Rapid WebGUI development for a CMS detector DB (R&D)

Lukas Thiemeier
Matthias Bergholz
Wolfgang Friebel
HEPiX Meeting
Ithaca, Nov 5, 2010

HELMHOLTZ ASSOCIATION

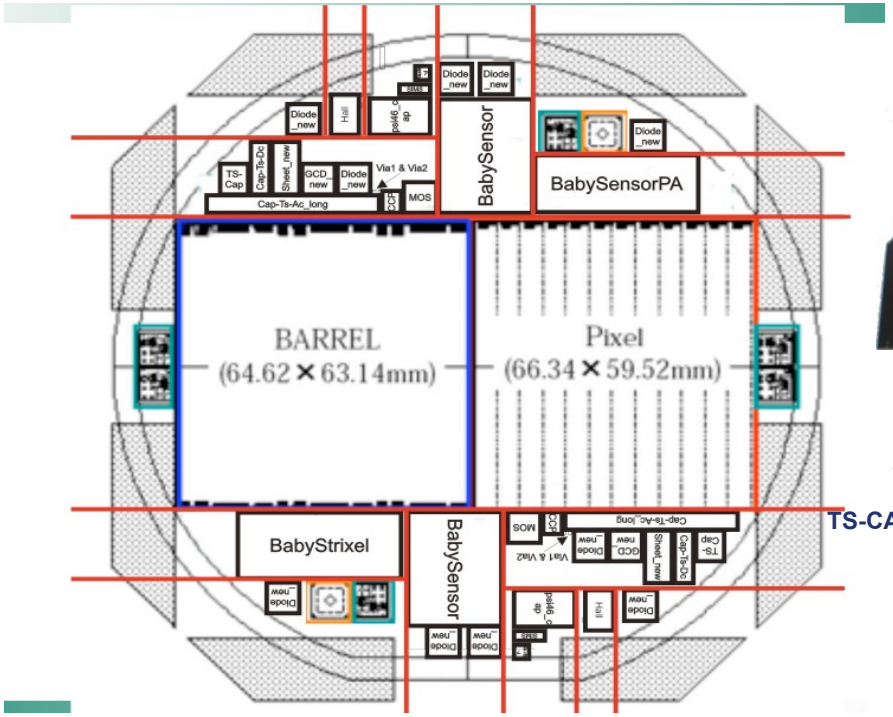
DESY

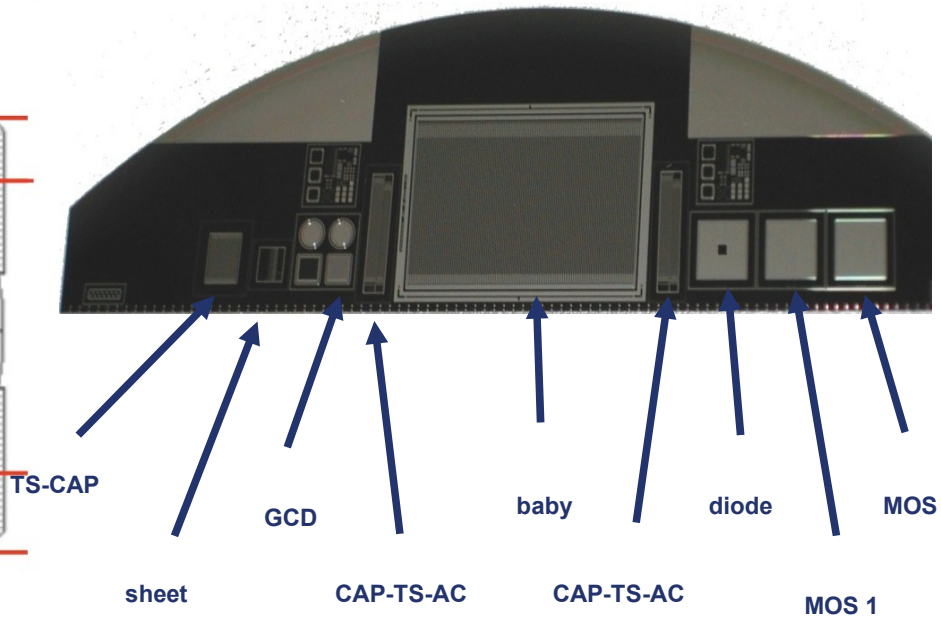
Outline of the talk

- Description of the Problem and Requirements
- Design Principles and Choices
- Catalyst and InstantCRUD
- The Database
- Developing for the Web GUI
- Summary

Test structures to measure

- Schematic layout of a test structure and a real wafer





Description of the Task

- Development of new Silicon detectors for LHC underway
  - Upgrade projects for CMS and ATLAS with similar technologies
- New materials and detector layouts are investigated
  - Silicon wafers with test structures produced
- Measurements of electrical properties of all test structures
  - Huge task, coordinated effort of many sites
  - Automated measurements using probe stations
- Database for storage of measurements essential
  - Allows retrieval of measurements from remote sites
  - Allows systematic studies of detector parameters
  - Central database for CMS silicon detector measurements at Lyon
  - Local database to organize and store measurements was regarded useful

Requirements for the GUI and the DB

- Requirements for a measurements DB formulated in 2009
- First measurements were expected early in 2010
- Working DB prototype with GUI planned for I/2010
  - Ambitious schedule especially for GUI development
  - No chance to meet schedule with traditional GUI programming (1FTE!)
- Accessibility of the DB
  - Without knowledge of SQL, OS and location independent, few or no prerequisites
- Be prepared for design changes
  - Due to the nature of an R&D project flexibility is important
  - Changes should affect as few components as possible
  - Changes should be possible by less involved maintainers of the DB

Design principles

- Rapid prototyping essential due to time constraints
  - Working solutions can be discussed, fast iterations, proof of concepts
  - Frequent database schema changes are feasible
- Layered concept to hide implementation details from user
  - Allows for design changes without user visible effects
  - Access to the DB possible but discouraged
- Web application instead of GUI
  - No need to install software for using the DB
  - Web access almost everywhere possible (no restrictions)
- Flexibility in the selection of a database
  - Use SQL without advanced or DB specific constructs if possible
  - Migration to a different DB should be easily possible

Web application frameworks

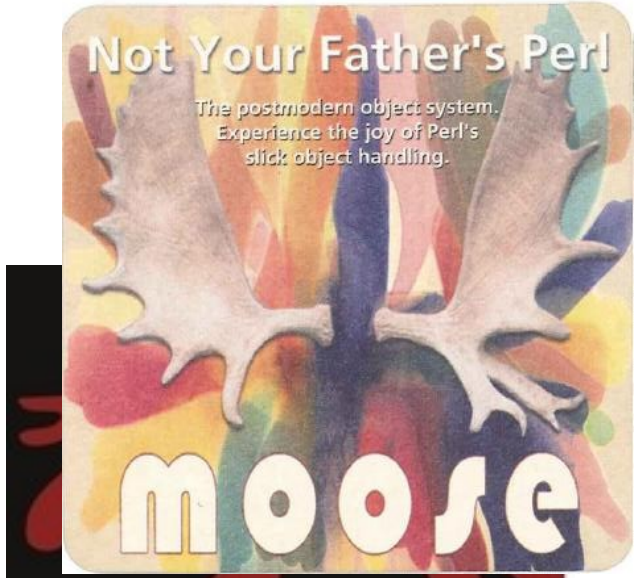
- Several solutions for rapid prototyping in web application frameworks
  - For many languages available, most of them inspired by Ruby on Rails (2004)
- Popular frameworks include
  - Zend Fraemwork for PHP
  - Django for Python
  - Merb for Ruby (or Rails)
  - ASP.NET MVC for .NET
  - Seaside for Smalltalk
  - Catalyst for Perl
  - Spring MVC for Java
- Implementations are using a Model-View-Controller (MVC) design
- Not all frameworks equally simple and powerful
  - Huge number of frameworks listed in compilations, need to look closer

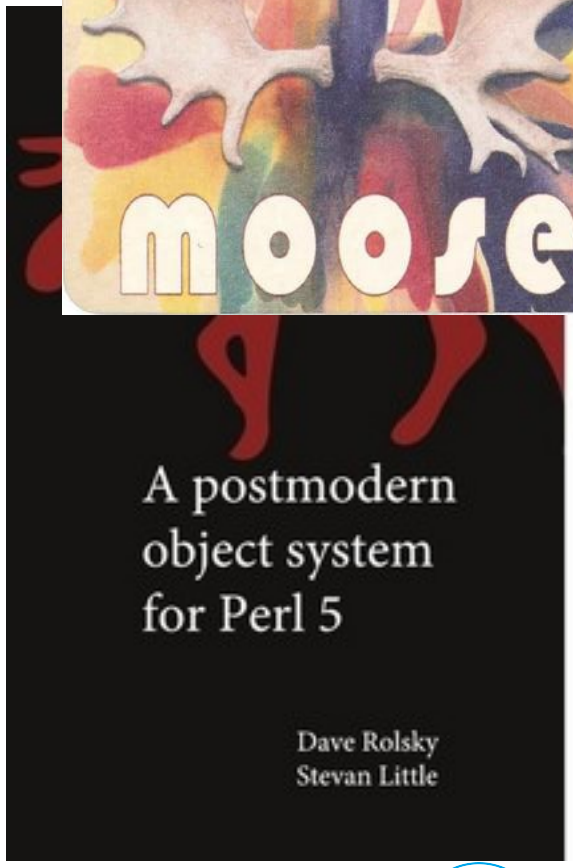
Design choices

- Catalyst (perl) has been chosen
  - Huge user base, vast number of perl modules on CPAN
  - Familiarity with the language
  - Many design principles can be easily fulfilled (eg. DB choice)
  - Highly automated generation of web user interfaces
- Django (python) also very popular
  - Has imho more constraints concerning choice of components
- Starting with mysql as DB backend
  - Also tried: migration to sqlite3 as proof of concept
- Using the Template Toolkit 2 for web pages layout
  - e.g. used in mailing list management software “sympa”

Important Catalyst building blocks

- Moose
  - New way of creating classes, objects, accessors in perl
  - “a postmodern object system”
  - “not your father's perl”
  - OO concept that is built into perl6
- DBIx::Class
  - A powerful object – relational mapper
  - “build an object model from your database automatically”
- Interface to a templating system
  - To specify the HTML pages content in a compact notation
  - Default templating system “Template Toolkit” chosen

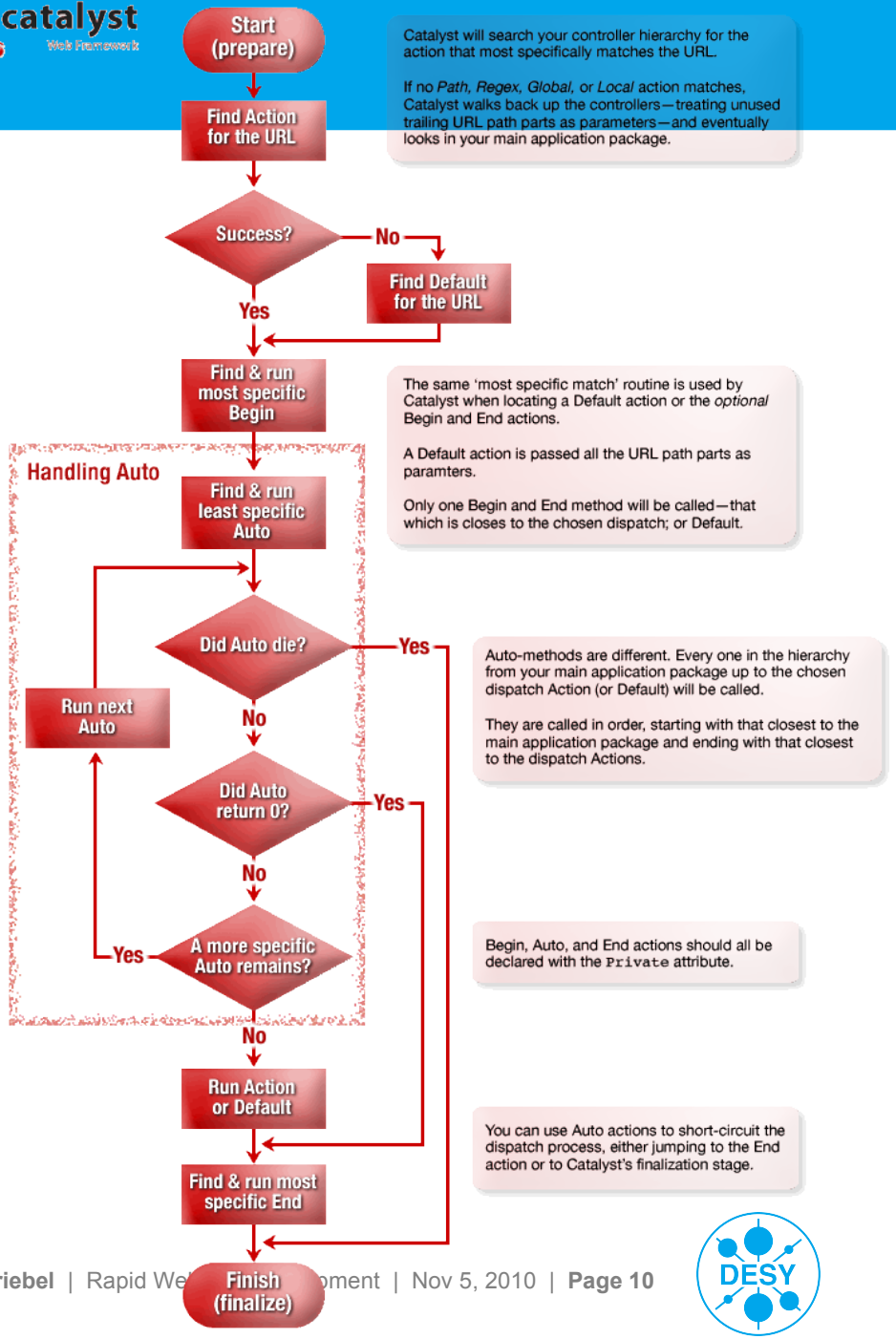




DESY

Catalyst dispatching system

- Very powerful system to organize the flow of pages driven by HTML requests (using clean and simple URLs)
- Methods allow to alter the processing sequence (e.g. go, forward, visit)
- Sensible defaults keep the amount of code to implement the program logic small



Even more rapidity: InstantCRUD

- Automatic creation of classes that do represent the DB schema

```
package CecDB::Schema::Result::Role;
use base 'DBIx::Class::Core';
__PACKAGE__->load_components("InflateColumn::DateTime", "TimeStamp", "EncodedColumn");
__PACKAGE__->table("role");
__PACKAGE__->add_columns(
  "name",
  {
    data_type => "varchar",
    default_value => \"NULL",
    is_nullable => 1,
    size => 15,
  },
  "id",
  { data_type => "integer", is_auto_increment => 1, is_nullable => 1 },
);
__PACKAGE__->set_primary_key("id");
__PACKAGE__->has_many(
  "person_roles",
  "CecDB::Schema::Result::PersonRole",
  { "foreign.role_id" => "self.id" },
  { cascade_copy => 0, cascade_delete => 1 },
);
# Created by DBIx::Class::Schema::Loader
use overload '""' => sub {$_[0]->name}, fallback => 1;
__PACKAGE__->many_to_many('people', 'person_roles' => 'person');
1;
                                                        23,1          All
```

one column

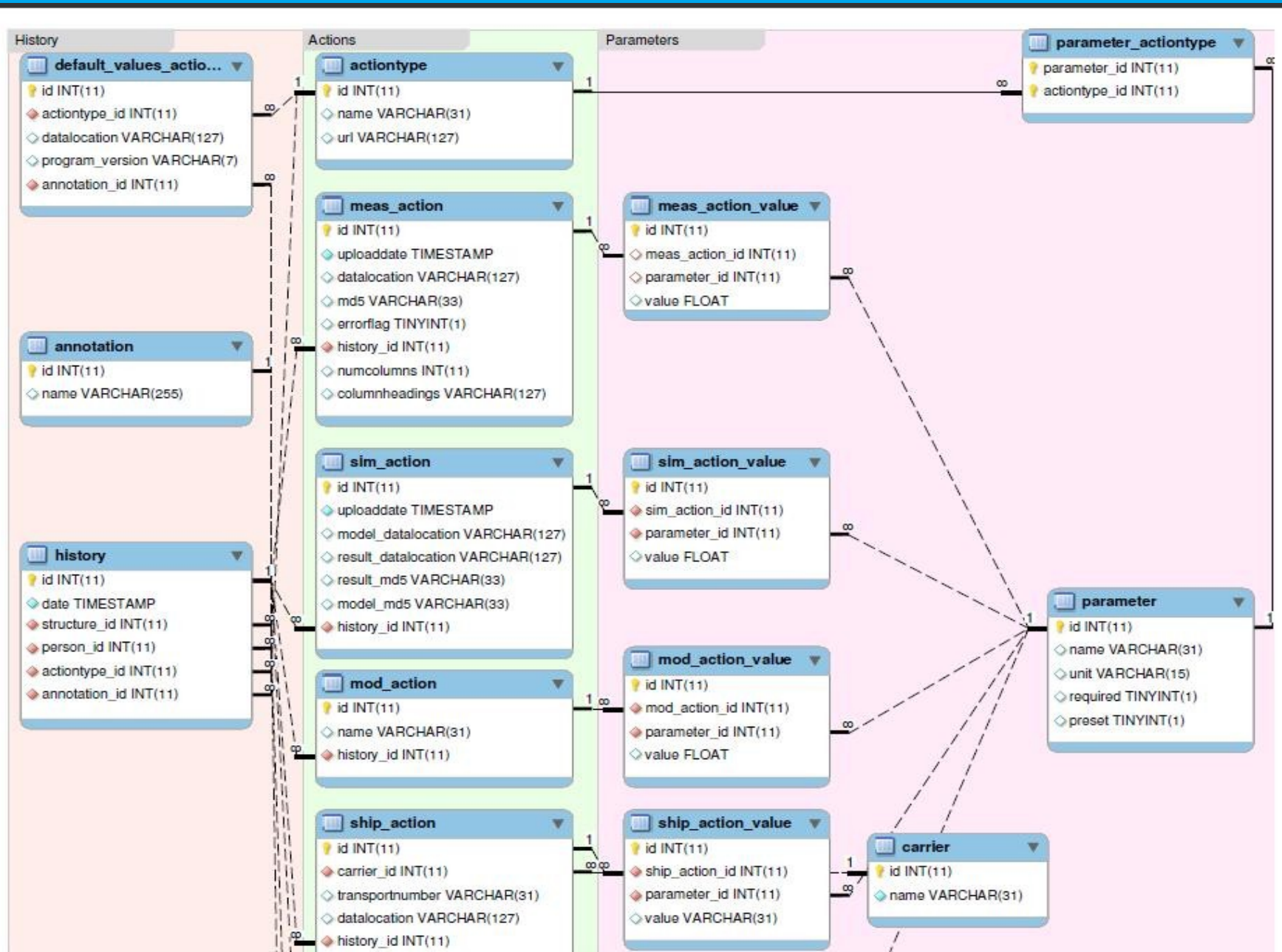
relation between tables

DESY

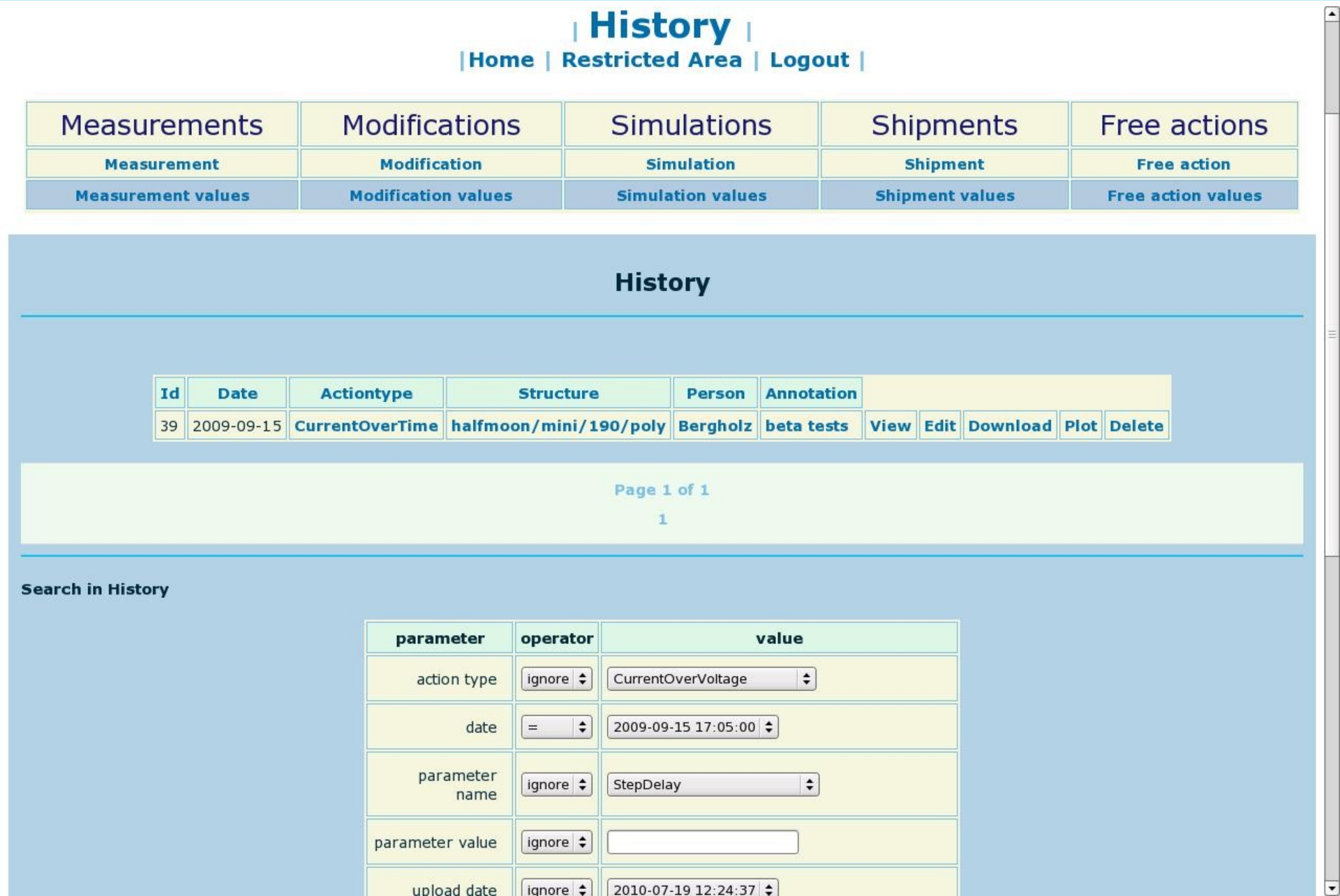
InstantCRUD

- Creates perl classes for a complete application
  - Takes the schema of an existing database to generate code
  - All perl classes for the Model, View, Controller
  - The SQL actions create, read, update and delete (CRUD) are implemented
  - Sophisticated treatment of foreign keys
- Creates all templates to render HTML pages
- Generates scripts to start a built in web server or to start the application as a FCGI program
- Often needed but missing: SQL action to search something
  - Got implemented as enhancement to InstantCRUD by Lukas Thiemeier
  - Automatically adds search forms that reflect the tables layouts
  - Easily customizable to generate complex search forms

The DB schema (partial view)



The Web GUI


History
Home | Restricted Area | Logout
Measurements
Modifications
Simulations
Shipments
Free actions
Measurement
Modification
Simulation
Shipment
Free action
Measurement values
Modification values
Simulation values
Shipment values
Free action values
History
Id
Date
Actiontype
Structure
Person
Annotation
39
2009-09-15
CurrentOverTime
halfmoon/mini/190/poly
Bergholz
beta tests
View
Edit
Download
Plot
Delete
Page 1 of 1
1
Search in History
parameter
operator
value
action type
ignore
CurrentOverVoltage
date
=
2009-09-15 17:05:00
parameter name
ignore
StepDelay
parameter value
ignore
upload date
ignore
2010-07-19 12:24:37

Enhancing the generated GUI

- Avoid adding program logic, if possible
  - Try to enhance the data model, not the controller
  - Less bugs, less maintenance problems
- Can filter or transform data from user input (forms, files, ...)
  - Done by applying predefined procedures to the data before they enter the DB
  - Store only encrypted passwords in the DB
  - Enforce password rules
  - Validate user data (e.g. correct date format, voltage or temperature range, ...)
- New pseudo columns can be created that do not exist in the DB
  - Columns do become methods that can be called from the template files
  - Methods can be generated on the fly (e.g. using “has_many” or “many_to_many”)

Adding password encryption and checks

- A few lines are sufficient to alter the password processing

```
package CecDB::Schema::Result::Person;
...
__PACKAGE__->table("person");
__PACKAGE__->add_columns(
  "password",
  {
    data_type => "varchar",
  },
);
# overwrite generated definition
__PACKAGE__->add_columns(
  "password",
  {
    data_type => "varchar",
    encode_column => 1,
    encode_class => 'Digest',
    encode_args => {salt_length => 10},
    encode_check_method => 'check_password',
  },
);
1;
~
~
~
~
~
                                                          10,32         All
```

Enhanced flexibility in the WebGUI by using roles

- We do assign roles to persons
  - Can be used in the controller to restrict access rights
  - Can be used in templates to generate different views to the data
- Role information can be combined with other data
  - Location dependent access or views e.g. by using the ENV variable REMOTE_ADDR

The command line

- Methods to manipulate the DB are available in scripts
  - Leads to very compact programs that are easy to understand
  - All accessors and relations defined within Catalyst can be used
  - Results of select statements are available in “Resultset”s
  - The Catalyst configuration data (DB name, user, password etc.) are accessible
  - Simplified script without sanity checks

```
#!/usr/bin/perl

use CecDB::Schema;
use Config::JFDI;
my $abspath = '/opt/cecdb/'; # path to the application
# retrieve connect information from config file of application cecdb
my $config = Config::JFDI->new(name => 'cecdb', path => $abspath)->get;
# connect to the DB, use retrieved connect info
my $schema = CecDB::Schema->connect($config->{'Model::DB'}->{'connect_info'});
# email address provided
my @users = $schema->resultset('Person')->find({email => $ARGV[0]});
# accountname given
@users = $schema->resultset('Person')->find({accountname => $ARGV[0]}) if ! @users;

# do the password change, assume we found only one user
my $user = $users[0];
my $line = 'secret!';
print "updating password for ", $user->accountname, "\n";
# create password according to the specs given in the user_model
$user->password($line);
$user->update;
exit;
                                                          5,54          All
```

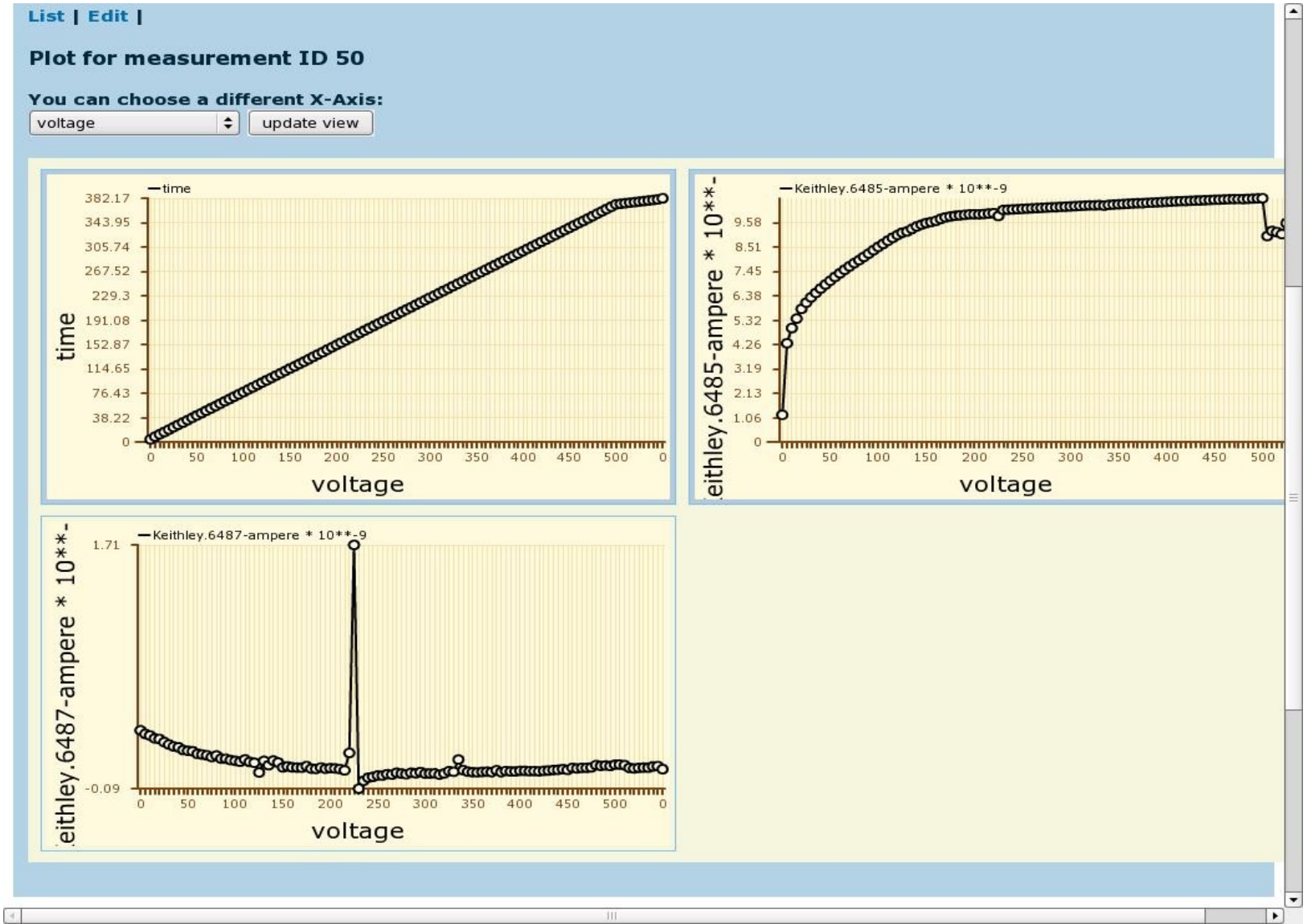
select statement

update statement

DESY

Integration with other packages

- Combining Catalyst with AJAX (e.g.jQuery) or Flash is straightforward



Summary

- Development of WebGUI's with minimal effort possible
- Modularity helps to delegate work
- Choice of the framework influenced by existing skills
- In Catalyst almost every single aspect of the GUI can be influenced
- Smooth integration with own scripts possible

DESY

References

- Comparison of web application frameworks

http://en.wikipedia.org/wiki/Comparison_of_web_application_frameworks

http://en.wikipedia.org/wiki/Model-ViewController

- Web GUI in action (password protected)

https://www-zeuthen.desy.de/cecdb