Institutionen för datavetenskap
Department of Computer and Information Science

Final thesis

Feature Store

by

Oskar Lundvall

LIU-IDA/LITH-EX-G--11/002--SE

2011-03-23
Final Thesis

Feature Store

by

Oskar Lundvall

LIU-IDA/LITH-EX-G--11/002--SE

2011-03-23

Supervisor: Andy Stacy
Examiner: Anders Fröberg
Abstract
This thesis describes the development of an internal web based tool for Ericsson AB in Linkoping. The problem addressed in this work is finding and managing new sales opportunities. The tool consists of an extensive structure of websites written in PHP and JavaScript with the database connection handled through MySQL. The content is a register of products but also has the ability of automatic generation of statistics. The purpose of the tool is to make the work with Ericsson’s customers more efficient, the tool will make it easier for the sales person to find a sales opportunity. The work structure consisted of two phases, an Alpha-stage to determine if there was value in developing the product and after that was evaluated the project went on to a Beta-stage to set the foundations for the proper product.
Feature Store – thesis by Oskar Lundvall, 2011
Introduction

Background
Ericsson AB is a Swedish telecom company which also acts internationally. They have since the start in 1876 developed telephones but the focus has shifted towards the building of telephone stations and their functionality. During the period of 1990-2000 Ericsson became one of the world’s leading mobile phone manufacturers but this section of the company has since been transferred to a new company called Sony Ericsson in 2001. Today the business focus is development of software and more specifically the software ran on telephone servers.

Purpose
Ericsson today lacks a user friendly tool to give the sales personnel an overview of the customers they work with and their needs. Feature Store is supposed to be a register of products and functionality but also ease the work of finding sales opportunities. The requirements of the product were very loosely defined at the start of the project, this was to let the result adapt to the actual needs of the users.

Techniques
A brief description of the different techniques implemented in the project.

HTML
HTML stands for HyperText Markup Language and is code that describes the structure of webpages. It consists of so called tags which enclose different parts of a webpage; an example is the <b> tag which together with its end tag </b> would make text within it bold (Wikipedia, 2011).

PHP
PHP stands for Hypertext Preprocessor and is a scripting language used for creating dynamic webpages. PHP code is embedded in HTML and interpreted on a webserver to generate additional HTML (php.net, 2011; Wikipedia, 2011).

JavaScript
JavaScript is a scripting language that is ran on the client side for a webpage; its main use is creating enhanced and dynamic interfaces (Wikipedia, 2011).

MySQL
MySQL is a language for communicating with the database, with the help of queries you may add, remove or modify information stored (mysql.net, 2011; Wikipedia, 2011).
Ajax
According to Wikipedia Ajax stands for Asynchronous JavaScript & XML and is a group of interrelated web development techniques used on the client to create dynamic functionality (Wikipedia, 2011).

CakePHP
CakePHP is a PHP framework modeled after the concept of Ruby on Rails, this means it contains built in functionality to make development in PHP more stable and modular (www.cakephp.org, 2011).

JQuery
JQuery is a JavaScript framework; it contains functionality for easier and more efficient creation of JavaScript elements like for example dynamic forms (Jquery.org, 2011; Wikipedia, 2011).
Requirements

The requirements of the project are not easy to describe seeing how they have changed gradually for each iteration; a big part of working agile.

The initial goal was however to create a reference of the products sold by Ericsson and have the customer relations to these products represented as to gain insight in their future needs.

To elaborate on the Ericsson products they are structured as follows; a product is a large collection of related functionality which in turn consists of many releases, or updates if you will. Each release has basic functionality but also single features which can be bought separately. Every function of a release is however referred to as a feature and this is the origo of the entire system, most objects in the database have a relation to the features.

Once the project had lift off a number of additional requirements were added:

- A customer should have the country attribute since some features would only be available in certain areas.
- A feature should have a type, state, dependencies (to other features), conflicts (with other features).
- A feature should be able to belong to a package.
- There should be users with specific access to different customers and packages.
- A user should be able to have favorites to easier access relevant information within the system.
- There should be log entries automatically made for different objects in the system.

In appendix 4 we can see a graphical representation how these points were implemented in the database.

Upon the initial testing phase more requirements were found:

- For another scope commentary should be available for most objects.
- A feedback system should exist for testers.
- Dependencies and conflicts were deemed out of scope because of their very complicated nature.
- A customers relation to a feature turned out to be more complicated then initially estimated which resulted in the 4 split solution implemented.

Interpretation of the requirements

Based on the very loosely defined requirements we set our own goals for the project. Here is the functionality we decided to include in this scope.

- A webpage in PHP using MySQL for database connection and the framework CakePHP. Our reasoning were mainly being familiar with these techniques from before while at the same time they are some of the industry standards
- The possibility to add customers and have connections to features, products and releases. Other variables are partly described in the last paragraph and in appendix 4 but they include i.e. which country they operate in.
• Different relations between a customer and a feature specified in separate tables to be easier to work with
• An index of the features in the products MSC, HLR and MGW. The features would include the variables specified in the last paragraph
• Login required to access the page
• The possibility to add most of the object types as favorites
• Log entries for all actions related to a users favorite
• A relation for features called Needs this means how they provide value for a customer
• The possibility to assign needs and prioritize them for a customer
• Automatic detection of upgrade possibilities for a customer
• Automatic display of related packages for a customer based on owned features
• Automatic advice on sales opportunities for features based on interest, needs and related packages for a customer
Realization

Alpha-stage

Work
The stage was initiated by explanations and discussions regarding the mockup that had been created during the upbeat of the project; this consisted of an Iphone application that lacked a proper database connection. Based on this development of a similar web based solution was started. The alpha had very loose requirements and came to include features like:

Login
![Top login in the alpha-stage.](image)

Filtering of products
![Filtering in the alpha-stage.](image)

Furthermore the techniques of Ajax & JavaScript were introduced for dynamic searches.

Acceptance
The result of the work so far was presented to the decision makers who cleared the transition into beta-stage based on the possibilities they had seen in the Alpha.

Beta-stage

Introduction
At the stage of this phase there was a need for additional resources in the project, another programmer was brought in and the system was rebuilt from scratch with the new goal of setting the foundations of a proper product instead of showing the possibilities ahead. The decision to utilize a framework was now taken to make the system more modular, stable and easier to work with. Furthermore the temporary webserver was rebuilt to better fill the needs of performance that had came with the new stage and therefore additional users in the form of a testers.

Continuation
After setting the foundation it was now possible to set a time plan for the continued work and also to give the current developers continued confidence in the finalization of the beta.
Development & Input of data

The goal was now to make the system ready for the first testing-phase; some functionality was required to be able to assimilate the valuable feedback. For this purpose a number of spoof customers, products, features and connections in-between them was created. A proper webserver was also set up with the purpose of running a stable version of the system for the testers. For the beginning of testing a feedback system was set up, it consists of a form reachable from any page with the help of a button. The thought was to be able to give instant feedback whenever a tester got an idea.

For reviewing the feedback a separate page was created, it contains information about the suggestion but also shows the status, in the case below the suggestion was executed.

Fig. 3 – The global feedback button

Internal Unstructured Testing

The unstructured testing consisted of asking a number of colleagues at Ericsson to try out the system however they wanted without instruction. After this they were free to give any feedback they could think of and the development team would decide were to go from there.

Further development based on feedback & Planning of further testing

Feedback sessions were held with the chosen testers to discuss the input they had given, based on this decisions were taken for what to change. Additional functionality was created to prepare for the structured testing.

Internal Structured Testing

For this task a dedicated testing expert was assigned to hold sessions with some of the potential end users of the product, they were in turn then instructed to fill in their feedback by the same form that had been used previously. This week ended with a meeting in Skavsta to evaluate all the valuable ideas of the group.
Preparation for external testing
With all the constructive criticism received over the previous weeks modifications were now made to the product, many new features had been requested and big amounts of data were entered in the system to give the external testing a more real feel to it.

External testing
External testing is to be a major out of office test taking place in South Africa over the course of a week with actual sales personel. It was however out of scope for this thesis.
Feature Store
This section will describe the finalized system until end of scope for the thesis.

Login
The login page is the first page and to reach anywhere else in the system the user is required to log in.

![Login](image)

Start
The start page is where an accepted login takes you, at the first run you will not have any favorites and the following screen will be shown:
As can be seen we are directed to the help page since the user have no favorites, we also use a messaging system based on the action taken, in this case being logged in. From this page we can also see the navigation options in the top menu as well as the more general categories at the bottom.

**Customers**

The customers page lists the customers the current user has access to and we may also filter these based on some categories. Another option is to add a new customer if the one we are working with is not already in the system.

There is also as we can see the option to add customers as favorites already on this page.
A customer page

Upon selecting a specific customer we are taken to its respective page, this is somewhat the heart of the system as this is what could give ideas for new sales.

From here we can as we see navigate to several pages described later and we see log entries related to the current customer.
Customer Needs

The customer needs page is just an index of the different needs and we can navigate on to specific need pages.

The needs in turn belong to more general categories loosely defined as “goals” though not officially used in the project but represented in the database.
A need page from a customer perspective
A need page is not really useful unless browsed from a customer perspective, as we can see from the breadcrumbs I have gone in through the customer Brittish Com.

Fig. 8 – A need page

We can see the relations for the different features within the need and we can also easily navigate back to the customer or to other needs the customer have.
Packages
The packages page is the index of available packages.

Fig. 9 – Packages

A package page from a customer perspective
Just as for the needs a package page is truly interesting first when we enter through a customer

MSS SW Package - 2G/3G co-existence

Fig. 10 – Package page
Just as with the need page we now can see the individual relations to each feature in the package, it is easy to realize which value this can have once you know a customer has expressed interest in a specific package.
Products and Releases
The products page is the index of the products in the system

### Products

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Favorite</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M-MGW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HLR</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 11 – Products

Upon entering a product we will be able to navigate to a related release page, this is covered further down.
A product from a customer perspective
Once again the page becomes mostly relevant when browser from a customers perspective
When a customer is not up to date he will get an alert on his page like this

Fig. 12 – Update alerts

Selecting a product will then take us to the next picture
Which shows a listing of the releases within the product but also has the option to switch to roadmap view?

Fig. 13 – Product page

The roadmap is a visual representation of the release dates within the product; it is automatically generated based on the date information in the database.

Fig. 14 – The roadmap, in this case with colliding dates.
A release from a customer's perspective

On the release page there are two useful tabs, one is the information about features included but also what can be gained from a potential upgrade.

In this case the customer owns an older release of the product as indicated at the top, we then get a listing of what features we would gain from upgrading, this helps a salesperson motivate an affair to the customer.

The other tab is the statistics, this is for visual representation of the release and what value it brings, in this scope only needs are covered.
**Features**

The features page is yet another index page; here there are more filtering options as described before the features have many attributes and connections in the database.

![Features page screenshot](image)

Fig. 17 – The features page
A feature page
The feature page is a bit different in the sense that it does not provide specific functionality for a customer, it is just information about a feature, this is however something potentially very useful because some can find the current product information a bit inaccessible.

As we can see the descriptions of objects can easily include pictures since it is HTML, it could potentially even include embedded media like videos, this is however for another scope. There has also been basic testing regarding the dependencies as we can see in the bottom left.
Testing
The testing of the project was not mainly handled by the programmers, external resources set up the testing and this allowed us to only take in the criticism and build from it. Apart from this we had continuous feedback from our supervisor within the company which also included day to day changes in functionality and design.

Internal Unstructured Testing
During the first month of development we had created the built-in feedback functionality. This allowed multiple colleagues to use and evaluate the system during progress. This part being unstructured means this mainly focused on cosmetics in the form of changing titles, whitespacing and the general feel of the website.

Internal structured testing
The next part of testing was set up in a proper way, however we had no part in this process. Knowledgeable people within the company were booked in sessions with our testing person Johan Boestad, they then sat down to test and discuss the system after which telephone interviews were held from our part. This part focused on functionality and among other things decided to scrap the support for multiple languages which we had briefly worked on in the early stages of the project.

Workshop
The final part of testing for this scope was done through a workshop in Kista, Stockholm. A number of future users were set up to test the system together with the programmer group and discuss the direction and future of the project. This led to the decision to implement some major functionality such as the visibility aids of roadmaps and pie charts.

External Testing
As this scope of the project ends the external testing is being set up for spring 2011, this will be a more extensive study with the help of actual end users of the system.
Discussion
In this section I will discuss some of the key aspects of the project.

Working with a framework
I would briefly describe a framework as a number of ready made functions for a programmer to utilize; it is however more than that. Specially in the sense of adding structure to a project, on this I would like to describe the Model View Controller structure that CakePHP uses.

MVC
The Model View Controller is a way of structuring your project. The basic principles are:

1. The Model represents the application data
2. The View renders a presentation of model data
3. The Controller handles and routes requests made by the client

Or to put it in other terms:
1. The model describes the object
2. The view controls what you see on screen
3. The controller has the functionality

This might seem like a basic idea and to some extent it is, however it is extremely useful. When you want to make a change you know exactly where you need to go.

In appendix 1A we can see an example of a simple model in our project, all the information here is related to the structure of the object itself, to comment on some of the lines:
- We can use $order to set rules for how a database query should behave for this object
- $belongsTo creates a connection within the framework (not to be confused with a real database connection) that will automatically query related objects whenever we request an instance of this object.
- $validate can set rules for how data are allowed to be entered through forms

In appendix 1B we can see a view for the object, note that it is a view since there can of course be several of these. The one in question is the index page for the feedback-entry object and it lists the available entries with the help of a show-result view, very modular. In this file we can see some normal HTML-tags like <div> but worth noting is that a lot of the HTML is generated with the help of inbuilt cakePHP functions like $form->create(), more on this later.
Finally in appendix 1C we see parts of the feedback entry controller, in this case on the index function. It is not a very exiting one since all it does is retrieve information about the feedback categories available and stores them in a variable so that the view will be able to use them.

**Scaffolding**

In appendix 1C that we just discussed we can see that the Feedback entry controller has a variable called $scaffold. This is a built in cakePHP feature to generate basic functionality for a database application automatically. If the programmer activates scaffolding but has not written any views for an existing object cakePHP will create temporary pages to ease further development. Below is an example of the Customer Needs index with scaffolding activated and its views removed.

As we can see there is a lot of functionality without the programmer having had to do anything, every related object (through the model) can be accessed as well!

**Build in HTML-generators**

As previously mentioned cakePHP features functions that will generate structured HTML for the programmer, an example is $html->image() that will create a normal `<img>`-tag on the client side. The upside with this method is that the tags will be treated like objects on the server side, if I would like to change the width of an image dynamically based on some conditions that would for example be possible.
Elements
In appendix 1B we can see that a function called $this->element() is used. This creates a
cakePHP element which is practically code that gets embedded in the file before compilation,
based on conditions of course. The modularity this adds is obvious and we can see that already in
the index file 3 instances of a filter element is used.
In appendix 2 we can see that the filter element more or less consists of the creation of a form,
however the possibilities are almost endless and there are for example CSS added this way which
would have had to be repeated otherwise.

Working with servers
A thing that might seem obvious but I feel have been very valuable during the project is working
with two servers. Originally during the Alpha we ran only a local webserver due to the low
amount of users but once Beta was initiated we brought in a proper webserver to use as a stable
version, this meant that we have been able to make any changes to the original development
server before deploying this for the testers to see.
Upon setting up the second server I also learned about the excellent software Xaamp which
makes the process of installing a webserver so much simpler. It basically puts all the needed
software in one package.
Something which we did lack during the project which could have been useful was space on one
of Ericsson’s real servers; it would have been good to see how the system behaved in a proper
environment.

Backup
During my education I have utilized several different backup solutions such as SVN. This was
something that we did not have access to during this project and I definitely feel was missing.
Our way of doing backup was instead to manually put the current project in a winrar-archive and
copy it to a network disk (since these are backuped themselves within Ericsson). This was tedious
and also made it hard to revert to older versions.

Testing & Feedback
Working with feedback is something which I really enjoy, in our case this was often very direct
since the system we set up for it worked very well. I have really learned how much easier it is to
work with clear goals and directions; I feel we got these from the testers.
The one tricky thing with this is to determine which suggestions to implement; luckily all the
major changes went through our supervisor which would decide this for us.

Working agile
Since the start of the Beta-phase we have worked as a 2-man team, without a clear intent to do so
I would say we have been highly inspired by the method called Extreme Programming.
Extreme Programming

Extreme Programming or XP for short is a methodology for software development. It is one of the most common agile methods out there and the basic ideas is short iterations or development cycles which means the team is very adaptable to change. Working in pairs is a common element and also being ready for changes to the product. XP also have lower requirements on documentation than some of the more classic approaches like Waterfall Development (http://www.extremeprogramming.org/, 2011; Wikipedia, 2011).

Our method

We have to a large extent worked as a pair, this can in my opinion be very effective in some cases. For example you often spot potential bugs right away, also being able to always discuss the code you are currently working on normally improves it dynamically. Short iterations is something we have taken to hearth in the form of dividing tasks into very small parts and having constant customer contact, this means we have been able to change our direction pretty much on a daily basis. When it comes to documentation we have had non-existent requirements which have been quite a relief in my opinion, this has led to the little documentation that does exist being actually useful. We have produced a developers guide intended for potential new members to the project and we are also working on an integrated help-system on the page, this will hopefully remove the need of a user’s manual.

Managing the tasks for an iteration have been based on direct customer contact (our supervisor), we have then used a whiteboard to organize the tasks into manageable pieces. These have then been given a priority and on a daily basis we have chosen what to work on.
The speed of web pages

When working with modern web development there is one thing that can really separate a good developer from a worse one, this is the optimization for a system.

Firebug

Throughout the project our development browser of choice have been Firefox, the reason for this are the extensive addon library easily available for download. One such addon is called Firebug and is used to monitor various elements of a webpage, grouped with the Firebug feature Page speed we gain the ability to judge the optimization of a page automatically!
As we can see the webpage is then given a score based on different aspects of performance, in this case our development version have caching disabled and also the compression of JavaScript is turned off (http://getfirebug.com/, 2011).

**Optimizing SQL**

A big focus for us have been optimization of the SQL requests, for a web system of a larger scale it is essential to not do unnecessary requests since this can overload the server and make it less responsive. Because of this we have worked to reduce the information flow to and from the database on every page of the system, an example of this is given in appendix 3A. As we can see there is an extensive join-query being created with the help of looping through different objects, the narrower we can make a query like this the more effective it will get. The result can be seen in 3B, all of that information through just one query!

**Compression**

Compression for web pages means that we try to reduce the information sent from the server to the client, this will obviously make browsing faster. There are 2 main ways to do this, compressing code and compressing images.

When it comes to the code there are many resources available to help with this, we have chosen to work with GZip the idea is to first remove anything in the code that is unnecessary, for example whitespace or in some cases additional brackets added only for better overview of the code. An example of the difference between compressed and uncompressed code can be seen in appendix 5A and 5B. The second part is to actually compress the information before sending, this is hard to give a visual representation of but works in much the same way as image compression and is of course always loss-less.

For the images the compression is related to how they have been handled by the image program they have been created with, the goal is generally loss-less compression whenever it is available.
This means reducing the size of images without losing any information. One example of doing this could be to describe several pixels with the same color at once, in plaintext this could be explained as “Fill a 10x10 cm square with black paint” instead of saying “Fill the first 1x1 cm with black paint, then fill the next one…” and so on.

Conclusion

By the end of the thesis Feature Store gave a clear picture of the usefulness of the tool and the project was prolonged into a probable development of a live version, it contains the functionality it was set out to with more having been added during the course of the project. The original purpose have clearly been fulfilled with the tool already having some use within the company even though it is not in a finalized version.
Tools
A brief description of the tools used in the project.

Xaamp
XAMPP is an Apache distribution containing MySQL, PHP and Perl. We used it because it is easy to configure and run (Wikipedia, 2011).

Notepad++
This is the text editor that has been used in the project; it is very lightweight but has a lot of functionality and the possibility to get add-ons to satisfy your specific needs (notepad-plus-plus.org, 2011).

WinRAR
WinRAR is a compression software which we used for taking backups. (rarlab.com, 2011)

Photoshop
Photoshop is used to create and manipulate images which there were plenty in our system. (Wikipedia, 2011).

Incscape
Incscape is a free-ware software for vector graphics and we used it for some of our small resolution icons (inscape.org, 2011).

Opera, Chrome, Firefox, Internet Explorer & Safari
These are the biggest 5 web browsers in the world and they were used to control the compability of the system (see references section).
References

Extreme Programming
http://www.extremeprogramming.org/ (22/02/11)
http://en.wikipedia.org/wiki/Extreme_programming (22/02/11)

Xaamp – A webserver compilation
http://www.apachefriends.org/en/xampp.html (22/02/11)

PHP - Hypertext Preprocessor
http://www.php.net/ (22/02/11)
http://en.wikipedia.org/wiki/Php (22/02/11)

CakePHP - A PHP-framework
http://cakephp.org/ (22/02/11)

MySQL - Structured Query Language
http://www.mysql.com/ (22/02/11)
http://en.wikipedia.org/wiki/Mysql (22/02/11)

HTML - HyperText Markup Language
http://en.wikipedia.org/wiki/HTML (22/02/11)

JavaScript - Scripting Language
http://en.wikipedia.org/wiki/Javascript (22/02/11)

Jquery - JavaScript library
http://jquery.com/ (22/02/11)
http://en.wikipedia.org/wiki/Jquery (22/02/11)

Ajax – Asynchronous JavaScript & XML
http://en.wikipedia.org/wiki/Ajax_%28programming%29 (22/02/11)

Notepad++ - Text editor
http://notepad-plus-plus.org/ (22/02/11)

WinRAR - Compression software
http://www.rarlab.com/ (22/02/11)

Photoshop - Image program
http://en.wikipedia.org/wiki/Photoshop (22/02/11)

Inkscape - Vector Graphics program
http://inkscape.org/ (22/02/11)

Firebug – Plugin for Mozilla
http://getfirebug.com/ (22/02/11)

Opera, Chrome, Firefox, Internet Explorer & Safari - Browsers
http://www.opera.com/ (22/02/11)
http://www.google.com/chrome (22/02/11)
http://www.mozilla-europe.org/en/ (22/02/11)
http://www.apple.com/safari/ (22/02/11)
Appendixes

1A

```php
<?php

class FeedbackEntry extends AppModel
{
    var $name = 'FeedbackEntry';

    var $order = 'feedbackEntry.created ASC';

    var $belongsTo = array(
        'User',
        'FeedbackEntriesCategory' => array(
            'counterCache' => true
        )
    );

    var $validate = array(
        'content' => 'notEmpty'
    );
}
```
1B

```php
<?php // get(url( array_merge($appdata['params']�이['named'], array('a'=>"b")) )); ?>

<div class="filter-wraper">
  <div class="filters">
    <form method="post">
      null,
      array(  
        'action' => 'show_result',
        'class' => 'ajax-form',
        'id' => 'FeedbackEntry-filter-form'  
      )
    );
  </input type="hidden" class="output-target" value="#FeedbackEntry-content" />

  <?php
    echo $form->hidden('page');
    echo $form->hidden('sort');
    echo $form->hidden('direction');
  ?>

  <div class="filter-row first">
    <?php echo $this->element(  
      'filter',
      array(  
        'field_name' => 'FeedbackEntry.content',
        'params' => array(  
          'size' => 24,
          'label' => __('Content'),
          'class' => 'filter',
          'type' => 'text'  
        )
      )
    );
    echo $this->element(  
      'filter',
      array(  
        'field_name' => 'FeedbackEntriesCategory.id',
        'params' => array(  
          'options' => $viewdata['feedback_entribe_categories'],
          'label' => __('Category'),
          'empty' => __('(Any)')  
        )
      )
    );
    echo $this->element(  
      'filter',
      array(  
        'field_name' => 'FeedbackEntry.resolved',
        'params' => array(  
          'options' => array(  
            '0' => __('Unresolved'),
            '1' => __('Solved')  
          )
        )
      )
    );
    echo $this->element(  
      'filter',
      array(  
        'field_name' => 'FeedbackEntry.content',
        'params' => array(  
          'options' => array(  
            '0' => __('(Any)')  
          )
        )
      )
    );
  </div>

  <div class="classres">nbsp;</div>
</div>

</div>
</div>

<?php echo $form->end(); ?>
</div>

<div class="ajax-content" id="#FeedbackEntry-content"></div>
```
```php
class FeedbackEntriesController extends AppController
{
    var $name = 'feedbackEntries';
    var $scaffold;
    function index()
    {
        $this->viewData['feedback_entries_categories'] = $this->FeedbackEntry->FeedbackEntriesCategory->find('list');
    }
}
```

```php
$wrapper_class = 'filter';

if (isset($params['class']) && strpos($params['class'], 'first') !== false)
{
    $wrapper_class .= ' ' . $params['class'];
    unset($params['class']);
}

$params = array_merge(
    array(
        'div' => false,
        'class' => 'input'
    ),
    $params
);

<?php echo $wrapper_class; ?>

<?php echo $form->input($field_name, $params); ?>
```

```php
// Conditions
$conditions = get_conditions_from_data($this->data, $search_options);

// Joins
if ( isset($conditions['AND']['Log.object_type']) ) {
    $object_types = array_unique($conditions['AND']['Log.object_type']);
} else {
    $object_types = array('Feature', 'Package', 'Customer', 'Product');
}

$joins = array();

foreach ( $object_types as $object_type ) {
    $joins[] = array(
        'table' => Inflector::tableize($object_type),
        'alias' => $object_type,
        'type' => 'LEFT',
        'conditions' => array(
            'AND' => array(
                'Log.object_id = '. $object_type . '.id',
                'Log.object_type' => strtolower($object_type)
            )
        )
    );
}

// Favorite
if ( isset($conditions['AND']['Favorite.user_id']) ) {
    $joins[] = array(
        'table' => 'Favorites',
        'alias' => 'Favorites',
        'conditions' => array(
            'AND' => array(
                'Log.object_id = Favorite.object_id',
                'Log.object_type = Favorite.object_type'
            )
        )
    );
}
```
3B

SELECT 'Log'.'object_id', 'Log'.'created', 'Log'.'content', 'Log'.'object_type' AS 'object_type', CONCAT_WS('', Feature.name, Package.name, Customer.name, Product.name) AS object_name FROM 'logs' AS 'Log' LEFT JOIN features AS 'Feature' ON (('Log'.'object_id' = 'Feature'.'id') AND ('Log'.'object_type' = 'feature')) LEFT JOIN packages AS 'Package' ON ((('Log'.'object_id' = 'Package'.'id') AND ('Log'.'object_type' = 'package')) LEFT JOIN customers AS 'Customer' ON (((('Log'.'object_id' = 'Customer'.'id') AND ('Log'.'object_type' = 'customer')) LEFT JOIN products AS 'Product' ON ((('Log'.'object_id' = 'Product'.'id') AND ('Log'.'object_type' = 'product')))) LEFT JOIN favorites AS 'Favorite' ON (((('Log'.'object_id' = 'Favorite'.'object_id') AND ('Log'.'object_type' = 'Favorites'.'object_type')))) WHERE 'Favorite'.'user_id' = '1' ORDER BY 'Log'.'created' DESC LIMIT 20
```javascript
// Global vars
var filterTimeout = undefined;
var filterTimeoutDelay = 300;

// Functions
function unfirst (str) {
  str = '';
  var f = str.charAt(0).toUpperCase();
  return f + str.substr(1);
}

// custom jQuery methods/functions
(function($) {

  // fn.setloading = function(settings) {
    return this.each(function() {
      var obj = $(this);
      var style = 'style="width: ' + obj.width() + 'px; height: ' + obj.height() + '

      obj.before('<div class="loading"><div class="loading-loading" style="" ' +
                   'x=""></div><div class="loading-spinner" style=""></div></div>');
    });
  }

  // fn.removeLoading = function() {
    return this.each(function() {
      $(this).siblings('div.loading').first().remove();
    });
  };

  // function orientationChange() {
    var orientation = window.innerWidth < window.innerHeight ? 'profile' : 'landscape';
    $('@content').attr('class', orientation)
```

---

Feature Store – thesis by Oskar Lundvall, 2011
5B

```javascript
var filterTimeout=undefined; var filterTimeoutDelay=300; function ucfirst(str){str+='';var t=str
(function(){$.fn.ajax=function(function(){return this.each(function(){var obj=$(this);var
stylenew='dialog-overflow');).hide();stylenew='#dialog-content');html(''));
});document.ready(function(){window.ajaxForm.prototype.live('submit',function(e){e.preventDefault();
var output_target_value=output_target_value.attr('value');var output_target=output_target_value
var id_dialog=(obj).parent();attr('id')=='dialog-content';if(has_target){output_target Serialized
$.ajax({type:'POST',url:obj.attr('action'),data:obj.serialize(),success:function(data)
if(id_dialog&&data.data.indexOf('CLOSE_DIALOG')===-1)
{$('#feedback-message').fadeIn('fast',function(){(this).fadeOut(5000,1,function(){(this).slider
if(has_target){output_target.html(data).removeLoading();}
else{}})});}
else{}})})
);$('#filter input[type=text]').live('keyup',function(){var obj=$(this);var form
formTimeout=setTimeout(function(){formTimeout=undefined;form.trigger('submit');});filterTim
var part=1;var parts=1;id:if(part.indexOf('')!=-1)
{var pair=part.split(' ');var input=#('model'+ucfirst(pair[0])).attr('value',pair[1]);
function()$('a.toggle').live('click',function(e){e.preventDefault();
$('[target]').toggleClass('hidden');});$('div.hoverable').hover(function(){$(this).toggleClass('hover
var is_toggled=obj.hasClass('toggled');if(obj.hasClass('radio-button'))
var group=group.closest('.li.radio-group');if(is_toggled)group.addClass('radio-group-selected')
else{group.removeClass('radio-group-selected');
}
group.find('div.toggled').not(obj).removeClass('toggled').children('input').removeAttr('checked
var input=obj.children('input');if(is_toggled)input.attr('checked','checked');
else{input.removeAttr('checked');});
});$('select.select-group-parent').bind('change',function()
var html=($('#drop').html());var drop=(div.drop).find('.drop').length>0
{drop.each(function(){var obj=$('[id]=this');var children='ul.drop-list');css('width',obj.width()+
p=('a.form-submit,div.form-submitter').live('click',function(e){e.preventDefault();
this);c [FormWrapperInner.removeLoading].html('data'))]];});$('a.tab-link').live('click',function(e
var prioritized_list=('#sortable-container').find('ul.prioritized-list');prioritized_list.
#('a.omniscient').click(function(e){e.preventDefault();var obj=#(this);var item=obj.closest(''
var neighbor=item.prev();item.detach();neighbor.before(item);
else if(obj.hasClass('movedown'))
{var neighbor=item.next();item.detach();neighbor.after(item);
else if(obj.hasClass('add'))
item.detach();prioritized_list.append(item);
else if(obj.hasClass('delete'))
item.detach();containment.find('ul.unprioritized-list').prepend(item);
updatePrioritizeButtons();item.fadeTo('fast',0.5, function(){item.fadeTo('fast',1);});});$('#ul.
tinyinput.tiny').submit();
});
});
});

```

Feature Store – thesis by Oskar Lundvall, 2011 41
Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extra-ordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnt som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/