A success story:
Dynaform – User-managed document forms in Lotus Notes

Jan Gundtofte-Bruun
jan@g-b.dk
2639 7565

WHAT DID I DO?

I was tasked with creating a tool for managing access requests to innumerable applications world-wide. Because application owners in diverse parts of the company would want to ask very different questions on the access request form, the tool required potentially thousands of custom request forms (which are normally crafted by developers before deployment).

I wrote a solution that enables ordinary (unprivileged non-programmer) users to design and update their own forms using a simple and pretty UI. The “recipes” they create get turned into real working code and forms by a fully automated back-end running on a production system.

WHY WAS IT NECESSARY?

Normally, Lotus Notes programmers create a handful of more or less static forms for an application, so managing thousands was going to be a prohibitively work-intensive burden in regard to initial development as well as subsequent maintenance. To sidestep that obstacle, I invented, architected, and prototyped a radical and innovative method for on-demand form management.

This concept proved solid and was incorporated into the project.

HOW DID I DO IT?

Before starting to implement this concept, I carried out performance tests to see whether Lotus Notes could even handle thousands of forms, and it turned out to be no problem at all.

First, I designed a form with all kinds of entry fields, exported this as DXL (Domino XML), and chopped it into little pieces. Then, I designed a user-friendly interface through which application owners could compose their forms using whatever combination of fields they desired. Apart from the data-entry field itself, a “field” definition also included a label, some help text, and data validation rules.

The output is a “recipe” document, which is detected by a background task that parses the recipe and composes appropriate DXL for the corresponding form, and compiles it into a design element, ready for use. Recipes can be updated, in which case the old revision is kept and existing access requests that are “in the pipeline” are completed using the old revision. A recipe also includes parameters so that forms can be created in advance and be available only within select regions or divisions, as well as from, until, or between specific dates.

End users are presented with an interface to browse available applications based on the existing recipes and their parameters, and may create access request documents.

Documents created with such forms have a number of common fields (information about the requester, which application it's for, etc.) plus a varying number of custom fields. The custom fields follow an internal naming scheme that ensures views can be built relatively easily and the request tool does not exceed the number-of-fields limit.

The prototype turned out to be very successful, and was incorporated with only minor changes into a working product. The product included workflow and approval mechanisms and so on, but that is outside the scope of this story.

WHAT'S SO COOL ABOUT THAT?

I was not able to find any evidence online of previous attempts to create a Notes application with this type of user-driven self-modifying behaviour. To boot, I received overwhelmingly positive feedback on the form-building UI.

The points I feel most proud about are:

- It's a self-modifying application (on IBM's production servers, no less), that in itself is pretty cool!
- Pulling off dynamic compilation based on user-generated designs without any risk or skill on the part of the end users.
- Setting up a prototype and test it with a large-scale dataset with positive results.
- The reaction from end application owners has been overwhelmingly positive (even though they are, in a way, doing our development work for us).