xBuildSite – a GUI for creating xataface database applications

# Introduction

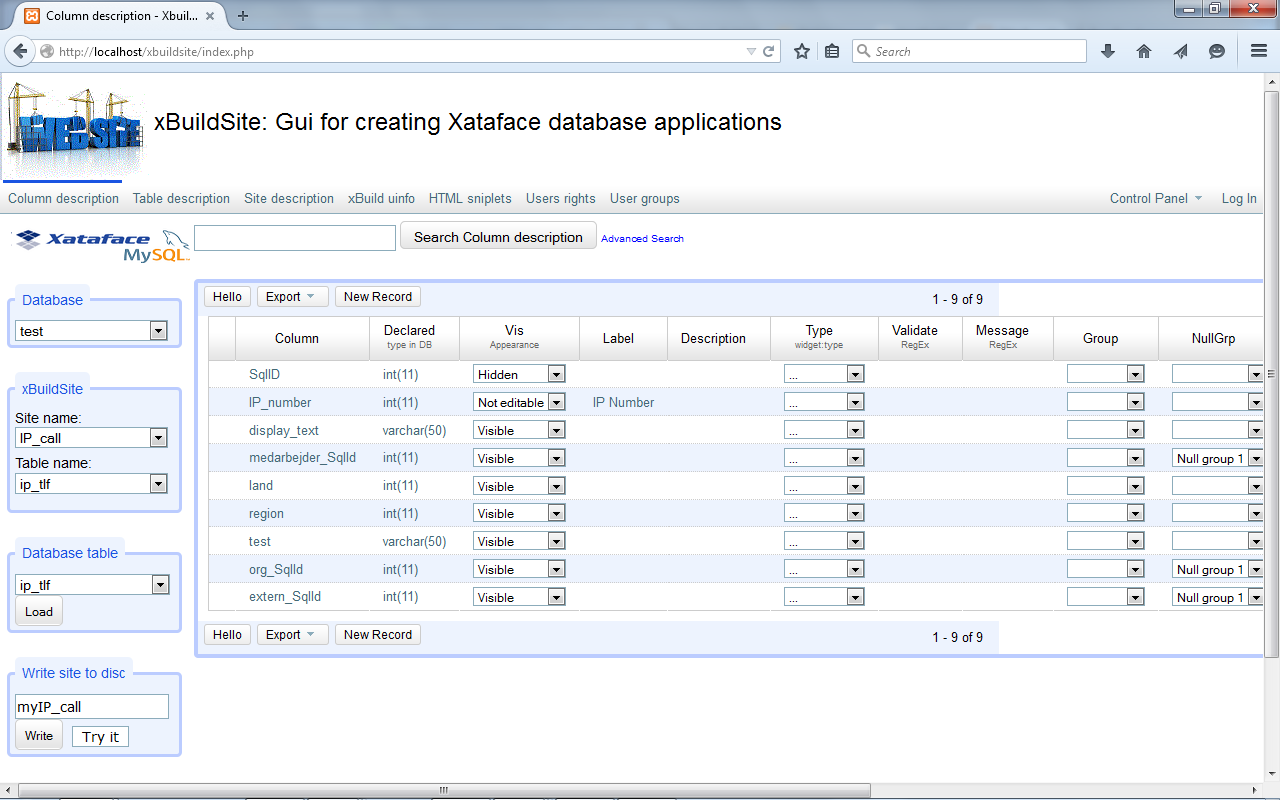
Many organisations use databases for storing vital data. In oder to be able to view and edit these data you are either faced with the option of writing customized database applications or to use some sort of customizable front-ends. In the later group – among others – are xataface witch is a customizable framework for building data-driven applications in PHP and MySQL. [(\*) A driver for MsSQL is also avaible] The use of xataface require a number of ‘.ini’ and other files to be written to define the database application – hereafter call a *site*. For some users it may be at hurdle to create these files. This is the background for developing xBuildSite.

xBuildSite is a GUI application for generating xataface site’s. xBuildSite write the entire site to disk without the need of doing any coding. xBuildSite is using the xataface framework – meaning all you can see in xBuildSite you are able to make I your own site.

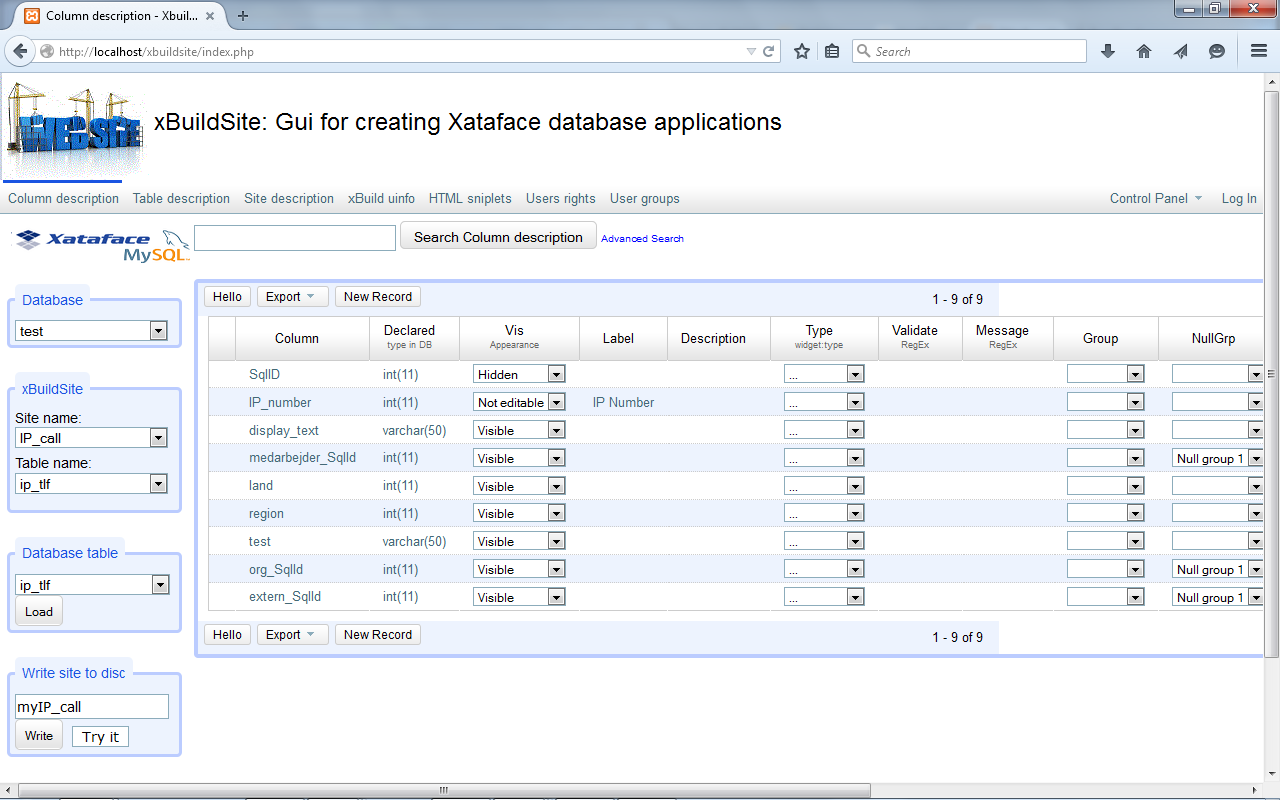
# xBuildSite userinterface

xBuildSite maintain a collection of site’s you have designed. One site include one or more database tables.

The following picture shows a typically xBuildSite session:



The left most column is where most of xBuildSite’s functionality is reached. A closer look at this is shown below:



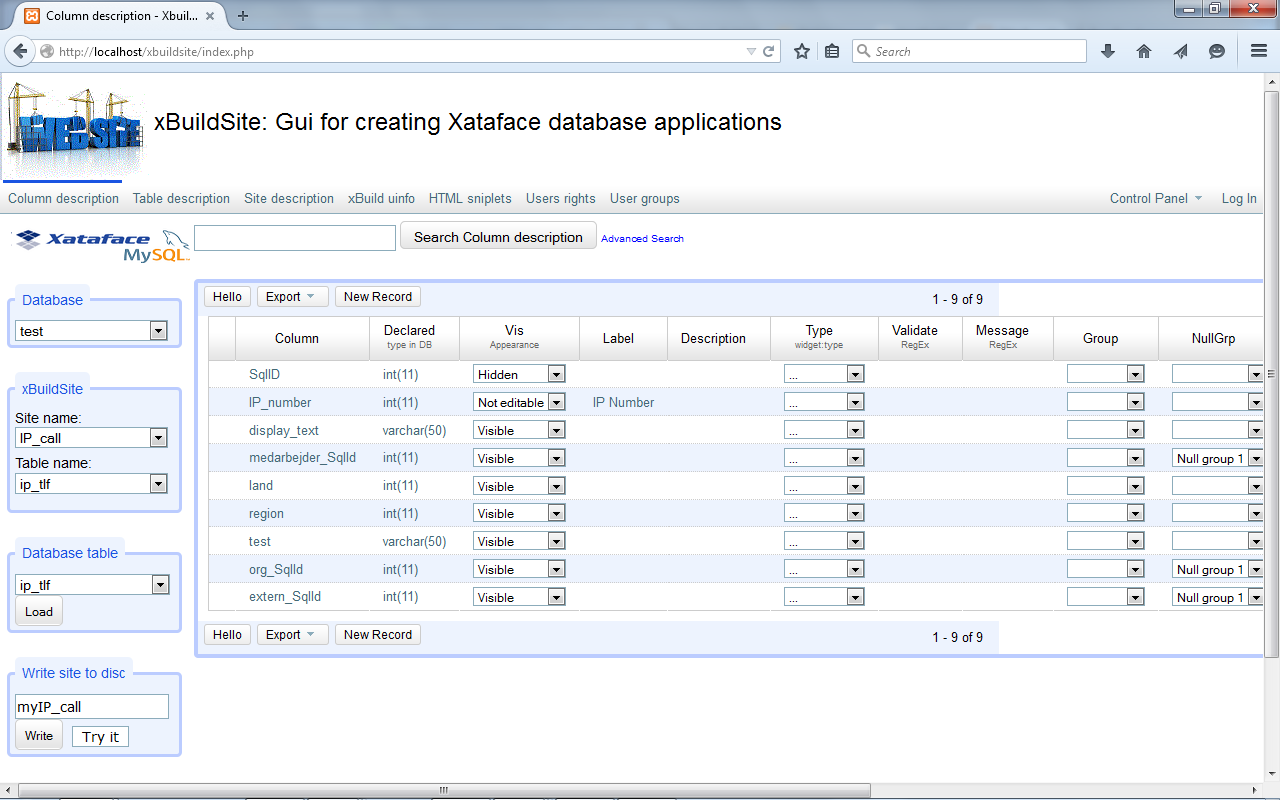
Xataface logo with indication of used database driver.

The current site information: ‘Site name’ and ‘Table name’

The name of the database table to load into xBuildSite

The name of the site on disc. Write the new site or try it out.

At (nearly) top of the screen a list of databases used in xBuildSite is shown. These tabs are where the varies aspects of the application is designed. The following picture shows this part of the xBuildSite application:



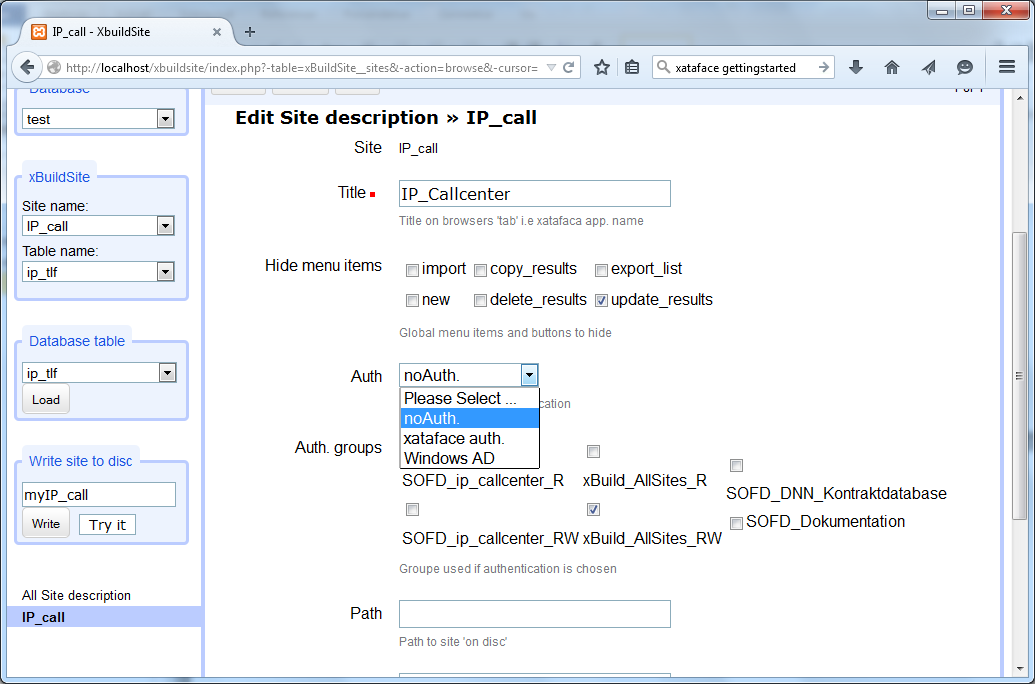
Properties of the columns in the current database table.

Properties of the current table.

Properties of the current site.

# Authentication

Xataface has the ability to use ’Permissions’ and ’roles’. Permissions include such as: ‘view’, ‘edit’, ‘delete’, ‘add new record’ etc. These can be applied on a ‘per table’ basis.



The roles can be one of the following:

|  |  |
| --- | --- |
| NO ACCESS | Role does not have any permissions. |
| READ ONLY | This role can view, show all, find, and navigate - which boils down to: this role can look but not touch. |
| EDIT | This role can do everything that READ ONLY can do plus edit existing records, add and remove related records, import records, and translate records.  In essence, this role can look, and touch - but not necessarily delete. |
| DELETE | Can do everything that EDIT can do, plus it can delete records. |
| OWNER | Same access as DELETE, but by a more intuitive name.  In theory OWNER may have different permissions than DELETE if actions were introduced that made sense for the OWNER role but not the DELETE role. |
| USER | Includes all of READ ONLY's permissions, but also has the ability to create new records. |
| ADMIN | Same as owner. |

xBuildSite comes with following authentication options:

noAuth No authentication used i.e. everyone has ‘ADMIN’ rights (read, write, delete, insert ..)

xataface auth. Use xataface authentication i.e. use the ‘role’ the logged-in user has.

Windows AD Use Windows Active Directory (AD)

xBuildSite comes with a database table used for authentication ‘xBuildSite\_\_users’. This table is defined in SQL as:

CREATE TABLE IF NOT EXISTS `xbuildsite\_\_users` (

`userId` int(11) NOT NULL,

`userName` varchar(32) CHARACTER SET latin1 COLLATE latin1\_danish\_ci NOT NULL,

`password` varchar(32) CHARACTER SET latin1 COLLATE latin1\_danish\_ci NOT NULL,

`role` enum('READ ONLY','NO ACCESS','ADMIN') CHARACTER SET latin1 COLLATE latin1\_danish\_ci NOT NULL DEFAULT 'READ ONLY',

`groups` varchar(256) DEFAULT NULL

)

The fields ‘userName’, ‘password’ and ‘role’ contains xataface authentication information (the role) in case ‘xataface auth.’ is selected.

## Application groups

The idea behind ‘Application groups’ is: Say you have a number of different database applications. The different users need to have different rights (read, read/write, no access ..) to the different applications. Some applications may belong to the same ‘category’ i.e. a user granted rights to one of the applications in this category would automatically have the same rights to the other applications in the same category. A list of applications grouped together might look like:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Group** | **Application** | **Users** | **Role** | **Application description** |
| Economy | Contracts | Controller | RW |  |
|  |  | Manager | R |  |
|  | Salery | Manager | RW |  |
|  |  |  |  |  |
| Employee | List staff | Any | R |  |
|  | Edit staff | Human rel. | RW |  |
|  |  |  |  |  |

This will give raise to the following 5 application groups that will need to be defined:

|  |  |  |
| --- | --- | --- |
| **App. groups** | **Role** | **Application group description** |
| Contracts\_RW | RW | Read/Write contracts |
| Contracts\_R | R | Read only contracts |
| Salery\_RW | RW | Read/Write salery data |
|  |  |  |
| View staff\_R | R | Read staff data |
| Edit staff\_RW | RW | Read/Write staff data |
|  |  |  |

Looking at the different users they need to be member of the following groups:

|  |  |
| --- | --- |
| **User** | **Member of** |
| Controller | Contracts\_RW  View staff\_R |
| Manager | Contracts\_R  Salery\_RW |
| Human rel. | Edit staff\_RW |
| Any | View staff\_R |

### Windows Active Directory

Many organisations choose to use Windows Active Directory (Win. AD) as a way to control and grant users rights to different software applications etc. When an organisation use Win. AD a user will normally login to his/her computer using Win. AD username/password.

In the case you want to use ‘Windows AD’ authentication

Resten:

|  |  |  |
| --- | --- | --- |
| **Group** | **Role** | **Role description** |
| AllApp\_RW | USER | ‘USER’ for all applications |
| AllApp\_R | READ ONLY | ‘READ ONLY’ for all applications |
| App1\_RW | USER | ‘USER’ for ‘App1’ |
| App2\_R | READ ONLY | ‘READ ONLY’ for ‘App 2’ |
| App2\_RW | USER | ‘USER’ for ‘App2’ |
|  |  |  |

To the different applications are then attached some of the above groups. The following table shows how this attachment might look like:

|  |  |
| --- | --- |
| **App:** | **Groups** |
| App1 | AllApp\_R  App1\_RW |
| App2 | App2\_R  App2\_RW |
| MyApp | AllApp\_RW |

Likewise are users attached to application groups. This gives the individual user rights (role) for the different applications. The following table shows how the users might be attached to the application groups:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Application** | **Users** | | | |
|  | **User1** | **User2** | **User3** |  |
| AllApp\_RW | x |  |  |  |
| AllApp\_R | x | x |  |  |
| App1\_RW |  |  | x |  |
| App2\_R | x |  |  |  |
| App2\_RW |  |  | x |  |
|  |  |  |  |  |

Let us take an example ‘User1’:

App1 Groups: ‘AllApp\_R’, ‘App1\_RW’. The user belongs to ‘AllApp\_R’ => role = READ ONLY.

App2 Groups: ‘App2\_R’, ‘App2\_R’. The user belongs to ‘All2\_R’ => role = READ ONLY.

MyApp Groups: ‘AllApp\_RW’ The user belongs to ‘AllApp\_RW’ => role = USER.

This will give the different users the following role’s to the different applications:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Users** | | | |
| **Group** | **User1** | **User2** | **User3** |  |
| App1 | READ ONLY |  |  |  |
| App2 | READ ONLY |  |  |  |
| MyApp | USER |  |  |  |

that you define a number of groups pictured in the following:

These are explained in the following subsections.