летал в космос (имею опыт): {{ &$has_experience}}

Все данные в виде JSON:
{{ &$$data }}
```

- Команда **Отправить текст**:

```
Все верно?
```

- Пункты меню данного скрипта, с условиями:

КОД	НАДПИСЬ	СКРИПТ	УСЛОВИЕ ВЫВОДА
1	Да, все верно!	Подтвердил сохранение	

2	Нет, заполнить повторно	Форма в виде ссылки	`let mode = memory.getAttr('mode')`
			`if (mode !== 'tg_inline' && mode !== 'tg_keyboard') {`
			`  return true`
			`}`

2	Нет, заполнить повторно	Форма в виде inline кнопки	`let mode = memory.getAttr('mode')`
			`if (mode === 'tg_inline') {`
			`  return true`
			`}`

3	Нет, заполнить повторно	Форма в виде keyboard кнопки	`let mode = memory.getAttr('mode')`
			`if (mode === 'tg_keyboard') {`
			`  return true`
			`}`

Условия кнопок необходимы для вызова скрипта соответствующего текущему формату формы.

Условие для вывода универсальной формы в виде ссылки:

```
let mode = memory.getAttr('mode')
if (mode === 'tg_inline') {
  return true
}
```

Условие для вывода формы в Web App при нажатии на inline-кнопку:

```
let mode = memory.getAttr('mode')
if (mode !== 'tg_inline'  && mode !== 'tg_keyboard') {
  return true
}
```

Условие для вывода формы в Web App при нажатии на keyboard-кнопку:

```
let mode = memory.getAttr('mode')
if (mode === 'tg_keyboard') {
  return true
}
```

Версия #3
Юрий Гарашко создал 14 June 2023 13:53:53
Ирина Петрова обновил 5 February 2024 14:28:10