NEXT WIKI APPROACH FOR 101 COMPANIES

- Motivation and client design -

Thomas Schmorleiz - 2012, Software Languages Team
MOTIVATION
MOTIVATION

Where are we?

Initial goal: Document contributions, languages, technologies, features, concepts, ...

Current status: 144 (!) contributions
MOTIVATION

How did we do it?

What we needed: Web application to store pages and categories, have revisions and user management

MediaWiki:

• Widely used

• Easy to install and maintain

• Extendable via hooks
MOTIVATION

Hooks?

Idea: Bind some piece of code to a MediaWiki event (e.g.: “pages saved”, “wikiText parsed”…)

Usage:

• Parsing some piece custom piece of wikiText syntax

• Aggregation of links

• Manipulation of layout

Result: Gets the job done, but development is rather painful/hackish
What happened? - We are social!

Community: 101 companies is a community project of different stakeholders: researchers, developers, community engineers, teachers, ...

Third-party user authentication

- OAuth (GitHub, Twitter, Facebook, ...)
- Insufficient support by MediaWiki

Integration of third-party information

- Together with OAuth allows great UX (e.g. contribution)
MOTIVATION

What happened? - Ontology engineering!

101 ontology: Classification all relevant entities: languages, techs, contributions, themes of contributions, ...

Refactorings

• MediaWiki UI only allows Create/Edit/Move/Delete

• We further need to:
  • Promote/demote pages/categories
  • Move all members of a category to another
  • ...

Friday, October 26, 12
MOTIVATION

What happened? - Ontology engineering!

101 ontology: Classification all relevant entities: languages, techs, contributions, themes of contributions, ...

Ontological relationships
• MediaWiki: “instanceOf” and “refersTo”

• We a domain specific relationships:
  • A contribution “implements” a feature
  • A contribution “uses” a technology
  • A technology is “partOf” a technology
MOTIVATION

What happened? - Documentation models

MediaWiki: No support to define models for documents. E.g. contribution page: Must have "Motivation", might have a "Issues"

Validation: Based on document models

Further refactoring: E.g. Renaming "Motivation" to "Summery" in the model rename all motivation sections
DESIGN
DESIGN

Client

Backbone.js

REST API (JSON)

Server

Ruby on Rails

MediaWiki API

Storage

MediaWiki Storage
Oh JavaScript...

• Vanilla JavaScript applications are a mess
  
• How to structure?

• We store object information in the DOM

• Using jQuery for fetching data and events gets you into callback hell > readability suffers, brain implodes
Oh JavaScript...

- Vanilla JavaScript applications are a mess
- How to structure?
- We store object information in the DOM
- Using jQuery for fetching data and events gets you into callback hell > readability suffers, brain implodes
What can Backbone.js do for you?

- Structure! Separating data and UI: MVC
- Abstraction
- Less callbacks
  - E.g.: Instead of some explicit `$.post(...)` use `model.fetch()`
- Model events (to update views)
- Routing
What can Backbone.js do for 101?

• Structure, Abstractions, Readability

• Client models for different kinds of documents, sections

• Views

• Enables us to more easily write better UX
Models (in CoffeeScript):

```coffeescript
class Wiki.Models.Section extends Backbone.Model

  defaults:
    title: null
    content: null

  url : () ->
    "/api/pages/" + Wiki.page.get('title') + "/sections/" + @get('title')

class Wiki.Models.Sections extends Backbone.Collection
  model: Wiki.Models.Section
```
Controlling (in CoffeeScript):

```coffeescript
show: ->
    Wiki.page.set 'sections', new Wiki.Models.Sections()
    $.each $('mw-headline'), ->
        section = new Wiki.Models.Section({title: @id})
        Wiki.page.get('sections').add(section)
        sectionview = new Wiki.Views.Sections(model: section, el: @)
        sectionview.render()
    appview = new Wiki.Views.Pages()
    appview.render()
```
Views (in CoffeeScript):

class Wiki.Views.Pages extends Backbone.View

  render: ->
    # add page title
    $('#.container').prepend(...('<h1>').text(Wiki.page.get('title')))

    # add category links
    $.each Wiki.page.get('categories'), (i,catname) ->
      $('#infofooter').prepend(...$('<span>').text(catname))
TOPICS
TOPICS

• Locks in the new wiki

Having locks for editing is not trivial. We want to enable section editing, but might also have some cross-section constraints, e.g. "Motivation should mention all implemented features". These constraints could be expressed in document models. Working on this topic would imply being a member of the 101wiki front :)
TOPICS

- Query DSL for 101 data

Depending on how we expose our data we might want to have one or another way of querying our data. For JSON, for instance, that could be s.th. on top of jLinq.
• Search interface for the new wiki

We do ontology engineering, so we also want to search it. This topic would be related to (or merged with) the query topic. This topic would also include coming up with some interface to compose queries: "Give me all implementations of theme X that use language Y and technology Z".