How to Store Report Templates

Some applications require the storing of report templates in a database. This simplifies template support - all reports are stored in one place, and allows to differentiate the access rights to different templates. I noted only one shortcoming of this technique - slight complication of the code in the client application.

For example, we need to create Access database (mdb) named "reports" and a table for storing our report templates. I named this table "reports_table". Three tables are enough in our data table:
- Field name Type Description
1 Id counter Auto-increment, index
2 ReportName text Name of report
3 ReportFile BLOB Binary data with template

You can store templates in any other database similar to the one in our example.
To do this:
• open the designer;
• show custom form with a list of templates from our database instead of standard dialog when users open a file in the designer;
• load template in the designer after selecting it in the custom form;
• Save the template in a database where the user saves a file in the designer.
Despite the apparent complexity, the tasks above will be simple because FastReport.NET has special features.
Create a new project in Visual Studio first. Then you need to drag-n-drop "Report" object from toolbox to the empty form. Also drag the " EnvironmentSettings" object (properties of reporting engine).

You can setup many settings of the report and the designer in "environmentSettings1" object. You have to override the event handlers CustomOpenDialog, CustomOpenReport, CustomSaveDialog and CustomSaveReport.

Event handler CustomOpenDialog allows releasing any algorithm of the load report template in the designer - load template from the database field and load it in the designer or set e.Cancel = true; when loading fails.

The problem reduces to showing a dialogue with a list of files from which you want to select the report template from the database and load the template in the designer. Of course, if the user presses the Cancel button in this dialog, we set e.Cancel = true. The loading process is clear but saving code need some details.

The user can choose two ways to save a file - simply by clicking "Save", in this case report should be rewritten to the old place, or the second option - "Save as...", the user is expected to select path of file. Storage location of report templates cannot be changed in our case. You need to save template in the old place (database field) in both ways - capture the CustomSaveDialog event by empty function and save template in database in the CustomSaveReport event.

Now you need to add a new data source to the project - reports.mdb.Drop the button on form for launching of the reports designer.

Write in the Click event handler:

```
private void button1_Click(object sender, EventArgs e)
{
report1.Design();
}
```

Create an additional dialog form for template selection:

Assign the buttons OK and Cancel corresponding to DialogResult.
Do not forget to bind your data source to DataGrid - select the field name of the record and make the index field invisible.
You need to save the selected index when you click on the 'OK' button:

```
public int reportID;
...
private void OKBtn_Click(object sender, EventArgs e)
{
// save id of report for use in future
reportID = (int)dataGridView1.CurrentRow.Cells[0].Value;
}
```

We return to our first form. It is time to handle the events of opening and saving a report. We make loading the first thing to do:

```
// define variable for store of report ID
private int reportID;
// define array byte[] for store of template
private byte[] blob;

....
private void environmentSettings1_CustomOpenDialog(object sender,
FastReport.Design.OpenSaveDialogEventArgs e)
{
using (ReportListForm reportListForm = new ReportListForm())
{
// show dialog for report selection
if (reportListForm.ShowDialog() == DialogResult.OK)
{
// get report ID
reportID = reportListForm.reportID;
// load report in array from BLOB
blob =
(byte[])this.reports_tableTableAdapter.GetDataByID(reportID).Rows[0]["ReportFile"];
// read file name of report for designers title
e.FileName =
```

```
(string)this.reports_tableTableAdapter.GetDataByID(reportID).Rows[0]["ReportName"];
}
else
// cancel loading
e.Cancel = true;
}
}
```

Second handler CustomOpenReport for loading the template in designer should look like this:

```
private void environmentSettings1_CustomOpenReport(object sender,
FastReport.Design.OpenSaveReportEventArgs e)
{
using (MemoryStream stream = new MemoryStream())
{
// skip all garbage created by MS Access in begin of blob field - we seek the tag of XML
int start = 0;
for (int i = 0; i)
```