# Namespace Bot.Exceptions

## Classes

AppException

Constructs custom error message to add more information to the original error. Also generates the error response to be sent to the client. Logs the error to App Insights. and to console.

ErrorCodes

Custom error codes to be used in AppException Use these codes to construct custom error messages.

# Class AppException

Namespace: Bot.Exceptions

Assembly: Bots.dll

Constructs custom error message to add more information to the original error. Also generates the error response to be sent to the client. Logs the error to App Insights. and to console.

```
public class AppException : Exception, ISerializable
```

**Inheritance**

object ← Exception ← AppException

**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

### AppException(string, string, Exception?, int, ILoggerFactory)

```
public AppException(string code, string message, Exception? error, int statusCode,
ILoggerFactory loggerFactory)
```

#### Parameters

`code` string

A custom error codes, use one of ErrorCodes, or Error code in the format of `CUSTOM_ERROR_CODE`

`message` string

Error custom message to make it easier for clients to understnad

`error` Exception

The actual captured internal error,if any

`statusCode` int

An HTTP status code for this error, will be returned as response. Defaults to 500

`loggerFactory` ILoggerFactory

# Properties

## Code

```
public string Code { get; }
```

### Property Value

string

## StatusCode

```
public int StatusCode { get; }
```

### Property Value

int

# Methods

## GetErrorResponse()

```
public Dictionary<string, string> GetErrorResponse()
```

### Returns

Dictionary<string, string>

# Class ErrorCodes

Namespace: Bot.Exceptions

Assembly: Bots.dll

Custom error codes to be used in AppException Use these codes to construct custom error messages.

```
public class ErrorCodes
```

**Inheritance**

object ← ErrorCodes

**Inherited Members**

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

# Fields

## DatabaseError

```
public static string DatabaseError
```

### Field Value

string

## InternalError

```
public static string InternalError
```

### Field Value

string

## InvalidInput

```
public static string InvalidInput
```

### Field Value

string

## UnauthorizedAccess

```
public static string UnauthorizedAccess
```

### Field Value

string

# Namespace Bot.Logger

## Classes

Logger

Logger to log messages to App Insights and to console. This is added as a singleton service in Program.cs and can be injected in any class. Usage:

```
public class SomeClass
{
private readonly Logger _logger;
public SomeClass(Logger logger)
{
_logger = logger;
}
public void SomeMethod()
{
_logger.Log.LogInformation("Some message");
}
}
```

# Class Logger

Namespace: Bot.Logger

Assembly: Bots.dll

Logger to log messages to App Insights and to console. This is added as a singleton service in Program.cs and can be injected in any class. Usage:

```
public class SomeClass
{
private readonly Logger _logger;
public SomeClass(Logger logger)
{
_logger = logger;
}
public void SomeMethod()
{
_logger.Log.LogInformation("Some message");
}
}
```

```
public class Logger
```

**Inheritance**

object ← Logger

**Inherited Members**

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

# Constructors

## Logger(ILoggerFactory)

Logger to log messages to App Insights and to console. This is added as a singleton service in Program.cs and can be injected in any class. Usage:

```
public class SomeClass
{
```

```
private readonly Logger _logger;
public SomeClass(Logger logger)
{
_logger = logger;
}
public void SomeMethod()
{
_logger.Log.LogInformation("Some message");
}
}
```

```
public Logger(ILoggerFactory loggerFactory)
```

### Parameters

`loggerFactory` ILoggerFactory

Logger factory to create logger

## Properties

### Log

```
public ILogger Log { get; }
```

### Property Value

ILogger

# Namespace Bot.Src

## Classes

Configuration

# Class Configuration

Namespace: Bot.Src

Assembly: Bots.dll

```
public static class Configuration
```

**Inheritance**

object ← Configuration

**Inherited Members**

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

## Properties

### HasBeenLoaded

Flag to check if the config has been loaded.

```
public static bool HasBeenLoaded { get; }
```

#### Property Value

bool

True if the config has already been loaded.

### LocalConfig

Returns the local config.

```
public static Dictionary<string, dynamic> LocalConfig { get; }
```

#### Property Value

Dictionary<string, dynamic>

A

# Methods

## LoadConfig()

Loads config from the local config file.

The config file is expected to be in JSON format and path must be set in the `CONFIG_PATH` env variable. This method is called automatically when the Configuration class is initialised, no need of callig it explicitly.

```
public static void LoadConfig()
```

### Exceptions

Exception

If config env variable is not found or value is null.

Exception

If the config file contents is null.

# Namespace Bots

## Classes

FunctionExecutorHostBuilderExtensions

Extension methods to enable registration of the custom IFunctionExecutor implementation generated for the current worker.

WorkerHostBuilderFunctionMetadataProviderExtension

Extension methods to enable registration of the custom IFunctionMetadataProvider implementation generated for the current worker.

# Class FunctionExecutorHostBuilderExtensions

Namespace: Bots

Assembly: Bots.dll

Extension methods to enable registration of the custom IFunctionExecutor implementation generated for the current worker.

```
public static class FunctionExecutorHostBuilderExtensions
```

**Inheritance**

object ← FunctionExecutorHostBuilderExtensions

**Inherited Members**

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

# Methods

## ConfigureGeneratedFunctionExecutor(IHostBuilder)

Configures an optimized function executor to the invocation pipeline.

```
public static IHostBuilder ConfigureGeneratedFunctionExecutor(this IHostBuilder builder)
```

### Parameters

`builder` IHostBuilder

### Returns

IHostBuilder

# Class WorkerHostBuilderFunctionMetadataProviderExtension

Namespace: Bots

Assembly: Bots.dll

Extension methods to enable registration of the custom IFunctionMetadataProvider implementation generated for the current worker.

```
public static class WorkerHostBuilderFunctionMetadataProviderExtension
```

**Inheritance**

object ← WorkerHostBuilderFunctionMetadataProviderExtension

**Inherited Members**

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

## Methods

### ConfigureGeneratedFunctionMetadataProvider(IHostBuilder)

Adds the GeneratedFunctionMetadataProvider to the service collection. During initialization, the worker will return generated function metadata instead of relying on the Azure Functions host for function indexing.

```
public static IHostBuilder ConfigureGeneratedFunctionMetadataProvider(this
IHostBuilder builder)
```

#### Parameters

`builder` IHostBuilder

#### Returns

IHostBuilder

# Namespace Sachiv.Function

## Classes

SachivAatmaramBhidu

# Class SachivAatmaramBhidu

Namespace: Sachiv.Function

Assembly: Bots.dll

```
public class SachivAatmaramBhidu
```

**Inheritance**

object ← SachivAatmaramBhidu

**Inherited Members**

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

## Constructors

### SachivAatmaramBhidu(ILoggerFactory)

```
public SachivAatmaramBhidu(ILoggerFactory loggerFactory)
```

#### Parameters

`loggerFactory` ILoggerFactory

## Methods

### Run(HttpRequestData)

```
[Function("sachiv")]
public HttpResponseData Run(HttpRequestData req)
```

#### Parameters

`req` HttpRequestData

## Returns

HttpResponseData

# Namespace Tests.Src

## Classes

ConfigurationTests

LoggerTests

Tests for Logger This not an comprehensive test,just enough to make sure the logger is being created and returned correctly. Will not be adding more tests unless there is a change in the logger class, related to its format and on a custom functioanlity added to it.

# Class ConfigurationTests

Namespace: Tests.Src

Assembly: Bots.dll

```
public class ConfigurationTests
```

**Inheritance**

object ← ConfigurationTests

**Inherited Members**

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

## Methods

### TestHasBeenLoadedFlagIsSetToTrue()

Test has been loaded flag is set to true after static constructor is called.

```
[Fact]
public void TestHasBeenLoadedFlagIsSetToTrue()
```

### TestLoacalConfigIsLoaded()

Tests that the config is loaded from the local config file.

```
[Fact]
public void TestLoacalConfigIsLoaded()
```

#### Exceptions

Exception

### TestLoadConfigThrowsExceptionIfConfigFileContentsIsNull()

Exception is thrown if config file contents is null.

```
[Fact]
public void TestLoadConfigThrowsExceptionIfConfigFileContentsIsNull()
```

## TestLoadConfigThrowsExceptionIfConfigFileIsNotValidJson()

Exception is thrown if config file is not valid json.

```
[Fact]
public void TestLoadConfigThrowsExceptionIfConfigFileIsNotValidJson()
```

## TestLoadConfigThrowsExceptionIfConfigPathEnvVariableNotFound()

Exception is thrown if config path env variable is not found.

```
[Fact]
public void TestLoadConfigThrowsExceptionIfConfigPathEnvVariableNotFound()
```

## TestLocalConfigIsLoadedOnce()

Test that once config is loaded, it is not loaded again.

```
[Fact]
public void TestLocalConfigIsLoadedOnce()
```

# Class LoggerTests

Namespace: Tests.Src

Assembly: Bots.dll

Tests for Logger This not an comprehensive test,just enough to make sure the logger is being created and returned correctly. Will not be adding more tests unless there is a change in the logger class, related to its format and on a custom functioanlity added to it.

```
public class LoggerTests
```

**Inheritance**

object ← LoggerTests

**Inherited Members**

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

## Constructors

### LoggerTests()

```
public LoggerTests()
```

## Methods

### Logger_CreatesLogger_OnConstruction()

```
[Fact]
public void Logger_CreatesLogger_OnConstruction()
```

### Logger_ReturnsLogger_OnGetLog()

```
[Fact]
public void Logger_ReturnsLogger_OnGetLog()
```