**Custom Pages: Ministering with Meaningful Information**

**Matt Patterson – mattp@rmcalvary.org  
Rocky Mountain Calvary, Colorado Springs**

**Background**

Consult 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. New table naming conventions. Tables names should:
   1. Begin with a prefix to differentiate those tables from tables created by ThinkMinistry.
   2. Be plural (e.g., **Contacts** to store each contact, **Events** to store each event).
   3. Use underscores rather than spaces between words. Do not use any special characters (e.g., “/” or “-“).
2. New 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. New 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 (FK) that relates to the primary key of associated tables.
4. Plan and document the fields, 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**

**Page Creation**

A page is the object used to display the information in a new table.

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 Field List** – 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** – The **Start Date** field with the **datetime** data type is 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 data 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 the **Copy** verb to appear so 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).
      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**

Users must be given security rights to the page (including Administrators) in order for the page to be viewed.

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 data displays properly in every view that is applicable to the page (i.e., Grid, Calendar, Gantt and Images views).
3. Test any views created for the page.
4. 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.
5. 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.

**Transferring to Production**

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

When moving tables into production, there are three primary concerns: creating the tables, creating the proper relationships between tables, and inserting the data in the proper order if the **new** tables have a parent/child relationship. Listed below is one approach:

1. Create a script of the table(s) to transfer to production.
   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.
2. Run the create table statements on the production server (see **Table Creation Notes** below).
3. Follow the steps to create sections, pages, views, etc. as was done on the test server.
   1. Copy the information for each object in the System Setup folder on the test server and paste it to the objects on the production server.
4. Manually insert data on the test server pages into the new product pages (If a significant amount of data has been added during testing, the tables may need to be imported into the production SQL Server database).
5. 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.