KissLog

On-premises

Installation and Configuration Guide

Last revision 11.04.2019

Table of contents

Last revision 11.04.2019

About

This guide will walk through installation and configuration steps for deploying KissLog on-premises.

If you have any issues or questions, please feel free to send me an email to catalingavan@gmail.com and I will try to answer them as soon as possible.

Links

KissLog.net
https://kisslog.net/

KissLog SDK
https://github.com/KissLog-net/KissLog.Sdk

Last revision 11.04.2019

Prerequisites

Artifacts

1. KissLog.Backend.zip
2. KissLog.Frontend.zip

Services

KissLog on-premises requires the following services:

1. MongoDB Community Server (free)
2. MS-SQL Server or MySQL Community Server
3. IIS web server with .NET Core Runtime

Last revision 11.04.2019

Architecture

KissLog on-premises consists of two .NET Core 2.0 web applications:

KissLog.Frontend

User-interface application used by users (developers, IT administrators, application managers and implementation consultants) to visualise the captured errors, logs and other metrics data.

KissLog.Backend

Backend application responsible for managing the logs data. Exposes REST endpoints which can be used to save and to query the logs.

Storage providers

KissLog uses two storage providers:

1. MS-SQL / MySQL

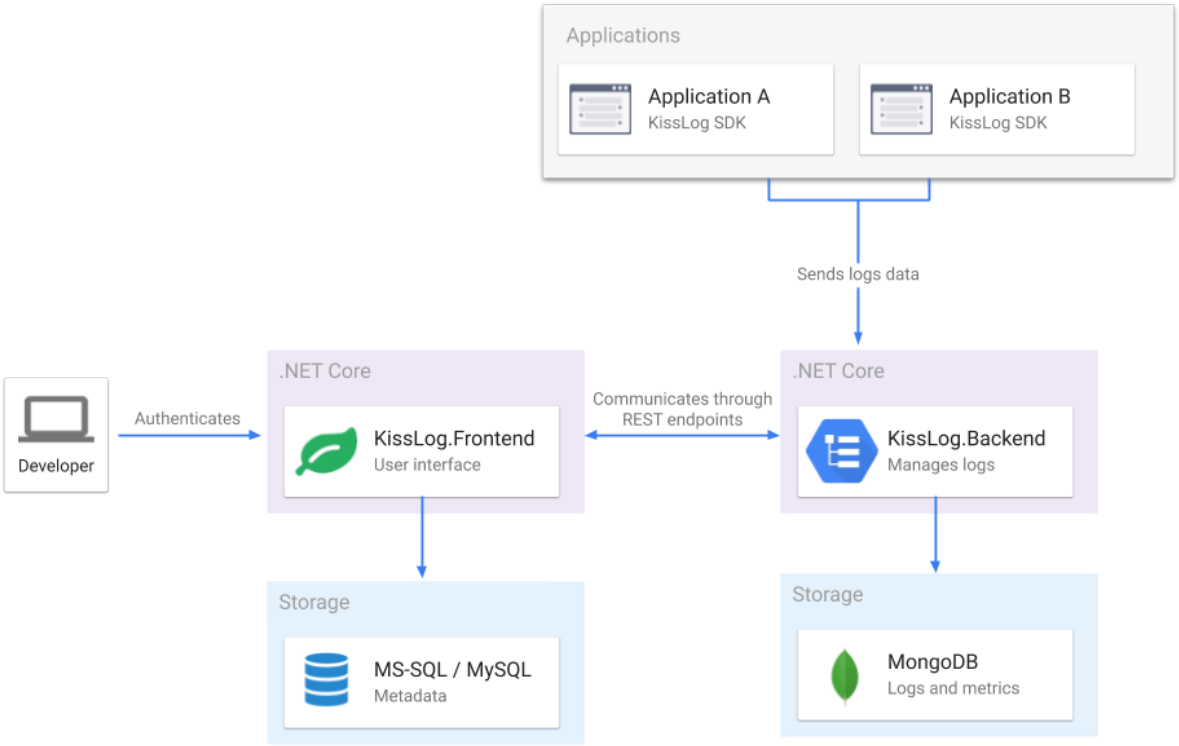
   Used for storing metadata information in the database, including information about organizations, applications and other user-interface related properties.

2. MongoDB

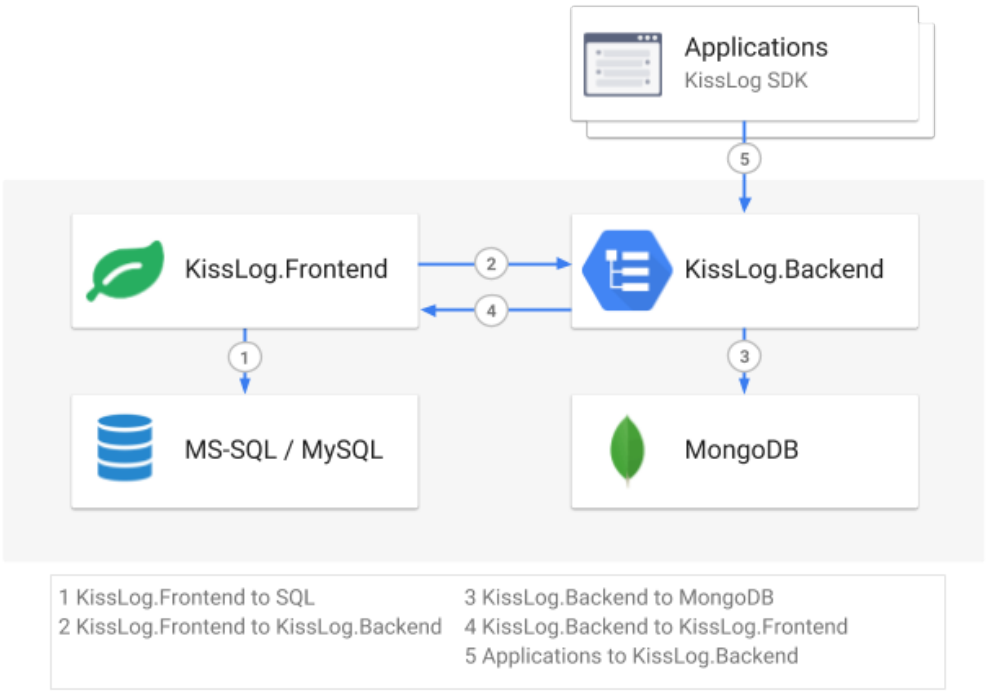
   Used for storing logs, captured exceptions and other metrics data.

Important
On the startup process, KissLog will automatically create the required SQL and MongoDB databases.

KissLog architecture



Network access



Last revision 11.04.2019

Installation

MongoDB

For installing MongoDB server, please check the official tutorial.

MS-SQL / MySQL

We will not cover the installation guide for these services. There is a high possibility that the existing system will already have a running instance of MySQL or MS-SQL server.

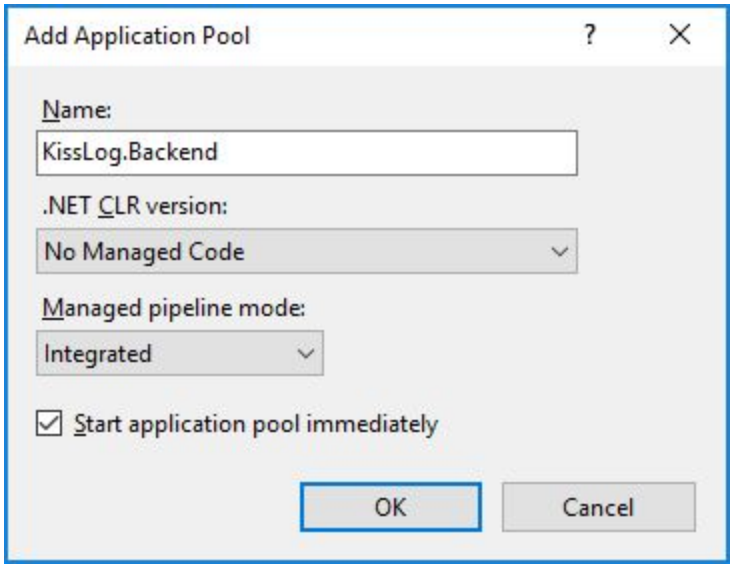
IIS web applications

1. On the IIS server, install .NET Core Runtime

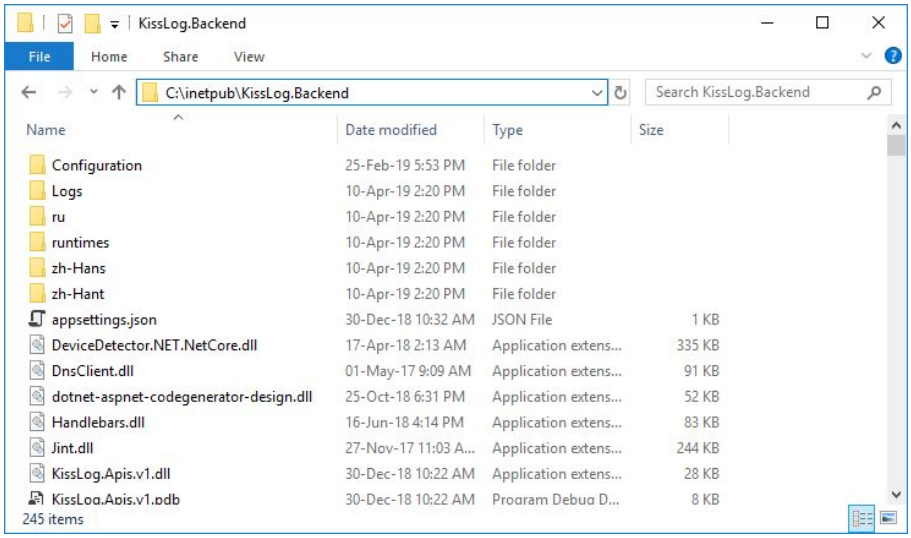
2. Create two IIS applications, one KissLog.Backend and the second KissLog.Frontend

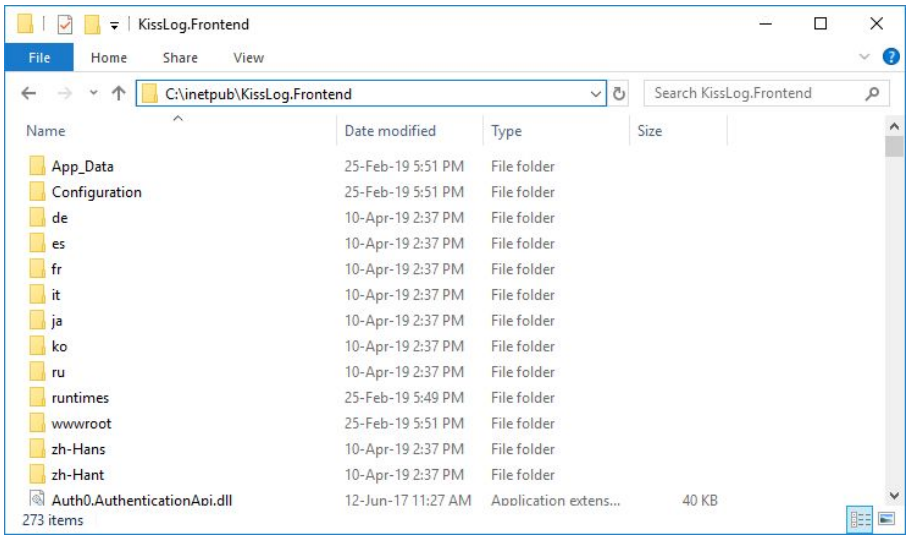
IIS write permissions
Make sure IIS has **write** permissions to the application folder.

3. Update the Application Pool settings for both of the applications to the following:



4. Extract into each site folder the corresponding deploy packages.





5. Update the Configuration.json file for both of the applications:
   KissLog.Backend\Configuration\Configuration.json
   KissLog.Frontend\Configuration\Configuration.json

Configuration.json backup
If you are updating an existing version of KissLog, make a backup of the Configuration.json file and replace it after the new files have been copied.

KissLog.Backend

Once KissLog.Backend has been deployed and configured on IIS, Configuration\Configuration.json file needs to be updated.

A complete example of the Configuration.json file can be found below.

Configuration.json example

```
{
  "MongoDb": "mongodb://user:password@192.168.44.45:27017/KissLogBackend",
  "KissLogFrontendUrl": "http://localhost:53057",

  "Alerts": {
    "IsEnabled": true
  },

  "SearchIndex": {
    "IndexInputStream": true,
    "KeyRange": [ 2, 100 ],
    "ValueRange": [ 2, 100 ]
  },

  "DeleteJob": {
    "TriggerIntervalInMinutes": 720,
    "RequestLogs": [
      {
        "HttpStatusCode": [ 0, 400 ],
        "Hours": 48
      },
      {
        "HttpStatusCode": [ 400, 500 ],
        "Hours": 96
      },
      {
        "HttpStatusCode": [ 500, 600 ],
        "Hours": 336
      }
    ]
  }
}
```

Last revision 11.04.2019

Configuration.json

Property	Required	Type
MongoDb	required	string
Mongo database connection string. https://docs.mongodb.com/manual/reference/connection-string/		
KissLogFrontendUrl	required	string
URL for KissLog.Frontend application		
Alerts	required	{Alerts}
Alerts configuration		
SearchIndex	required	{SearchIndex}
Text-search configuration		
DeleteJob	required	{DeleteJob}
Automatic delete job properties		

Alerts

Alerts configuration properties

Example

<pre>{
  "IsEnabled": true
}</pre>

Property	Required	Type
IsEnabled	required	boolean
Enables / disables the alerts functionality.		

SearchIndex

Configuration for text-search functionality

Example

<pre>{
  "IndexInputStream": true,
  "KeyRange": [2, 100],
  "ValueRange": [2, 100]
}</pre>

Property	Required	Type
IndexInputStream	required	boolean
Flag which determines if the Request.InputStream values should be indexed for text-search		
KeyRange	required	Array<integer>
Minimum and maximum length of the parameter name that should be indexed. Two values only.		
ValueRange	required	Array<integer>
Minimum and maximum length of the parameter value that should be indexed. Two values only.		

Last revision 11.04.2019

DeleteJob

Configuration for the automatic data deletion job

Example

```
{
  "TriggerIntervalInMinutes": 720,
  "RequestLogs": [
    {
      "HttpStatusCode": [ 0, 400 ],
      "Hours": 48
    },
    {
      "HttpStatusCode": [ 400, 500 ],
      "Hours": 96
    },
    {
      "HttpStatusCode": [ 500, 600 ],
      "Hours": 336
    }
  ]
}
```

Property	Required	Type
TriggerIntervalInMinutes	required	integer
Interval in minutes when the automatic delete job will trigger		
RequestLogs	required	Array<RequestLog>
Array for request logs data retention properties. At least one value must be provided. It is mandatory that both 0 and 600 status codes are provided - this ensures that all the request logs will be eventually deleted.		

RequestLog

Request logs data retention properties

Example

<pre>{
  "HttpStatusCode": [0, 400],
  "Hours": 48
}</pre>

Property	Required	Type
HttpStatusCode	required	Array<integer>
Two values only. Defines the range of the http status code. First value provides a **greater than or equal to >=** value. Second value provides **less than <** value. **Example:** [0, 400] will match request which have Response.StatusCode >= 0 and Response.StatusCode < 400		
Hours	required	integer
Data retention period (in hours) for the matching request logs.		

Startup

After Configuration.json file has been updated, KissLog.Backend application should be ready to use.

To initialize the startup process, make a single request to the application root URL ("/").

If the startup process went successful, a **200 OK "Running"** response will be returned.

This will bootstrap all the necessary components, including configuration validation and MongoDB database generation.

A **Logs** folder will be created in the application physical path.

IIS write permissions
On the first run, the Logs folder will be automatically created, regardless if the startup process went successful or not. If this folder has not been created, make sure IIS has **write** permissions to the application folder.

Updating Configuration.json
After each update of the Configuration.json file, the IIS application needs to be restarted in order to validate and load the new configuration properties.

Troubleshooting

Log files

All the application errors are logged under the Logs folder.
In case of any problems, this should be the first place to check, as they will give a hint about the current issue.

/api/status

In addition to text logs, KissLog.Backend exposes **/api/status** endpoint, which provides useful information about the current status of the application.

GET /api/status


```
{
    "success": true,
    "steps": [
        {
            "step": "CheckConfiguration",
            "messages": [],
            "success": true
        },
        {
            "step": "CheckMongoDb",
            "messages": [
                "MongoDb collections: 35"
            ],
            "success": true
        }
    ]
}
```

KissLog.Frontend

Once KissLog.Frontend has been deployed and configured on IIS, Configuration\Configuration.json file needs to be updated.

A complete example of the Configuration.json file can be found below.

Configuration.json example

```
{
   "ApplicationName": "Generic IT",

   "KissLogBackendUrl": "http://localhost:53335/",
   "KissLogFrontendUrl": "http://localhost:53057/",

   "Database": {
      "Provider": "SqlServer",
      "KissLogDbContext": "Data Source=192.168.44.45;Initial
Catalog=KissLog;UID=user;PWD=pass;"
   },

   "Smtp": {
      "Host": "smtp.sendgrid.net",
      "Port": 587,
      "UserName": "user",
      "Password": "pass"
   },

   "Slack": {
      "ProxyUrl": "http://192.168.177.23:8080"
   },

   "EmailTemplates": {
      "Logo": {
         "Src": "Generic-IT\\favicon.png",
         "Width": 16,
         "Height": 16
      },
      "From": {
         "Address": "logs@generic-it.com",
         "DisplayName": "Generic IT Logs"
      }
   },
```

Last revision 11.04.2019

```
   "UserInterface": {
      "Favicon": "/Generic-IT/favicon.png",
      "NavbarBrand": "<img src='/Generic-IT/navbarBrand.png' alt='Generic
IT' />",
      "HomePageLogo": {
         "Src": "/Generic-IT/homePageLogo.png",
         "Alt": "Generic IT"
      }
   },

   "Authorization": {
      "DefaultEmailDomain": "generic-it.com",
      "HS256Secret": "HS256-SECRET",
      "RS256CertificateFilePath": "Configuration\\Certificates\\public.xml"
   },

   "AnonymousAccessToken": {
      "IsEnabled": true,
      "ValidForSeconds": 7200,
      "HS256Secret": "HS256-SECRET"
   }
}
```

Configuration.json

Property	Required	Type
ApplicationName	optional	string
Display name of the on-premises KissLog application		
KissLogBackendUrl	required	string
URL for KissLog.Backend application		
KissLogFrontendUrl	required	string
URL for KissLog.Frontend application		
Database	required	{Database}
Provides information used to connect to the database server.		
Smtp	optional	{Smtp}
Provides information for connecting to SMTP server.		
Slack	optional	{Slack}
Slack configuration. Used for sending slack notifications.		
Authorization	required	{Authorization}
Credentials used to validate the JWT tokens used to authenticate the users.		
UserInterface	optional	{UserInterface}
Configuration for customizing the user-interface of the application.		
EmailTemplates	optional	{EmailTemplates}
Confguration for customizing the email templates which KissLog sends.		
AnonymousAccessToken	optional	{AnonymousAccessToken}
Configuration for temporary URLs, which can be anonymously accessed.		

Database

Provides information for connecting to database.

Example

<pre>{
  "Provider": "SqlServer",
  "KissLogDbContext": "Data Source=192.168.44.45;Initial Catalog=KissLog;UID=user;PWD=pass;"
}</pre>

Property	Required	Type
Provider	required	string
Specifies the database provider to use. Accepted values: • SqlServer • MySql		
KissLogDbContext	required	string
The database connection string used by the KissLog application. SqlServer example: "Data Source=192.168.44.45;Initial Catalog=KissLog;UID=user;PWD=pass;" MySql example: "server=192.168.44.45;port=3306;database=KissLog;uid=user;password=pass;Charset=utf8;"		

Smtp

SMTP configuration properties.

Example

<pre>{
  "Host": "smtp.sendgrid.net",
  "Port": 587,
  "UserName": "user",
  "Password": "pass"
}</pre>

Property	Required	Type
Host	required	string
Port	required	integer
UserName	optional	string
Password	optional	string

Slack

Slack configuration properties

Example

{
  "ProxyUrl": "http://192.168.177.23:8080"
}

Property	Required	Type
ProxyUrl	optional	string
The Proxy url which is used by the server - if any.		

Authorization

Holds information used for authenticating the users.

Part of the login process, KissLog authenticates the users by validating a JWT token.

When creating the JWT token, you must use the same **signature secret** provided in the Configuration.json file.

Example

<pre>{
  "DefaultEmailDomain": "generic-it.com",
  "HS256Secret": "HS256-SECRET",
  "RS256CertificateFilePath": "Configuration\\Certificates\\public.xml"
}</pre>

Property	Required	Type
DefaultEmailDomain	optional	string
If the JWT token doesn't contains a fully qualified domain address, the **DefaultEmailDomain** will be appended to the logged-in user.		
HS256Secret	optional*	string
The HS256 algorithm secret used to validate the JWT token.		
RS256CertificateFilePath	optional*	string
The RS256 certificate path used to validate the JWT token.		
*While **HS256Secret** and **RS256CertificateFilePath** are optional, at least one of them needs to be provided.		

Authorization JWT tokens can be created using third-party tools.

Below can be found an example of creating the JWT token using https://jwt.io/ website.

https://jwt.io/ example

Last revision 11.04.2019

AnonymousAccessToken

If enabled, users can share request log details using a temporary URL, which can be anonymously accessed.
The publicly accessible URL will be available from the user-interface.

Example

<pre>{
  "IsEnabled": true,
  "ValidForSeconds": 7200,
  "HS256Secret": "MY-HS256-SECRET"
}</pre>

Property	Required	Type
IsEnabled	required	boolean
Enables anonymous sharing.		
ValidForSeconds	required	integer
Specifies the expiry timespan of the anonymous URLs.		
HS256Secret	required	string
The HS256 algorithm secret used to generate and validate the anonymous URLs.		

UserInterface

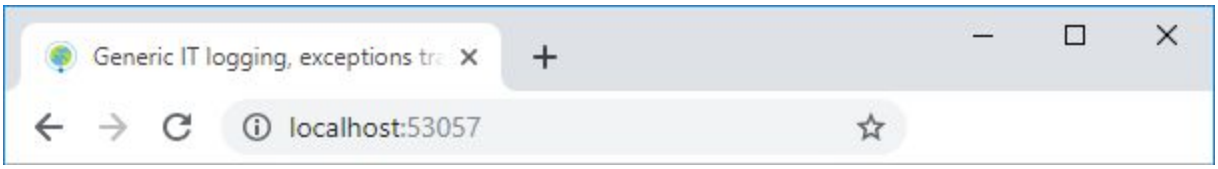
Holds information used to customize the user-interface of the KissLog application.
You can customize the logo image, favicon, and other user-interface related properties.

Example

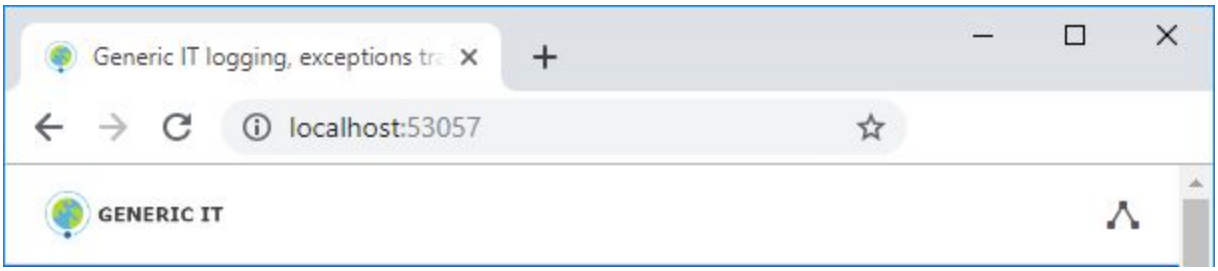
<pre>{
  "Favicon": "/Generic-IT/favicon.png",
  "NavbarBrand": "",
  "HomePageLogo": {
  "Src": "/Generic-IT/HomePageLogo.png",
  "Alt": "Generic IT"
  }
}</pre>

Property	Required	Type
Favicon	optional	string
The HTML favicon path		
NavbarBrand	optional	string
HTML to append on the navbar brand		
HomePageLogo	optional	{Image}
Home page logo image		

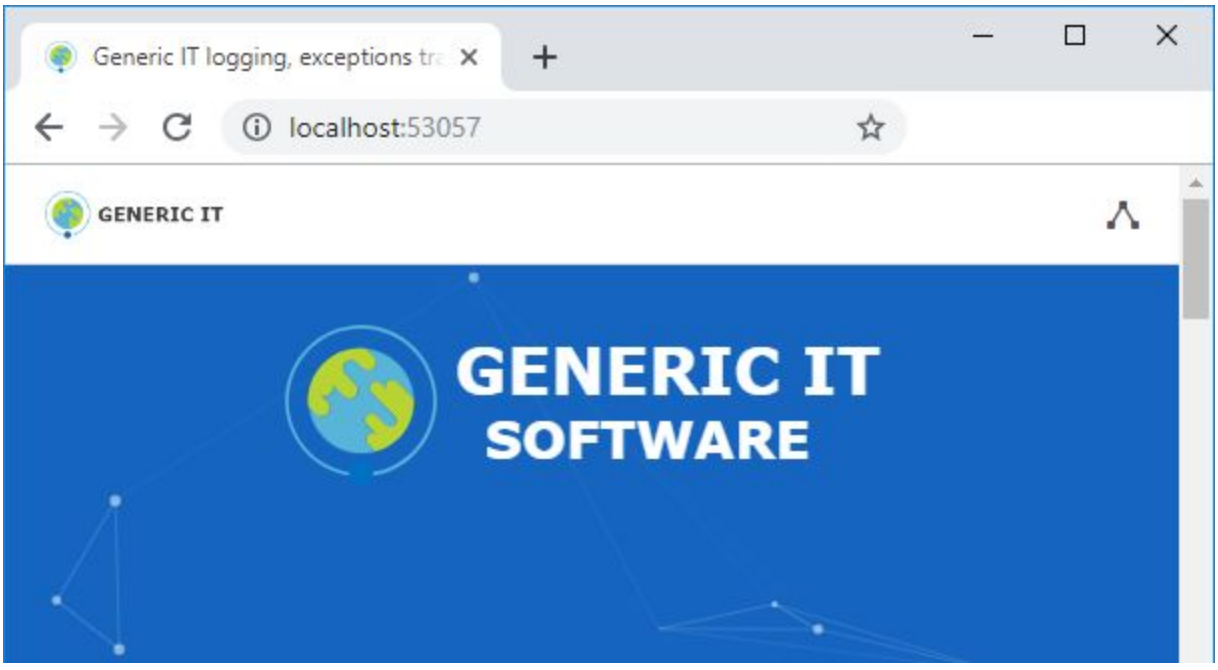
Favicon



NavbarBrand



HomePageLogo



Last revision 11.04.2019

EmailTemplates

Configuration for the emails templates.

Example

<pre>{
  "Logo": {
  "Src": "Generic-IT\\favicon.png",
  "Width": 16,
  "Height": 16
  },
  "From": {
  "Address": "logs@generic-it.com",
  "DisplayName": "Generic IT Logs"
  }
}</pre>

Property	Required	Type
Logo	optional	{Image}
The logo image used in the emails		
From	optional	{EmailAddress}
The email address from which emails are sent		

Image

Image properties.

Example

{
"Src": "/Generic-IT/HomePageLogo.png",
"Alt": "Generic IT",
"Width": 256,
"Height": 128
}

Property	Required	Type
Src	required	string
Alt	optional	string
Width	optional	integer
Height	optional	integer

EmailAddress

Email address properties.

Example

<pre>{
  "Address": "logs@generic-it.com",
  "DisplayName": "Generic IT Logs"
}</pre>

Property	Required	Type
Address	required	string
DisplayName	optional	string

Startup

After Configuration.json file has been updated, KissLog.Frontend application should be ready to use.

To initialize the startup process, make a single request to the application root URL ("/").

If the startup process went successful, a **200 OK** home page will be returned.

This will bootstrap all the necessary components, including configuration validation and SQL database generation.

A **Logs** folder will be created in the application physical path.

IIS write permissions
On the first run, the Logs folder will be automatically created, regardless if the startup process went successful or not. If this folder has not been created, make sure IIS has **write** permissions to the application folder.

Updating Configuration.json
After each update of the Configuration.json file, the IIS application needs to be restarted in order to validate and load the new configuration properties.

Troubleshooting

Log files

All the application errors are logged under the Logs folder.
In case of any problems, this should be the first place to check, as they will give a hint about the current issue.

/api/status

In addition to text logs, KissLog.Frontend exposes **/api/status** endpoint, which provides useful information about the current status of the application.

GET /api/status


```
{
    "Success": true,
    "Steps": [
        {
            "Step": "CheckConfiguration",
            "Messages": [],
            "Success": true
        },
        {
            "Step": "CheckDatabase",
            "Messages": [
                "KissLogDbContext created",
                "KissLogDbContext.Database.EnsureCreated() complete. New
database created: False"
            ],
            "Success": true
        },
        {
            "Step": "CheckKissLogBackend",
            "Messages": [
                "GET http://localhost:53335/api/status",
                "StatusCode: OK"
            ],
            "Success": true
        }
    ]
}
```


Last revision 11.04.2019