

## Submission “Toni’s Bauernsuche”

The idea for “Toni’s Bauernsuche” (Toni’s Farmer-Search) resulted from a project we (*muevo*) did together with one of our partners (*moodley*). The clients’ company (*Toni’s Freilandeier*, <http://www.tonis.at>) sells free range eggs that come from farmers in the local area. With the re-launch of his website the client required a feature showing his customers the origin of each egg (every egg sold in Austria has a special control number printed on its shell, connecting it to a specific farmer, and indicating how the animals are kept).

The *Flash* microsite allows users to browse information about the farmers who supply *Toni’s* (and are therefore committed to strict quality standards). It shows photographs, detailed information such as the number of hens, contact addresses, and phone numbers.

Users can also directly search for specific farmers using the farmer’s name, ZIP code, or egg control number as search criteria.

## Technical Facts

The project consists of two major parts: a *Flash* frontend for visualization and a PHP backend for content management. The data is held in a *MS SQL* database provided and maintained by the client’s company.

The PHP backend handles database connectivity and functionality for data retrieval, including search and random search queries. All content is loaded dynamically at runtime, including the loading of images, which is served by a binary action. The backend is based on the *CakePHP* framework, which allows a rapid development of the necessary database handling, and a simple administration area to directly maintain content and images online.

Using *Wouter Verweirder’s* great *CakeSWXPHP* implementation, it was fairly straightforward to add SWX functionality to the backend.

The frontend is implemented in *Flash* and *ActionScript 2.0*. By encapsulating the business logic in several distinct classes by making use of best-practice software design patterns like facades and singleton objects, it was easy for both (non-programming) designers and more advanced software engineers to work together on this project. SWX integrated very well through its object oriented approach.

## How It Uses SWX

At the beginning of the project we had a close look at different techniques for implementing *Flash* to the backend communication. With most techniques the necessary “overhead” for sending and retrieving data to and from the backend was far too high. After discovering *SWXFormat.org* it was soon clear that this would be the way to go. SWX not only has a low overhead when it comes to necessary function calls for data preparation and retrieval, but through its binary file format the processing of the retrieved data is also a snap.

To get all the details on our implementation you can examine the attached file *Backend.as* in which we call and use all the functions for the SWX communication. For a brief overview, continue reading here.

## Initialization

To begin we initialize a SWX object (with the *SWX.as* class) and set several SWX parameters via a *config.xml* file, which is loaded at first. Also we set the appropriate *gateway*, *encoding* and *debug* properties.

For correct processing and error handling we implemented some *functions* for result, fault, timeout, and progress handling as well as connection checking.

## Search Functions

The main reason for using SWX was the ability to have an easy framework for the search queries we would need to visualize inside *Flash*. We used “baking” models, views, and controllers with *CakePHP*, and inserted our custom query logic for the *MS SQL* database within it. Based on this, and the *CakeSWXPHP* classes, we had readily accessible functions for SWX, explained below:

### findInit

The SWX-method *findInit* is called with the function *getInitIDs()*. This is a query for the start screen and returns 20 random farmers including their pictures and details (text fields of the database). For the user it appears to be an endless list of farmers.

### findName

The SWX-method *findName* is called with the function *getIDsByName(p\_name:String)*. This is a query that returns all farmers whose surnames commence with the query string. At this point the application only shows the returned results and users can find out about the farmers details by clicking on them.

### findZip

The SWX-method *findZip* is called with the function *getIDsByZip(p\_zip:String)*. This is a query that returns all farmers located in the region defined by the query string. Here the user also gets feedback through a list that displays the results. Again by clicking on a farmer the user gets further details.

### findEiNummer

The SWX-method *findEiNummer* is called with the function *getIDsByEi(p\_ei:String)*. This is a query that returns the farmer indicated by egg-control-number. In this case the result is a single farmer that is shown in the application with all his or her details.

## Where to Find What

The screenshot shows the website 'Toni's Bauernsuche' in a Mozilla Firefox browser window. The page title is 'Toni's Bauernsuche' with the tagline 'Die besten Eier unter der Sonne.' Below the title is a large image of a family and chickens. A search bar at the bottom has three input fields: 'STICHWORT / NAME', 'EI-KONTROLLNR.', and 'PLz', followed by a green 'LOS GEHTS' button. A pop-up window on the right shows details for a farmer named 'SCHIRNHOFER GERTRAUD', including her photo, contact information (8225 Pöllau, Winzendorf, T. 03335 2527), and a description of her farm. Blue arrows point from text labels to these elements: 'farmer's details' points to the pop-up; 'selected farmer' points to the family image; 'findName' points to the first search field; 'findEiNummer' points to the second search field; 'findZip' points to the third search field; and 'search button' points to the 'LOS GEHTS' button.

## Search Queries

Since the application is in German, here is some help if you want to test it for search queries that lead to a result.

### findName

There are about 300 farmers in the database so any letter should work here, try "N", "B" or "T".

### findEiNummer

The egg-control-number is the code printed on the egg's shell. Try "0-AT-3049353", "1-AT-3172619" or "1-AT-3265871".

### findZip

A search for the zip code; try "8720", "8542" or "8543".

## Online version

URL to a running copy is <http://tonis.muevo.at/>.

## Thanks

We thank our partners at *moodley brand identity* who did the graphics and the final *Flash* polishing, especially Natascha and Maex.

We also thank the developer team of *CakePHP* and Wouter Verweirder for his excellent *Cake - SWX PHP* integration.

And last but not least we of course thank Aral Balkan for bringing us the great *SWX format* 😊

## Contact

### Flash implementation, SWX and PHP backend

Ms. Elisabeth Bieber, Mr. Andreas Forstinger  
*muevo Information Management | IT Consulting*  
Kärntner Strasse 525  
8054 Seiersberg  
AUSTRIA  
mail: [office@muevo.at](mailto:office@muevo.at)  
web: [www.muevo.at](http://www.muevo.at)

### Design and Graphics

*moodley brand identity*  
Grieskai 52  
8020 Graz  
AUSTRIA  
mail: [harald.kantner@moodley.at](mailto:harald.kantner@moodley.at)  
web: [www.moodley.at](http://www.moodley.at)