**Creating New Pages in MinistryPlatform**

**Background**

Read the ThinkMinistry KB articles at <http://www.thinkministry.com/kb/ministryplatform/pages-and-sub-pages> for detailed information on creating new pages. It is important to determine if a new page is needed prior to embarking on creating the items necessary for a new page. It is also very important to plan the data structures properly. Taking time to ask the important questions addressed in the ThinkMinistry knowledge base articles and proper planning will create lasting data structures that need little modification.

Before creating new tables in a production environment, be sure to create a test system where all changes to production can be thoroughly tested before implementation.

 **Table Creation**

1. Tables name should use the following conventions:
	1. All user created tables that extend MP should begin with a prefix to differentiate those tables from tables created by ThinkMinistry.
	2. Should be plural (e.g., **Contacts** to store each contact, **Events** to store each event).
	3. Should use underscores rather than spaces between words. Do not use any special characters (e.g., “/” or “-“).
2. Tables should have a field that uniquely identifies each record. This field will be the primary key (PK) for that table. The PK should be named the singular version of the table name followed by “\_ID”. For example, the **Contacts** table will have a PK named **Contact\_ID**.
3. Tables will most likely be related to existing tables to leverage existing information. In such cases, fields will need to be created to relate to existing tables. Such fields will have a foreign key that relates to the primary key of associated tables.
4. Plan and document the fields and the data types and properties for each field.
5. Each table will need to have a field with the following properties in order to work properly with existing tables created by ThinkMinistry:
	1. Field Name: **Domain\_ID**\*\*
	2. Data Type: **int**
	3. Allow Nulls: **No**

\*\*Note: A table without a Domain\_ID field will be treated as a system table. Only users where **Setup\_Admin = TRUE** can see/edit the values in a system table. Reserve this for drop down list only table/pages that have few and rarely changing values (Ex. Genders).

**Page Creation**

1. Create each page in MP under the **System Setup** folder.
2. Complete the fields on the new page as appropriate to your table. A few fields are noted here:
	1. **Default List View** – The view that is shown by default displaying selected data in the table. Fields in this list will be available as tags that can be used in other parts of MinistryPlatform.
	2. **Selected Record Expression** – This is the “friendly name” of the record. It’s what will appear when that record is used as a foreign key (drop down list) on another page. In other words, it’s the way a single record expresses itself on a page.
	3. **Start Date Field and End Date Field** – Necessary when a user desires the Calendar view of a page. The calendar view uses the start and end date in a record in order to plot it correctly on the calendar.
	4. **Default View** – When left blank, the default view will be all records in the table. Otherwise a user created view can be used to display the page.
	5. **Display Copy** – Allows users to copy an existing record.
3. Assign the page to a page section.
	1. Create a new page section if necessary (page sections are the folders that contain pages in the left window pane in MP).
		1. Open **System Setup** -> **Page Sections**
			1. Assign a name and a view order.
	2. Add the page to the appropriate section.
		1. Open **System Setup** -> **Page Sections** and open the desired page section.
		2. On the **Pages** subpage, click **New** to add the page to the desired section.

**View Creation**

Views can be created to display page data with the desired sort order and columns most users will appreciate.

1. Go to **System Setup -> Page Views**. Click **New**.
2. Configure view by editing view fields:
	1. **Page** - The page in which the view is available.
	2. **View Clause** – The condition used to filter the data.
	3. **Order By** – The field used to sort the data.
3. Add the view to the page.
	1. **System Setup** -> **Pages**. Open the page that will contain the view.
	2. Click on the **Page Views** subpage and add the view.
	3. If the view will be the default view for the page, then edit the page and change the **Default View** value to the name of the view.

**Security Configuration**

1. Give permissions to users to view and edit pages.
	1. Open **Administration** -> **Security Roles**.
		1. Edit each security role that needs to access the new page.
		2. NOTE: It is a good practice to give full rights to the **Administrators** role.
2. Log out and log back in to verify the page can be seen properly.
	1. Impersonate users needing to access the new page if necessary.

**Testing**

Once the tables and pages have been created, testing should be done to ensure everything has been created properly.

1. Data should be entered into each table to make sure new records can be added, edited and deleted. Data should be entered into every field of the new page.
	1. It is important to add a *variety* of items to the page *over time*. Doing so will help to ensure all the proper fields have been created to capture the data intended for the page. If time is not taken, you may discover that new fields need to be added after the page is in production.
2. Make sure the data displays properly in every view that is applicable to the page (i.e., Grid, Calendar, Gantt and Images views).
3. Check the pages that are related to the newly created tables. In most cases, when a new table is created there is a relationship to another table containing relevant data. For example, a table may have a relationship to the Contacts or dp\_Users table. In such cases make sure a record in the related table can be added and updated. Repeat this testing process for ALL related tables.
4. Test all the verbs for the newly created page to ensure they work correctly.
	1. Keep in mind an administrator needs to give the tester full access to all verbs for testing purposes.
5. Test any views created for the page.

**Transferring to Production**

*\*\*\*Always perform a backup before making any changes to the production database.\*\*\**

If a significant amount of data has been added during testing, the tables may need to be imported into the production SQL Server database. When moving the tables into production, there are three primary concerns: creating the tables, importing the data and creating the proper relationships between tables (see note below). Listed below is one approach:

1. Create a script of the table(s) to transfer to production (see **Table Creation Notes** below).
	1. In the source database, right-click the table to script and select **Script table as… -> CREATE to…** Then select the where to send the T-SQL script.
	2. Go through the wizard to select the tables to script.
	3. Run the script.
2. Import the data into the tables created in step 1 into the production server.
	1. Right-click the destination database and select **All Tasks -> Import…**
	2. Go through the wizard and select the tables containing the source data.
		1. Select the option **Copy data from one or more tables or views**.
		2. Once the tables are selected, click on each table and click the **Edit Mappings…** button.
	3. Select the **Enable identity insert** checkbox (see **Identity Insert Notes** below).
		1. Note: Since each table should be created with a primary key using the identity property, primary key values are inserted automatically by SQL Server when a new record is created. By enabling this option, SQL Server ignores this behavior while the data is being imported.
	4. If you wish to see the data to be imported, click the Preview… button.
	5. Once all the values for the wizard have been specified, run the wizard.
	6. Check the production MinistryPlatform application to ensure the application sees the data properly.
3. Follow the steps to create sections, pages, views, etc. as was done on the test server.
4. Grant only access to an administrator and test all new additions to the production server. Once testing is complete, permissions can be given to users.

**Table Creation Notes**

Creating the tables and relationships in the proper order are important. If there are any lookup tables to import (i.e., child tables), these tables must be created first with their data. The data in the parent table has a foreign key referencing data in the child table. If the data the key is referencing does not exist, you will get an error along the lines of “INSERT statement conflicted with the FOREIGN KEY constraint”. For example, let’s say you have two tables to import “Network Devices” (the parent table) with a foreign key that references a lookup table (or child table) titled “Network Device Types”. If the Network Devices table has a reference to a device type that is NOT contained in the Network Device Types table, the a fore mentioned error will return during the import process. This maintains data integrity between tables. This being said, when running CREATE TABLE scripts and importing data, run the scripts and import that data for the child table first, then run the script and import for the parent table second.

**Identity Insert Notes**

By default once a column has been assigned the identity property, SQL Server does not allow explicit values to be inserted into it. Any attempt to enter a manual value will result in the error: “Cannot insert explicit value for identity column in table ‘[Table name]’ when IDENTITY\_INSERT is set to off.”

At times, you may want to insert a specific value into an IDENTITY column. For example, you may want the first row inserted into the table to have an IDENTITY value of 101, rather than 1. Or you may need to reinsert a row that was deleted by mistake. In these cases, you must set a session property to allow the table accept explicit values in the identity column. In order to set this value, run the following statement:

SET IDENTITY\_INSERT [Table Name] ON

Once the values have been inserted, you can turn off the IDENTITY\_INSERT statement by running the following command:

SET IDENTITY\_INSERT [Table Name] OFF

You can also simply close your connection to the database as this property is associated with your session connection.

**Examples**

**Lock Up Schedule and E-Phone Schedule Extension Example**

An example of a church that schedules staff members to lockup the building and providing emergency phone coverage to the congregation.

**Approach**

Create a page titled **Lockup Schedules** and **EPhone Schedule** in the **Staff Stuff** section. The purpose of these tables are to document and publish the lock up schedule and e-phone schedule for all staff. These pages will have a calendar view. The information to track is the staff person locking up and on e-phone each month. One staff person will have many schedules throughout the year to participate in locking up the building and answering the emergency phone.

**Table Creation**

Create tables titled **RMC\_Staff\_Lockup\_Schedules** and **RMC\_Staff\_EPhone\_Schedules**.

|  |  |
| --- | --- |
| **Table Name** | **Field Name/Data Types** |
| RMC\_Staff\_Lockup\_Schedules | RMC\_Staff\_Lockup\_Schedule\_ID - PK, int, Identity, NOT NULL, increment by 1.Contact\_ID - int, NOT NULL, FK to Contacts tableStart\_Date – datetime, NOT NULLEnd\_Date – datetime, NOT NULLNotes – nvarchar(500)Domain\_ID – int, NOT NULL |
| RMC\_Staff\_EPhone\_Schedules | RMC\_Staff\_EPhone\_Schedule\_ID - PK, int, Identity, NOT NULL, increment by 1.Contact\_ID - int, NOT NULL, FK to Contacts tableStart\_Date – datetime, NOT NULLEnd\_Date – datetime, NOT NULLNotes – nvarchar(500)Domain\_ID – int, NOT NULL |

**Page Creation**

Create pages titled **Lockup Schedules** and **E-Phone Schedules** with the following properties:

|  |  |
| --- | --- |
| **Page** | **Properties** |
| Lockup Schedules | **Section** – Staff Stuff**Default Field List** – Contact\_ID\_Table.[Display\_Name] , Contact\_ID\_Table.[Mobile\_Phone] , RMC\_Staff\_Lockup\_Schedules.[Start\_Date] , RMC\_Staff\_Lockup\_Schedules.[End\_Date] , RMC\_Staff\_Lockup\_Schedules.[Notes]**Selected Record Expression** - Contact\_ID\_Table.[Display\_Name] + ‘ ‘ + Contact\_ID\_Table.[Mobile\_Phone]**Start Date Field** – RMC\_Staff\_Lockup\_Schedules.[Start\_Date]**End Date Field** – RMC\_Staff\_Lockup\_Schedules.[End\_Date]**Default View** – Current & Future Schedules |
| E-Phone Schedules | **Section** – Staff Stuff**Default Field List** - Contact\_ID\_Table.[Display\_Name] , Contact\_ID\_Table.[Mobile\_Phone] , RMC\_Staff\_EPhone\_Schedules.[Start\_Date] , RMC\_Staff\_EPhone\_Schedules.[End\_Date] , RMC\_Staff\_EPhone\_Schedules.[Notes]**Selected Record Expression** - Contact\_ID\_Table.[Display\_Name] + ‘ ‘ + Contact\_ID\_Table.[Mobile\_Phone]**Start Date Field** – RMC\_Staff\_EPhone\_Schedules.[Start\_Date]**End Date Field** – RMC\_Staff\_EPhone\_Schedules.[End\_Date]**Default View** – Current & Future Schedules |

**Network/Computer Device Example**

An example of a church that tracks network device information. The information is used for network troubleshooting, to communicate static IP address usage between departments, to locate devices in church buildings, indicate if a MAC address has been added to a MAC access control list for wireless devices and if devices are active or in reserve for later use.

**Approach**

Create two tables to keep track of network and computer devices on the RMC network. One table will be a lookup table to the main device inventory table. The purpose of the look up table is to keep track of device types and the users who oversee the use of those devices. The main inventory table will track the device names and other useful device characteristics.

|  |  |
| --- | --- |
| **Table Name** | **Field Name/Data Types** |
| RMC\_Network\_Devices | RMC\_Network\_Device\_ID - PK, int, Identity, NOT NULLDevice\_Name - nvarchar(50), NOT NULLModel\_Number – nvarchar(50)Description – nvarchar(225)Room - int, NOT NULL, FK to Rooms tableIP\_Address - nvarchar(15)MAC\_Address – nvarchar(17)Device\_Type – int, NOT NULL, FK to Device Types tableSerial\_Number – nvarchar(50), Notes – nvarchar(225)Active – bitIn\_MAC\_Address\_Table - bitDomain\_ID – int, NOT NULL |
| RMC\_Network\_Device\_Types | RMC\_Network\_Device\_Type\_ID – PK, Identity, int, NOT NULLNetwork\_Device\_Type – nvarchar(50), NOT NULLDescription – nvarchar(225)Coordinator – int, NOT NULL, FK to Users tableDomain\_ID – int, NOT NULL |

**Page Creation**

Create pages titled **Network Devices** and **Network Device Types** with the following properties:

|  |  |
| --- | --- |
| **Page** | **Properties** |
| Network Devices | **Section** – Events and Facilities**Default Field List** – RMC\_Network\_Devices.Device\_Name, RMC\_Network\_Devices.Model\_Number, RMC\_Network\_Devices.Description, Device\_Type\_Table.Device\_Type, Room\_Table.Room\_Name, RMC\_Network\_Devices.IP\_Address, RMC\_Network\_Devices.Static\_IP, RMC\_Network\_Devices.MAC\_Address, RMC\_Network\_Devices.Serial\_Number**Selected Record Expression** - Device\_Name**Start Date Field** – N/A**End Date Field** – N/A**Default View** –  |
| Network Device Types | **Section** – Lookup Values**Default Field List** – RMC\_Network\_Device\_Types.Device\_Type, RMC\_Network\_Device\_Types.Description, Coordinator\_Table\_Contact\_ID\_Table.Display\_Name**Selected Record Expression** - Device\_Type**Start Date Field** – N/A**End Date Field** – N/A**Default View** –  |