Developer Documentation Uniques

REST-API

The Rest API for the Project has been implemented with the standard .NET MVC Web API. Using the Usual “Controller” based Implementation. Meaning that every Object is handled by Controller and the given “MessageTypes” are defined by the Function Name.

The Definitions for the Routes and specific Limitations for the Parameters are defined in the WebApiConfig.

List of Functionality and Controllers

URL	Controller	Type	Desc
/api/users	users	GET, PUT, DELETE	Create, Retrieve or Delete Users
/api/users/where(%term%)	Search	GET	Searches for users in there attributes
/api/users/attributes	UserAttributes	GET, PUT, DELETE	Retrieves the Definitions for User Attributes or allows to alter them.
/api/users/attributes/categories	UserAttributeCategories	GET, PUT, DELETE	Retrieves the Definitions for User AttributeCategories or allows to alter them.
/api/users/authenticate	Authenticate	GET, PUT, DELETE	Allows the start a new Session or Delete them.
/api/users/{id}	Users	GET	Gets the user with the given Id
/api/users/{loginname}	Users	GET	Gets the user with the given Loginname
/api/users/{id}/attributes	UserAttributeValue	GET, PUT, DELETE	Allows to Get, Set or DELETE User Attributes for the given UserId
/api/users/{id}/images	Image	GET, FILE	Lists all Images for a given user or allows to upload images.
/api/users/{id}/images/{id}	Image	GET	Gets the Image as PNG
/api/users/{id}/images/{id}/thumbnail	ImagesThumbnail	GET	Gets the Image resized to a Thumbnail

Dependency Injection with Structuremap

StructureMap is used for Dependency Injection in the Project, so to reduce the dependencies between different Classes and improve the Lifetime of Objects, especially the Database Context.

Clean Code and Code commentation

To keep files and the code more readable the comments in the documents have been reduced to a minimum. Instead more time has been invested in keeping the functions small and the names of functions and classes self-describing.

Data Storage with Entity Framework 6 and MSSQL

The Data has been stored in an MSSQL Database. The connection string is present in the Project. Data Classes for the Entity Framework (ORM) are defined in "Uniques.Library.Data". The migration scripts to Up or Downgrade the Database are in "Uniques.DbMigrations".

JavaScript Templating with KnockOutJS

The Front End has been implemented as a Single Page Application. As a JavaScript Templating Engine "KnockOutJS" has been chosen. The Html Templates to be used by KnockOutJS will be loaded asynchronously or synchronously from the Server. The MVC Route for the Templates is "/Template"

Templates:

/Template/AdminUserAttributeCategories	Edit Mask for User Attribute categories
/Template/AdminUserAttributes	Edit Mask for User Attributes
/Template/LoggedIn	LoggedIn Display Mask
/Template/Login	Login form for authentication

FileUpload and Image Rescaling

The Files are Uploaded with a special WebApi Call and stored directly on the Server. The scaling operations for the Images are done after the upload.

FileLocations

For the File Locations the default .NET MVC Structure has been used, which means:

Files	**Location**
EF Data Models	Uniques.Library.Data
Javascript Classes	Uniques/Scripts/uniques
HTML Templates	Uniques/Views/Template
HTML "MasterPage"	Uniques/Views/Shared/_Layout.cshtml
Index	Uniques/Views/Home/Index.cshtml
CSS	Uniques/Content/Site.css

Abstraction

The data access has been capsuled into managers which ensure correct data or help retrieving the correct information and therefore reduce the amount of code in the controllers. Also caching may be implemented through the manager and with dependency injection.