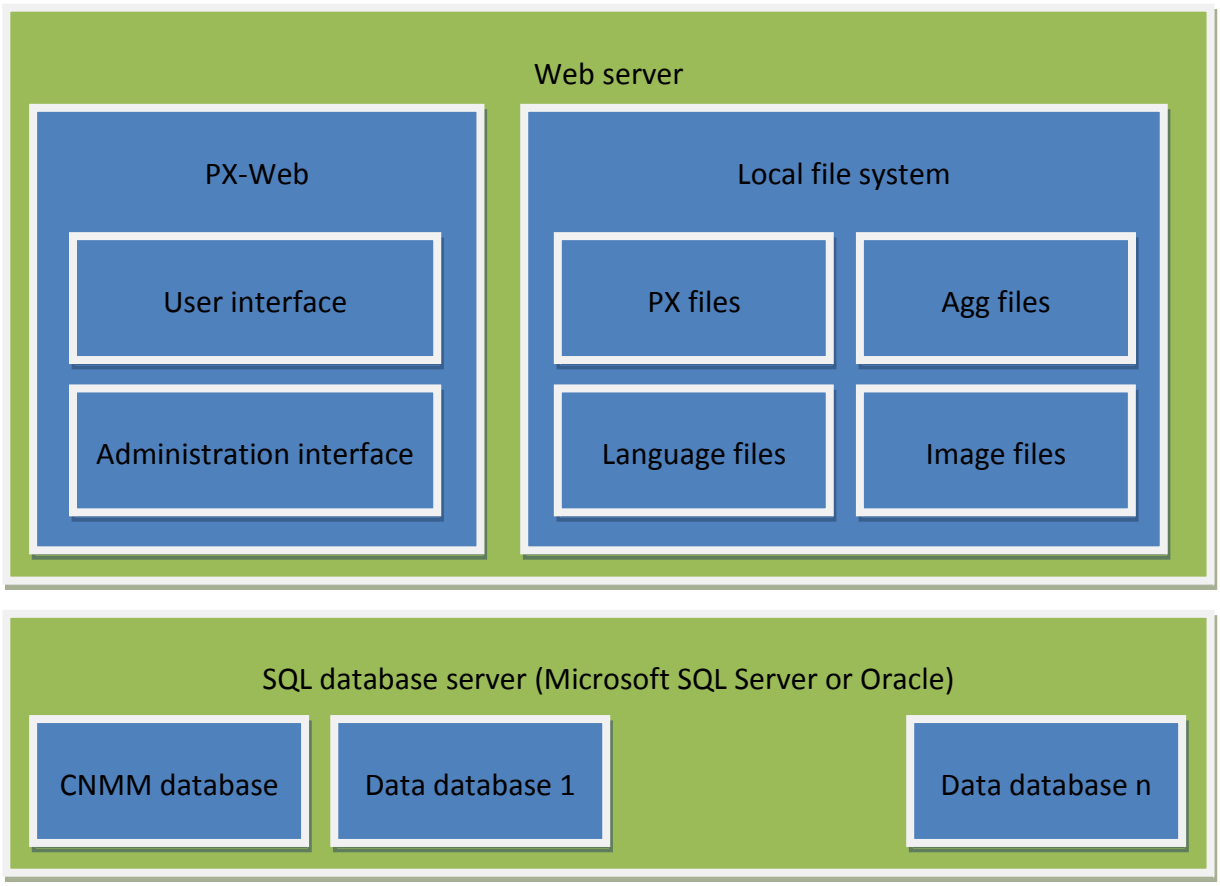
PX-Web system description

Revision history

Version 1.0	First version

PX-Web system diagram



PX-Web system description

General

The PX-Web application consists of two parts:

- Administration interface - The Administration interface is used by PX-Web administrators to manage and maintain their PX-Web installation.
- User interface - The User interface is the part of the application that is exposed to the end users, the actual dissemination application.

The statistical data that should be disseminated could either be stored in PX file located locally on the Web server or remote in a SQL server in a database having the Common Nordic Meta Model. PX-Web is also able to combine the two types of data sources.
The name and type of the SQL Server is specified in the SqlDb.config[1] file

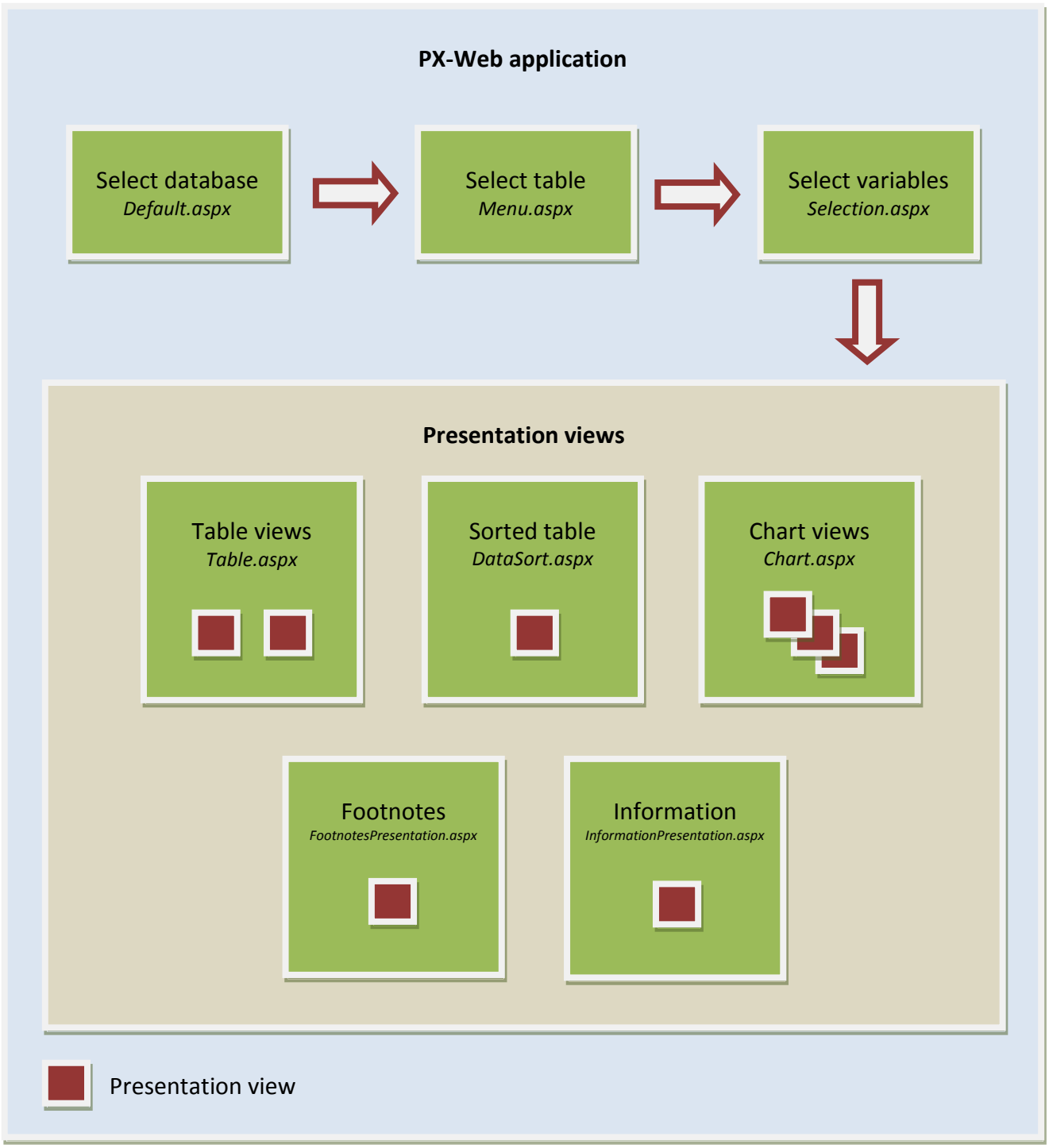
PX-Web permissions

The user account that runs PX-Web must have

- Modify rights to setting.config to be able to modify and save settings from the *Administration interface*.
- Modify rights to the Language folder and all of the folder content, so that new language can be added and existing language string be modify from the *Administration interface*.
- Modify rights to the Menu.xml files that are in the PX database folder, so that the content can be updated from the *Administration interface* when generating the database description file.

[1] The name and location of this file can be configured in web.config.

Application flow



When database, table, variables and values have been selected the resulting table can be displayed in various presentation views. The names of the web pages holding the presentation views can be configured in web.config[2]

Presentation views

Each presentation view displays the selected data in a specialized way. Examples of presentation views are tables with different layouts, different types of chart, footnotes and information about the table.

A presentation view belongs to a special web page which is illustrated in the application flow picture above. For example the table presentation views belong to the table web page, the chart presentation views belongs to the chart web page and so on. Which presentation view to display on the web page is given by the *layout* parameter in the URL query string.

[2] The *pages* setting in the *web.controls* section controls the names of the web pages holding the presentation views

Administration tool

The Administration tool is used by PX-Web administrators to manage and maintain their PX-Web installation.

The PX-Web settings[3] are managed from the Administration tool.

The Administration tool also contains tools for generating database files, management of language files and tools for reloading system data.

The Administrator tool is protected by username and password. The default username and password is *admin* and *pwd*. Per default the Administration interface is also protected by an IP filter and only the local loopback address is allowed access it.

The PX-Web Administration menu is generated from a sitemap file. Which sitemap file to use is defined in web.config.
Each node in the sitemap has a title which is the name of the link to display in the PX-Web Administration menu. The titles are not the link names in plain language but rather the keys to the according sentences in the language files.

For further description of the Administration tool see the PX-Web Configuration instructions.

[3] The PX-Web settings are stored in the file Settings.config

PX-Web File structure

Root (Application root)
- **Admin** (Administration interface)
  - Admin.master
  - Default.aspx
  - Settings-General-Databases.aspx
  - Modules.aspx
  - …
  - Tools-GeneratePxDatabase.aspx
- **Bin**
- **Logs**
- **Resources**
  - **Images**[4] (Contains all images)
  - **Languages**[5] (Language files)
  - **PX** (Contains PX database related content)
    - **Aggregations**[6] (Aggregation files)
    - **Databases**[7] (Contains the PX databases)
  - **Scripts** (javascripts)
  - **Styles** (css)
- Default.aspx
- Menu.aspx
- Selection.aspx
- Table.aspx
- Chart.aspx
- FootnotesPresentation.aspx
- InformationPresentation.aspx
- PxWeb.master
- PresentationMaster.aspx
- CommandBarCustomPlugins.config
- setting.config
- SqlDb.config[8] (SqlParser configuration file)
- Web.config

[4] Location can be changed in the settings.config file in section Settings.General.Paths.ImagesPath
[5] Location can be changed in the settings.config file in section Settings.General.Paths.LanguagesPath
[6] Location can be changed in the settings.config file in section Settings.General.Paths.PxAggregationPath
[7] Location can be changed in the settings.config file in section Settings.General.Paths.PxDatabasesPath
[8] The name and location for this file can be configured in web.config

Application settings

Settings overview

The administrator can configure how the PX-Web installation shall appear and behave to the users. The application settings are managed from the Administration tool. The PX-Web settings are stored in the Settings.config file.

Settings architecture

All settings are grouped into sections. Each section is defined by a .Net interface having read only properties that have one-to-one correspondence to the settings. The section interfaces are implemented in internal classes that are only accessible from the same assembly. These classes also give the ability to update the settings during run-time from within the administration interface of PX-Web without exposing that same ability to external code. Each section class is responsible to read and write their own settings from/to the configuration file. E.g. when the section is created a XmlNode object is passed as a parameter in the constructor. The constructor code reads the settings from the XmlNode and sets its properties accordingly. It also creates any other sub sections and pass in their respective sub XmlNode to their constructors. When settings are saved the Save function on the section is called and a XmlNode is passed as a parameter. Each Save method is responsible to write their properties to the XmlNode and call all of its sub sections Save method.

Reading settings

All of the settings is accessed through the Settings class. The Settings class is implemented according to the singleton pattern and the Current property exposes the Settings instance that PX-Web use to access the different settings. For example the *LanguagePath* setting of the *Paths* section which is a subsection of the *General* section is accessed in the following way:

```
string p = PXWeb.Settings.Current.General.Paths.LanguagesPath;
```

Updating settings

Updates of the settings at run-time should only be done from the administration interface using the following process:

1. Call the BeginUpdate method on Settings. This will create a copy of the current Settings object and expose through the NewSettings property on Settings. If BeginUpdate returns false it is not secure to proceed. The operation is then aborted.
2. Make all changes to the NewSettings and its children objects.
3. Call the Save method on Settings. This will save the settings to the configuration file and also update the current settings.
4. Call the EndUpdate method on Settings. This will clean up the state after the update.

Since the Setting only expose the interfaces which in turn only exposes the read only properties we have to manual cast them to their implementation classes so that we can change the properties.

In code the update of the *Paths* settings section could look like this:

```
// Create the NewSettings object by reading the values for all setting-sections
// from the configuration file.
// Returns true if it is safe to do the update (meaning that no one else is
trying
// to update right now).
if (PXWeb.Settings.BeginUpdate())
{
   try
   {
      // Thru PXWeb.Settings.NewSettings.General.Paths we only reach a readonly
      // interface, therfore we have to do a manual cast to the corresponding
      // settings-class to be able to update the settings.
      // Note that we are working against the NewSettings object!
      PXWeb.PathsSettings paths;
      paths = (PXWeb.PathsSettings)PXWeb.Settings.NewSettings.General.Paths;

      // Set the new values of the settings
      paths.LanguagesPath = txtLanguagePath.Text;
      paths.ImagesPath = txtImagePath.Text;
      paths.PxDatabasesPath = txtDatabasePath.Text;
      paths.PxAggregationsPath = txtAggregationPath.Text;

      // Save all the setting-section in the NewSettings object to the
      // configuration file. The settings are also exposed to PX-Web (through
      // PXWeb.Settings.Current)
      PXWeb.Settings.Save();
   }
   finally
   {
      // Remove the PXWeb.Settings.NewSettings object
      PXWeb.Settings.EndUpdate();
   }
}
```

Menu builder

Background

From the Administration tool the administrator can generate database files. This is a description of how the menu building process is done for file system based databases such as PX databases.

The system consists of three main components

- The Spider – This component works as an spider, it iterates through the files and folders and calls the appropriate handler and the pushes the information from the handler to the registered builders.
- The handlers – This component has different implementations depending on the file type. There purpose is to read the necessary information from a file of a certain format and provide that to the engine.
- The builders – This component is responsible for building something from the information given by the handler. Default there will be a builder that builds the Menu.xml file for the databases but there will be possibilities to create builders that generate RSS feeds etc.

The current implementation

The current implementation of the menu generation consist of the following classes

Spider
There is just one implementation of this class which are fairly generic in respect of that it can be used to compose different type of builder. The Spider iterates through the files system recursively. For each new folder it will call the BeginNewLevel/EndNewLevel on the registered builders and call NewItem for each file it finds.

Handlers
As stated earlier the purpose of the handlers is to read en interpret the files found by the spider and to provide a information object to the spider.

AliasFileHandler
This handler handles .alias files that is found by the spider. It parses the file name of the alias file to extract the alias and creates a AliasItem object setting the alias and the language. If the language is missing from the filename the language will be set to default language of the PXWeb installation.

LinkFileHandler
This handler handles .link files that is found by the spider. It parses the content of the link file to extract the link and the link "description" and creates a LinkItem object setting the link url, text and language. If the language is missing from the filename the language will be set to default language of the PXWeb installation.

PxFileHandler

This handler handles .px files that is found by the spider. It parses the metadata part of the px file and creates a PXMeta object containing the parsed metadata.

Builders

There is currently on one builder implemented which builds the menu.xml file for each database.

MenuBuilder

The diagram tries to explain how the MenuBuilder operates. From the initial state the BeginBuild method is called. This will allocate root nodes for every language and set the builder in a waiting state ready for adding new content to the node tree.