# Namespace Telegram_bots

## Classes

[Chat](#)

The object providing the chat

[Message](#)

The class representing the message

[TelegramBot](#)

This class is a telegram bot

[Update](#)

A class representing Update

[User](#)

A class representing the user

## Enums

[Chat.Types](#)

Chat Types

## Delegates

[TelegramBot.UpdateHandler](#)

Signature of the "OnUpdate" event

# Class Chat

Namespace: Telegram_bots

Assembly: Telegram bots.dll

The object providing the chat

```
public class Chat
```

**Inheritance**

object ← Chat

**Inherited Members**

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() ,
object.MemberwiseClone() , object.ReferenceEquals(object, object)

# Properties

## FirstName

Chat`s first name

```
public string? FirstName { get; set; }
```

### Property Value

string

## Id

Chat id

```
public required long Id { get; init; }
```

### Property Value

[long↗](#)

# LastName

Chat's last name

```csharp
public string? LastName { get; set; }
```

## Property Value

[string↗](#)

# Title

Chat Title

```csharp
public string? Title { get; set; }
```

## Property Value

[string↗](#)

# Type

Chat Type

```csharp
public required Chat.Types Type { get; init; }
```

## Property Value

[Chat](#).[Types](#)

# Username

Chat username

```
public string? Username { get; set; }
```

## Property Value

[string](#)⧉

# Methods

## FromJSON(string)

Make a chat from a json string

```
public static Chat FromJSON(string jsonString)
```

## Parameters

`jsonString` [string](#)⧉

   Json string

## Returns

[Chat](#)

   Chat

## FromJSON(JsonDocument)

Make a chat from JsonDocument

```
public static Chat FromJSON(JsonDocument jsonDocument)
```

## Parameters

`jsonDocument` [JsonDocument](#)⧉

   JsonDocument

## Returns

[Chat](#)

Chat

## Exceptions

[ArgumentException⧉](#)

Called if some argument is incorrect

# FromJSON(JsonElement)

Make a chat from JsonElement

```
public static Chat FromJSON(JsonElement rootJsonElement)
```

## Parameters

`rootJsonElement` [JsonElement⧉](#)

JsonElement

## Returns

[Chat](#)

Chat

# ToString()

Turns it into a string

```
public override string ToString()
```

## Returns

[string⧉](#)

A string representing the chat

# Enum Chat.Types

Namespace: [Telegram_bots](#)

Assembly: Telegram bots.dll

Chat Types

```
public enum Chat.Types
```

# Fields

Channel = 3

    Channel

Group = 1

    Group

Private = 0

    Private chat

Supergroup = 2

    Supergroup

# Class Message

Namespace: [Telegram_bots](#)

Assembly: Telegram bots.dll

The class representing the message

```
public class Message
```

**Inheritance**

[object](#) ← Message

**Inherited Members**

[object.Equals(object, object)](#) , [object.GetType()](#) , [object.MemberwiseClone()](#) ,
[object.ReferenceEquals(object, object)](#)

# Properties

## Chat

The chat to which the message was sent

```
public required Chat Chat { get; set; }
```

### Property Value

[Chat](#)

## Datetime

Date the message was sent

```
public required int Datetime { get; init; }
```

### Property Value

[int↗](#)

## From

The sender of the message

```
public User? From { get; set; }
```

### Property Value

[User](#)

## Id

Message ID

```
public required long Id { get; init; }
```

### Property Value

[long↗](#)

## Text

The text of the message

```
public string? Text { get; set; }
```

### Property Value

[string↗](#)

# Methods

## Equals(object?)

Checks if the objects are the same

```csharp
public override bool Equals(object? obj)
```

## Parameters

`obj` object↗

   Object

## Returns

bool↗

   Are the objects the same

# FromJSON(string)

Create a message from a json string

```csharp
public static Message FromJSON(string jsonString)
```

## Parameters

`jsonString` string↗

   Json string

## Returns

[Message](#)

   Message

# FromJSON(JsonDocument)

Create a message from a JsonDocument

```
public static Message FromJSON(JsonDocument jsonDocument)
```

## Parameters

`jsonDocument` [JsonDocument⧉](#)

    JsonDocument

## Returns

[Message](#)

    Message

# FromJSON(JsonElement)

Create a message from a JsonElement

```
public static Message FromJSON(JsonElement rootJsonElement)
```

## Parameters

`rootJsonElement` [JsonElement⧉](#)

    Root JsonElement

## Returns

[Message](#)

    Message

# GetHashCode()

Returns the hash code of the message

```
public override int GetHashCode()
```

## Returns

[int](link)

   Hash code

# ToString()

Turns it into a string

```
public override string ToString()
```

## Returns

[string](link)

   The string representing the message

# Class TelegramBot

Namespace: [Telegram_bots](#)

Assembly: Telegram bots.dll

This class is a telegram bot

```
public class TelegramBot : IDisposable
```

**Inheritance**

[object](#) ← TelegramBot

**Implements**

[IDisposable](#)

**Inherited Members**

[object.Equals(object)](#) , [object.Equals(object, object)](#) , [object.GetHashCode()](#) , [object.GetType()](#) ,
[object.MemberwiseClone()](#) , [object.ReferenceEquals(object, object)](#) , [object.ToString()](#)

# Constructors

## TelegramBot(string)

A constructor for creating a bot

```
public TelegramBot(string Token)
```

## Parameters

`Token` [string](#)

Token for the bot

## Examples

```
public static async Task Main(string[] args)
{
```

```
    using TelegramBot bot = new (args[0]);
}
```

## Exceptions

[IncorrectRequestException](#)

    Called if you entered the wrong token

# Fields

## Token

Token for the bot

```
protected readonly string Token
```

## Field Value

[string ⧉](#)

## httpClient

An object for network access

```
protected HttpClient httpClient
```

## Field Value

[HttpClient ⧉](#)

## lastChatId

Chat_id from the last message

```
protected long lastChatId
```

## Field Value

[long↗]

## lastUpdateId

Update_id from the last update

```
protected long lastUpdateId
```

## Field Value

[long↗]

# Methods

## Dispose()

Release the resources occupied by the bot

```
public void Dispose()
```

## Dispose(bool)

Release the resources occupied by the bot

```
protected virtual void Dispose(bool disposing)
```

## Parameters

disposing [bool↗]

    Release managed resources

## ~TelegramBot()

Release unmanaged resources occupied by the bot

```
protected ~TelegramBot()
```

# GetMe()

Receives information about the bot

```
public Task<JsonDocument> GetMe()
```

## Returns

Task <JsonDocument> >

Information about the bot

## Exceptions

IncorrectRequestException

It is thrown if an incorrect request was made

# SendMessage(string)

Sends a message to the chat from which the latest update was received

```
public Task<Message> SendMessage(string Text)
```

## Parameters

Text string

The text of the message

## Returns

Task <Message>

The message that was sent

## Exceptions

[ArgumentNullException](#)⧉

It is thrown if one of the arguments is null

[IncorrectRequestException](#)

It is thrown if an incorrect request was made

# SendMessage(string, long)

Sends a message

```
public Task<Message> SendMessage(string Text, long ChatId)
```

## Parameters

`Text` [string](#)⧉

The text of the message

`ChatId` [long](#)⧉

Chat_id of the chat to send the message to

## Returns

[Task](#)⧉ <[Message](#)>

The message that was sent

## Exceptions

[ArgumentNullException](#)⧉

It is thrown if one of the arguments is null

[IncorrectRequestException](#)

It is thrown if an incorrect request was made

# StartPolling()

Start polling

```csharp
public Task StartPolling()
```

## Returns

[Task](#)

## Exceptions

[IncorrectRequestException](#)

It is thrown if an incorrect request was made

# Events

# OnUpdate

Called if an update has arrived

```csharp
public event TelegramBot.UpdateHandler? OnUpdate
```

## Event Type

[TelegramBot](#).[UpdateHandler](#)

# Delegate TelegramBot.UpdateHandler

Namespace: [Telegram_bots](#)

Assembly: Telegram bots.dll

Signature of the "OnUpdate" event

```
public delegate void TelegramBot.UpdateHandler(Update update, TelegramBot bot)
```

## Parameters

update [Update](#)

    Signature of the "OnUpdate" event

bot [TelegramBot](#)

    Signature of the "OnUpdate" event

# Class Update

Namespace: [Telegram_bots](#)

Assembly: Telegram bots.dll

A class representing Update

```
public class Update
```

**Inheritance**

[object](#) ← Update

**Inherited Members**

[object.Equals(object, object)](#) , [object.GetType()](#) , [object.MemberwiseClone()](#) ,
[object.ReferenceEquals(object, object)](#)

# Properties

## Message

Message

```
public Message? Message { get; set; }
```

### Property Value

[Message](#)

## UpdateId

Update ID

```
public required long UpdateId { get; init; }
```

### Property Value

[long ↗](#)

# Methods

## Equals(object?)

Checks the similarity of objects

```
public override bool Equals(object? obj)
```

### Parameters

`obj` [object ↗](#)

 Object

### Returns

[bool ↗](#)

 The similarity of objects

## FromJSON(string)

Get an update from a json string

```
public static Update FromJSON(string jsonString)
```

### Parameters

`jsonString` [string ↗](#)

 Json string

### Returns

[Update](#)

 Update

# FromJSON(JsonDocument)

Get an update from a JsonDocument

```
public static Update FromJSON(JsonDocument jsonDocument)
```

## Parameters

`jsonDocument` [JsonDocument](#)⤢

   JsonDocument

## Returns

[Update](#)

   Update

# FromJSON(JsonElement)

Get an update from a JsonElement

```
public static Update FromJSON(JsonElement rootJsonElement)
```

## Parameters

`rootJsonElement` [JsonElement](#)⤢

   Root JsonElement

## Returns

[Update](#)

   Update

# GetHashCode()

Returns the hash code of the message

```
public override int GetHashCode()
```

## Returns

[int](#)

Hash code

# ToString()

Turns it into a string

```
public override string ToString()
```

## Returns

[string](#)

The line representing the update

# Class User

Namespace: [Telegram_bots](#)

Assembly: Telegram bots.dll

A class representing the user

```
public class User
```

**Inheritance**

[object](#) ← User

**Inherited Members**

[object.Equals(object)](#) , [object.Equals(object, object)](#) , [object.GetHashCode()](#) , [object.GetType()](#) ,
[object.MemberwiseClone()](#) , [object.ReferenceEquals(object, object)](#)

# Properties

## FirstName

First name`s user

```
public required string FirstName { get; init; }
```

## Property Value

[string](#)

## Id

User ID

```
public required long Id { get; init; }
```

## Property Value

[long](#)↗

# IsBot

Is the user a bot?

```csharp
public required bool IsBot { get; init; }
```

## Property Value

[bool](#)↗

# IsPremium

A premium user?

```csharp
public bool? IsPremium { get; set; }
```

## Property Value

[bool](#)↗?

# LastName

Last name of the user

```csharp
public string? LastName { get; set; }
```

## Property Value

[string](#)↗

# Username

Username

```
public string? Username { get; set; }
```

## Property Value

[string](#)

# Methods

## FromJSON(string)

Get an update from a json string

```
public static User FromJSON(string jsonString)
```

## Parameters

`jsonString` [string](#)

  Json string

## Returns

[User](#)

  User

## FromJSON(JsonDocument)

Get an update from a JsonDocument

```
public static User FromJSON(JsonDocument jsonDocument)
```

## Parameters

`jsonDocument` [JsonDocument](#)

  JsonDocument

## Returns

[User](#)

User

# FromJSON(JsonElement)

Get an update from a JsonElement

```
public static User FromJSON(JsonElement rootJsonElement)
```

## Parameters

`rootJsonElement` [JsonElement⧉](#)

Root JsonElement

## Returns

[User](#)

User

# ToString()

Turns it into a string

```
public override string ToString()
```

## Returns

[string⧉](#)

The line representing the user

# Namespace Telegram_bots.Exceptions

## Classes

[IncorrectRequestException](#)

An exception that is raised when an incorrect request was made

# Class IncorrectRequestException

Namespace: [Telegram_bots](#).[Exceptions](#)

Assembly: Telegram bots.dll

An exception that is raised when an incorrect request was made

```
public class IncorrectRequestException : Exception, ISerializable
```

**Inheritance**

[object](#) ← [Exception](#) ← IncorrectRequestException

**Implements**

[ISerializable](#)

**Inherited Members**

[Exception.GetBaseException()](#) , [Exception.GetType()](#) , [Exception.ToString()](#) , [Exception.Data](#) , [Exception.HelpLink](#) , [Exception.HResult](#) , [Exception.InnerException](#) , [Exception.Message](#) , [Exception.Source](#) , [Exception.StackTrace](#) , [Exception.TargetSite](#) , [Exception.SerializeObjectState](#) , [object.Equals(object)](#) , [object.Equals(object, object)](#) , [object.GetHashCode()](#) , [object.MemberwiseClone()](#) , [object.ReferenceEquals(object, object)](#)

# Constructors

## IncorrectRequestException()

Causes an exception

```
public IncorrectRequestException()
```

## IncorrectRequestException(string?)

Causes an exception

```
public IncorrectRequestException(string? message)
```

## Parameters

**message** [string↗]

    Error message

## IncorrectRequestException(string, Exception)

Causes an exception

```
public IncorrectRequestException(string message, Exception inner)
```

## Parameters

**message** [string↗]

    Error message

**inner** [Exception↗]

    Parental error

# Methods

## ThrowIfNotOk(JsonDocument)

Checks the value of "ok" in the JsonDocument and if it is "false" it throws an exception

```
public static void ThrowIfNotOk(JsonDocument json)
```

## Parameters

**json** [JsonDocument↗]

    The JsonDocument that needs to be checked