think php by MAYFLOWER

XTECH CONFERENCE

Enterprise app. development with Ajax & XUL

XTECH 2006| 14.05.2006 | Sebastian Schürmann

What youre gonna hear now !

- Why Sixt adtopted XUL
- Why we use AJAX (-X +J)
- What dev. Tools you can use for it
- Which components the framework consists of
- What kind of widgets we use and how they are constructed
- What Application(s) have we build
- Team Structure at Sixt
- Lessons we learned on XUL

What is Mayflower/ Thinkphp

- PHP service provider
- over 30 employees
- distributed on 2 sites: Munich and Würzburg
- 4 associates: Johann-Peter Hartmann, Albrecht Günther, Björn Schotte, Gregor Streng
- deep-rooted with the PHP-Community
- developer driven company

- PHP projects for various Enterprises as Sixt, Hypovereinsbank, Vaillant
- LAMP-Stack experts

Presenter

- developer since 1999
- Mayflower team member since 2005
- jack of all trades: Development to QA
- working in the XUL Project for over a Year
- part of various opensource projects (Mainly as QA)
  - Postnuke (cms)
  - PnCommerce (ecommerce)

- Email me: schuermann@mayflower.de

What is XUL ?

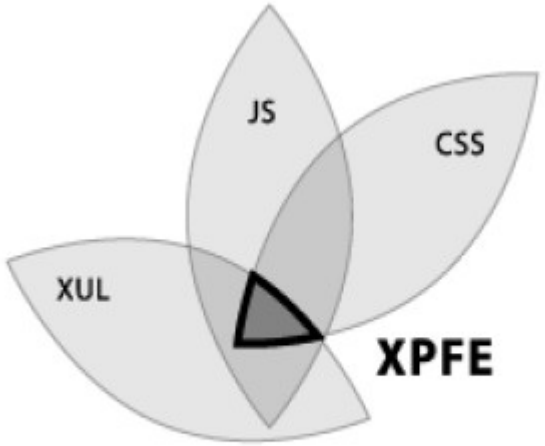
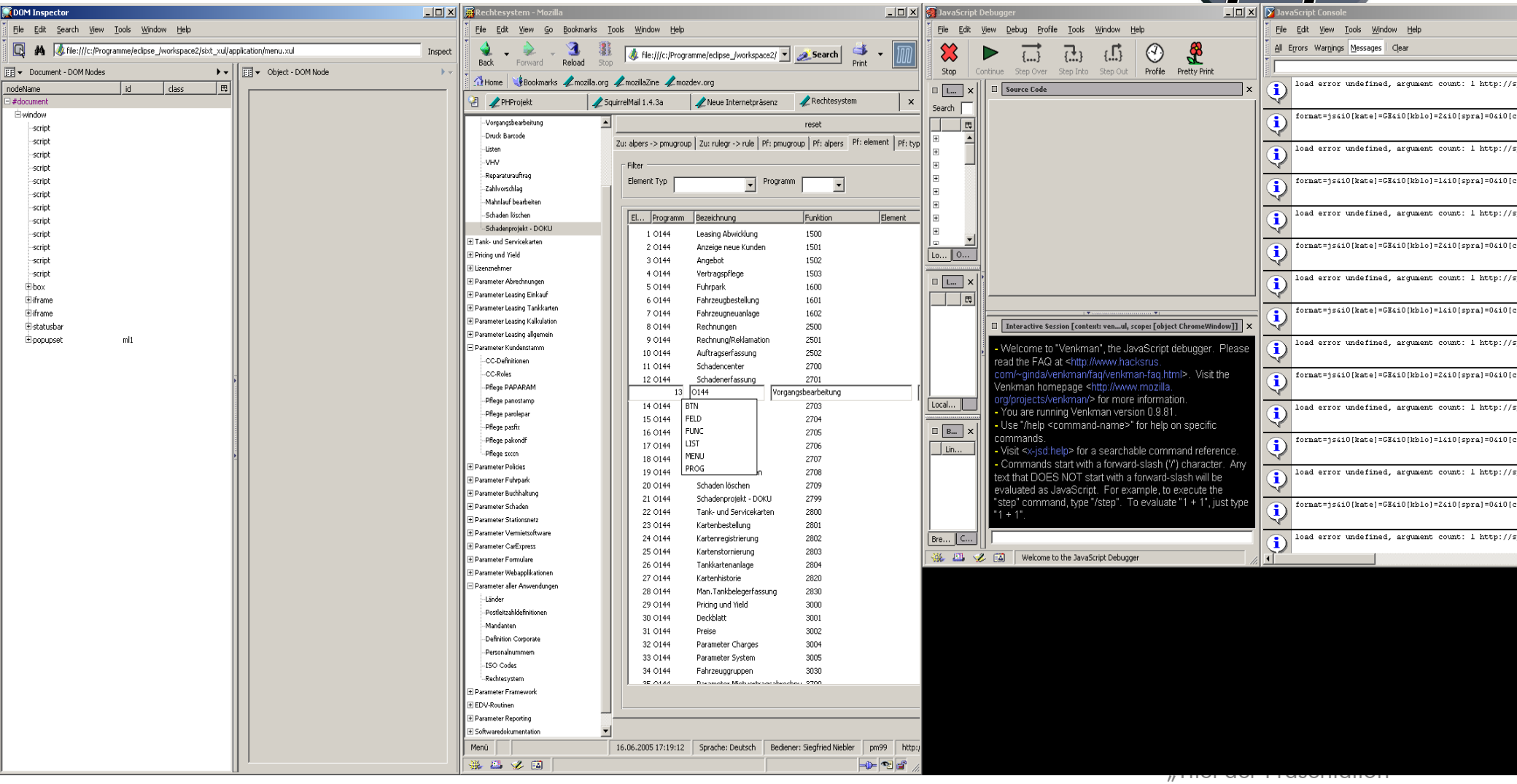


fig. B

- XUL is XML GUI Markup
- Own widgets like <tree />
- XUL runs in the Gecko Engine (Mozilla/Firefox)
- Styling via CSS
- Events and so on with DOM-Javascript

Could speak of it as a „***pimped XHTML***“

XUL development tools



Dive into XUL

- The Broswer supports the most important dev. Tools as browser ext.
  - Firebug as debug console
  - Debugger (Step Debugging – avail. as ext.)
  - DOM Explorer for XML Structures
  - Paranoid javascript.options.strict
- Eclipse and JSEclipse + XMLBuddy as dev. Plattform
- Happy vi users reported

May I XUL ?

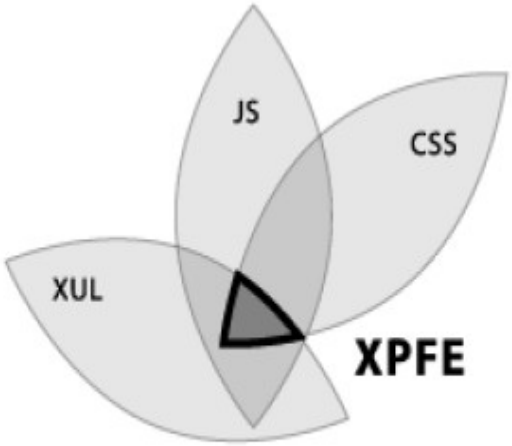


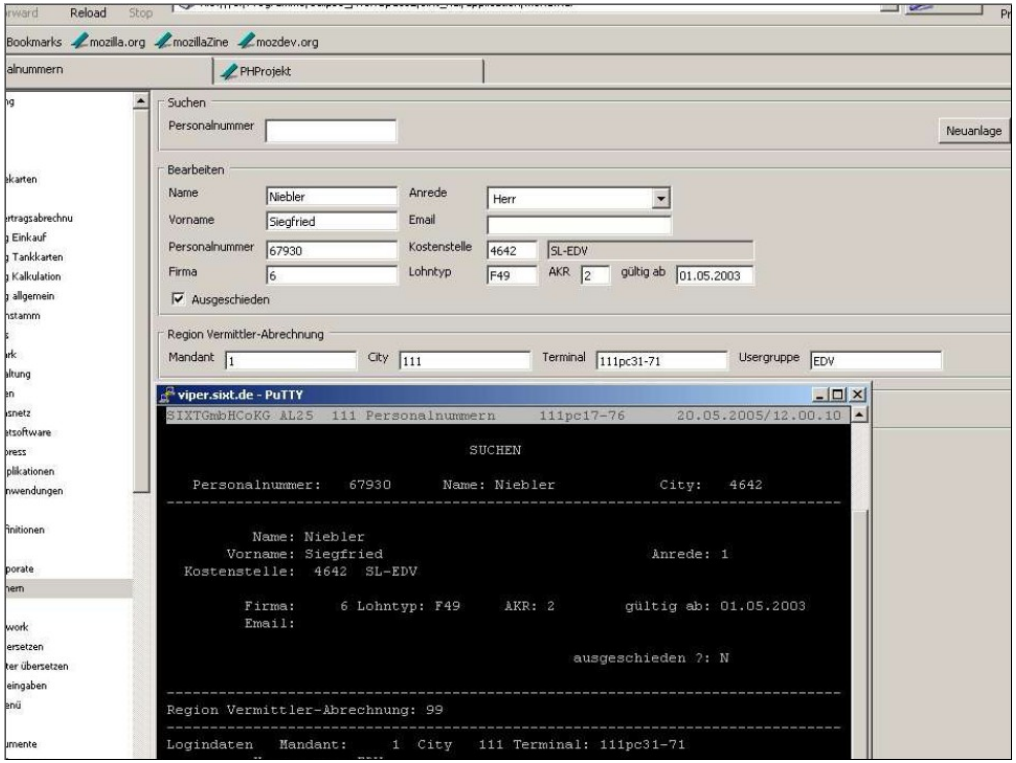
fig. B

- Do you XHTML ?
- Do you CSS ?
- Do you Javascript ?

- Do you C++, python .. ?

The Requirements

- Find a way to implement modern GUI applications on top of a COBOL based enterprise architecture without rewriting from scratch and be fast with it.



The Contenders/Inhouse GUI Technologies

- **COBOL**
  - Only terminal based applications
  - Limited developer ressources avail.
  - hundreds of Applications avail. @ Sixt
- **Powerbuilder**
  - Not open source, long development cycles
  - Island applications as result
- **Java**
  - Very limited number of deployed apps. @ Sixt
  - High development costs

The Solution

- Use XUL/XML/Javascript as **GUI layer** – aka develop a **browser based application (view & controller)**
- **Re-USE** the **stable in house COBOL architecture** via AJAX based access. (model)
- Develop a framework to provide advanced GUI components (**grids, trees**)
- Establish s**imple Deployment Processes** as the Sixt company is used to them.
- As far as possible: Give **Sixt the complete control** over the resulting product

The first Project - Incident Handling

- Companywide use: from callcenter to the legal department (different kind of users!)
- ~ 40 GUIs to enter and process data
- Connected to various serverside systems
  - PDF document generation server
  - E-Mail services
  - DTAUS bank file generation service
- Plattform: Mozilla on Cytrix dektop to minimize deployment issues.

Technical Elements of Sixt XUL Apps.

- Ajax Cobol/PHP/SQL connector object
- Excel like datagrid for serial data
- Treewidget
- Data dictionary with length and fieldtype definitions for clientside validation
- Enumerations (e.g. country lists)
- Helper API (data formats, comparisons etc.)
- Page controller API (DOM events and operations)

AJAX in Sixt XUL

- Data serialization with **JSON**
- Serverside JSON **C Module** for performant (de-) serialization
- **N Cobol-method calls in one http-request (batching)**
- Easy handling of dependent AJAX-Requests
- Large amounts of data transferred with no performance headaches.
- Simple PHP backend archictecture
- COBOL, SQL and PHP methods usable
- Cobol and SQL Calls mixable into one request

N-method calls in one request ?

- Data retrieval of ~ 30 Cobol Methods on in itial loading of a incident case
- Special dependencies (car-contractor-credit cards) make use of dependant calls
- Optimal Re-use of the CRUD and other Cobol Methods in Data-Grids

think php

by MAYFLOWER

An Example ……

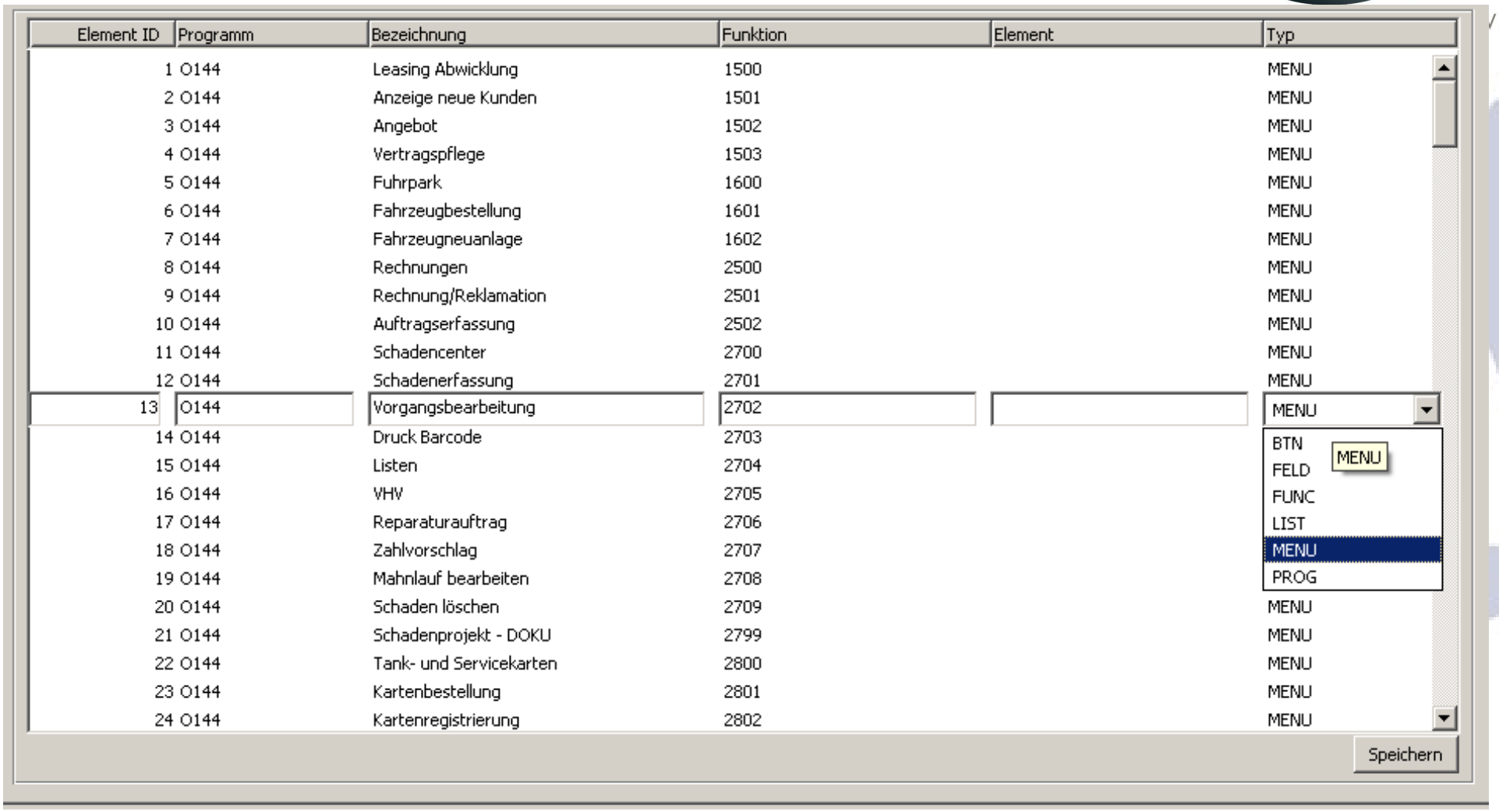
Request Batching - Why ?

- Save request instantiation and authentification
- Reduce concurrency Problems
- Use database transactions in AJAX-SQL
- Simplify errorhandling issues
- Better performance in high latency situations
- Delivers data of multiple function calls at once

Pagecontoller and helper API

- Objects that live in every XUL Page
- Events already bound in there, no need for DOM-attributes like onClick
- Allows clean partitioning of XML View and event dispatching
- Data validation with one line of Javascript code
- Form data-delta checks to avoid useless AJAX Requests
- eases routine developer works

Datagrid

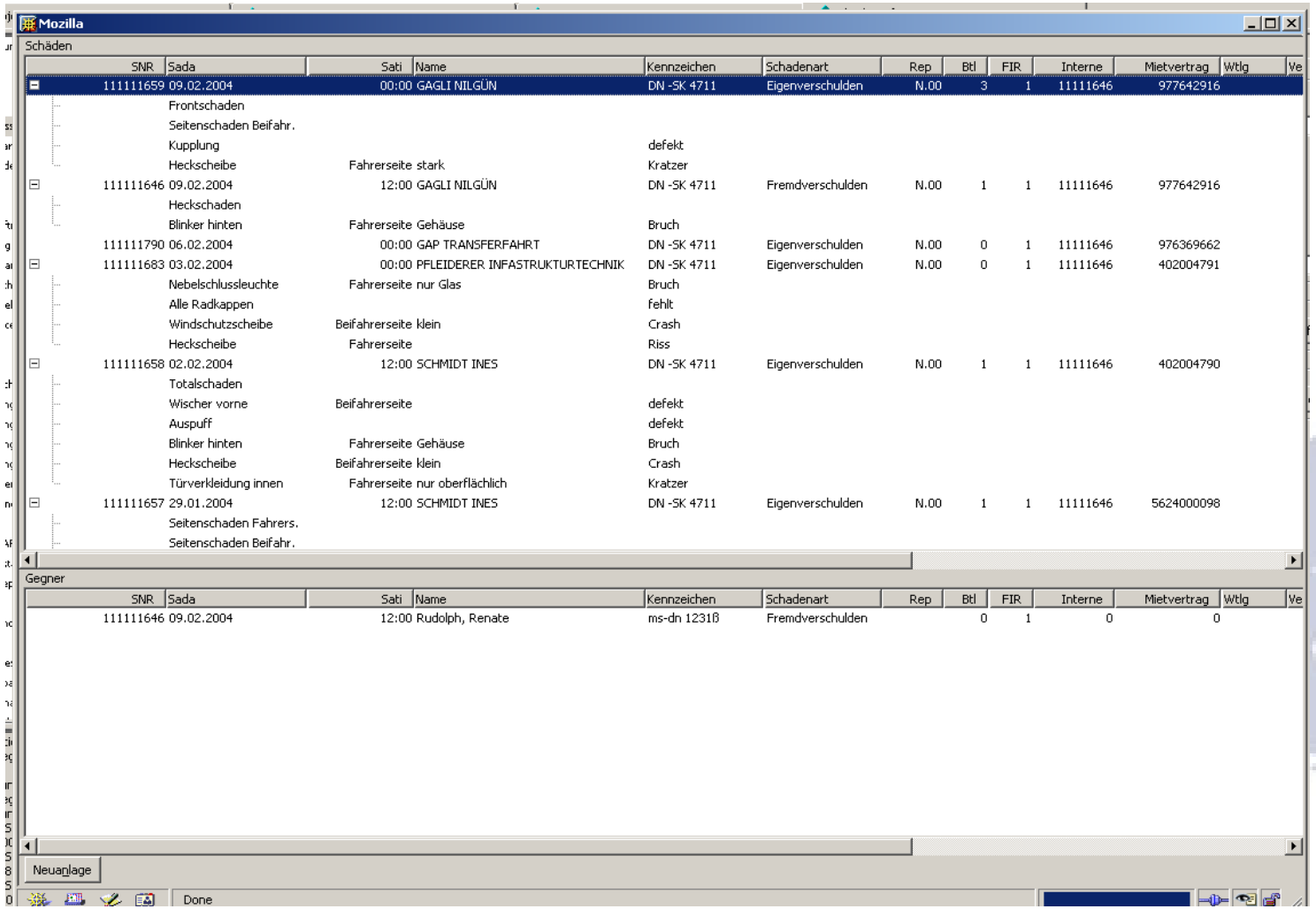


Element ID	Programm	Bezeichnung	Funktion	Element	Typ
1	O144	Leasing Abwicklung	1500		MENU
2	O144	Anzeige neue Kunden	1501		MENU
3	O144	Angebot	1502		MENU
4	O144	Vertragspflege	1503		MENU
5	O144	Fuhrpark	1600		MENU
6	O144	Fahrzeugbestellung	1601		MENU
7	O144	Fahrzeugneuanlage	1602		MENU
8	O144	Rechnungen	2500		MENU
9	O144	Rechnung/Reklamation	2501		MENU
10	O144	Auftragserfassung	2502		MENU
11	O144	Schadencenter	2700		MENU
12	O144	Schadenerfassung	2701		MENU
13	O144	Vorgangsbearbeitung	2702		MENU
14	O144	Druck Barcode	2703		
15	O144	Listen	2704		
16	O144	VHV	2705		
17	O144	Reparaturauftrag	2706		
18	O144	Zahlvorschlag	2707		
19	O144	Mahnlauf bearbeiten	2708		
20	O144	Schaden löschen	2709		MENU
21	O144	Schadenprojekt - DOKU	2799		MENU
22	O144	Tank- und Servicekarten	2800		MENU
23	O144	Kartenbestellung	2801		MENU
24	O144	Kartenregistrierung	2802		MENU

Tabular Data - The Datagrid

- Allows to edit data in a Excel like spreadsheet style
- Allows textfields and selectboxes
- Field based validation when data is changed without a request to the server
- Optimal usabillity for large amounts of data
- Automatic determination of changed data (deltas) and contol over resulting storage operations

Tree



Hierarchic datastructures - Tree

- Simple JS API to
  - Build trees
  - Attach event code to tree-nodes
  - Read tree data
  - Open and close subtrees
- NO Implementation as <XUL/>-tag/XBL
- Several DOM performance optimizations
- Eases complex tasks (list to hierarcy)

Whats so web 2.0 about all this ?

- Mashup of several services in the Sixt IT Landscape
  - Cobol data services
  - PDF generation
  - DTAUS file generation
  - Mailservices
- Browser based application
- Scripting language for GUI and glue layer (PHP)
- Rapid Development Process (RAD)

The Second Project: Styleguides and QA

- Straight naming conventions for XML-attributes and Javascript notation rules (BSD-Style !)
- Clear documentation for Project Managers what is possible and what not.
- Review of subprojects and refactoring of code and markup
- Limitation of possibilitiesover all to streamline the complete sourcecode.
- Guidelines for:
  - XUL markup
  - Javascript code
  - Interface design-guidelines and usabillity

The Third Project: Developer Trainigs

- New Team was formed Summer 2005 ~ 10 people working on XUL @ Sixt
- Different backgrounds and history of candidates
  - Web + Scripting Langs
  - Cobol
  - Powerbuilder
- One Week of training for the „*newbies*“
- First productive work started after 2-3 Weeks

More Projects

- Car leasing application (price calculation etc.)
- Leasing contract systems
- International money transfer via dtazv
- „Connector“ to the datawarehouse
- Extensive security & permission checks, down to the DOM-Level. Different users have different rights in the same application context.

  etc, etc, etc .....

Lesson 1: Code Issues

- Even if its just Javascript, it needs to be clean and performant.
- NO warnings in any JS Code or CSS
- Markup needs to be clean too
- Use or write Frameworks, dont re-invent the wheel, better fix it ;)
- Use a performant and warning-free framework
- Complex async-AJAX Requests require a bit more complex code. Make Sure every developer knows how to
  - Program functional in Javascript
  - use the .prototype keyword (NOT the framework)

Lesson 2: Process Issues

- Ajax Apps and the XUL-Framework are very agile, behave like it
  - release early – release often
  - don't do big Prototypes – start to be really productive in a early stage
  - don't fear refactoring
- Scrum as agile management method fits well to this kind of application development
- Use the right tools, instead of the well known ones
- Automated tests help finding side-effects of changes to the codebase

Lesson 3: Design & Performance Issues

- Serverside Methods, if re-used (optimal situation), might need a additional Layer on the client side (model layer)
- Trade more performance in async requests for complexity in the code
- Do not re-invent the serverside wheel in the client
- Even clients (browsers) might have limited processing power and RAM

More Infos

- mayflower.de
- xulplanet.com
- developer.mozilla.org

by MAYFLOWER

Thanks for listening ;)

Sebastian Schürmann

Mayflower GmbH
Sendlinger Tor. 42a
80331 München

+49 (89) 24 20 54 - 0
schuermann@mayflower.de