

# Building social applications with StatusNet

Evan Prodromou  
Zach Copley  
Brion Vibber

# Section 1: Overview

# Social stream model

- Each account generates activities
- Activities have properties
- Other users filter activities into streams
- Stream = reverse chronological list

# Blogging

- Similar model
- Generator and reader are the same software

# Available streams

- Public
- Group
- Tag
- Search
- List

# What's an “Activity”?

- Subject Verb Object
- Somebody did something
- Evan posted an Event
- Brion commented on a Note
- Zach listened to a Song

# Activity properties

- Author
- Category or tag
- Object
- Attachments
- Addressing (to Brion, to Dev Group)
- Scope (privacy)

# Profiles

- “My activities”

# Following

- Users can follow each other
- Asymmetric follow
- You can control who can follow you
- “Inbox” stream shows notices from people you follow

# Groups

- Opt-in
- Like a mailing list
- Receive all activities posted to the group
- Only members can post to the group

# Lists

- User-curated groups
- Addressable
- Followable
- Streams
- Compare: Google+ Circles

# Scope (“Privacy”)

- Limit who can see/reply to activities
- Different kinds of scope
  - Only followers (“private account”)
  - Only group (“private groups”)
  - Only this site
  - Only addressed individuals

# Federation

- Social connections extend beyond site boundaries
- Subscription-based
- OStatus: PuSH, Salmon, ActivityStreams, WebFinger

# Bridges

- IM systems: XMPP, AIM, IRC, MSN
- Facebook
- Twitter
- SMS (over email)
- Less data across “narrow” channels (plain text)

# Platform

- LAMP
  - Linux
  - Apache
  - MySQL
  - PHP
- memcached
- Queue server
- Extensible to use XMPP, other channels
- Off-line daemons for some systems

# Hands-on 1: Installation

# Installation prereqs

- Apache
- PHP
- MySQL

# Download the software

- <http://status.net/download>

# Create the database

- `mysqladmin create statusnet_db`
- `grant all on statusnet_db.* to 'statusnetuser'@'localhost' identified by 'statusnetpass'`

# Fancy URLs

- move `htaccess.example` to `.htaccess`
- Change the embedded path `/mublog/` to your real path

# Run the install program

- `/path/install.php`

# Enable/disable privacy

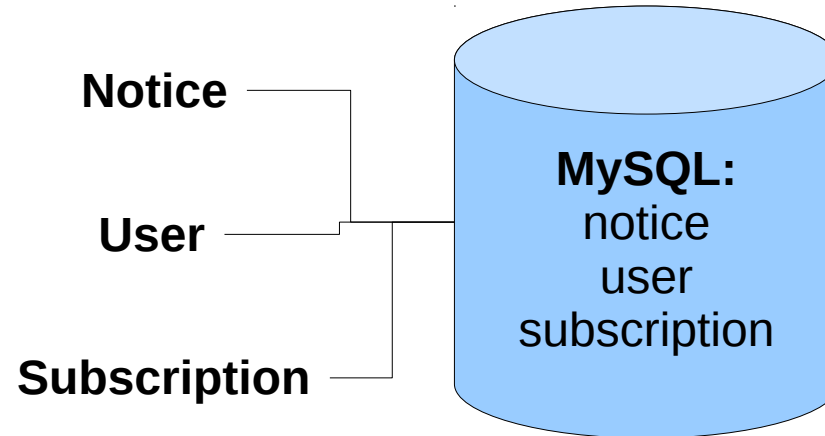
- `$config['site']['private'] = false;`

Click around

# Section 2: Architecture

# MVC (Kinda)

- Data classes



- Actions

**AllAction**

**PluginsAdminPanelAction**

- Router

/:nickname/all

/panel/plugins

# Data classes

- DB\_DataObject
- Memcached\_DataObject
- Managed\_DataObject

# Actions

- “actionname” => ActionnameAction
- prepare()
- handle()
- Output methods

# HTML output

- `elementStart()`
- `elementEnd()`
- `element()`
- `text()`
- `raw()`
- HTML-aware methods like `checkbox()`

# Widgets

- `__construct`
- `show()`

The image shows a social media interface with two main sections. The top section is a status creation form with tabs for 'Status', 'Draw', 'Music', 'Bookmark', 'Event', 'Poll', and 'Question'. The 'Status' tab is selected. Below the tabs is a large text input field with a character count of 140. Below the input field is a 'To:' dropdown menu set to 'My colleagues at Mublog' and a 'Private?' checkbox. A red box highlights the input field, the 'To:' dropdown, and the 'SEND' button. The bottom section shows a user profile for 'brionv' with a blue profile picture and an 'Edit' button. Below the profile are two sections: 'FOLLOWING 0' and 'FOLLOWERS 0', both showing '(None)'. Below the profile are two status posts, each with a red box highlighting the text and interaction icons (heart, reply, share). The first post is 'listened to Hey Bulldog by The Beatles about 21 minutes ago from atompub'. The second post is 'listened to Don't fear the Reaper by Blue Oyster Culdt about 22 minutes ago from atompub'.

# Page layout methods

- Recipe for building a page
- Overload them for your action
- `showContent()`
- Look in `lib/action.php`

# Event hooks

- Register a handler
- Called when something happens

# Example events to hook

- Data events (new notice, deleted user)
- UI events (main menu started, footer shown)
- Business rules (authorization, login)
- Camel-case (StartSaveNotice)

# Plugins

- OOP-y way to handle events
- “EventName” => onEventName()
- addPlugin()

# Schema maintenance

- CheckSchema event
- Schema class

# Configuration file

- `config.php` (or `/etc/statusnet.php`)
- Database (Config class)

# Hands-on 2: Hello world

- Add a new plugin
- Add a new action
- Modify the UI
- Modify the router

# Create HelloWorldPlugin.php

- in local/plugins/HelloWorldPlugin.php

# add a menu item

- `onEndPrimaryNav($action)`

# onAutoload()

- Load hello.php for HelloAction

# onRouterInitialize()

- map main/hello URL to hello action

# HelloAction

- extend Action class
- overload title() method
- overload handle() method
- overload showContent()

# Section 3: microapps

# ActivityStreams

- Data classes for social activities
- subject, verb, object
- identity == URI
- extensible types

# ActivityStrea.ms verbs

- Post
- Follow
- Fave
- Play
- Share

# ActivityStrea.ms object types

- Person
- Article
- Photo
- Audio
- Group

# Serialization

- Atom (default)
- RSS (kind of)
- JSON (preferred)

# Extensions ideas

- CRM
- Bug reports
- Task management
- Poll
- Q&A

# microappplugin

- makes it easy to create new activity streams schema
- interface class all microapps must implement
- additional classes for actions, data types

# Microapp examples

- Event
- Poll
- Q&A
- Blog
- Bookmark

# Defining new microapps

- data types
- save/delete actions
- entry form
- HTML format
- Serialization

# Hands-on 3: Social music plugin

# SocialMusicPlugin

# onAutoload()

- Leave it empty for now!

# onVersionInfo()

- Add version info

# title()

- Title for use on input
- “Music”

# tag()

- Used for various HTML IDs
- “music”

# types()

- Array of URIs of object types

# Hands-on 4: Social music data

# Song play activity

- UUID for identity
- artist name
- title
- URI
- created & modified

# Example only!

- Better example would use MusicBrainz IDs

# Note on URIs

- Universal identity
- Use HTTP URIs

# Notice and Listen

- URI for identity
- `getNotice()`
- `fromNotice()`

schemaDef()

onSchemaCheck()

# Hands-on 4: Social music widgets

# SongPlayForm

- Extends Form

# SongPlayListItem

- Extends NoticeListItemAdapter

Update onAutoload()

Overload entryForm()

Overload adaptNoticeListItem()

# Hands-on 5: Social music actions

# NewsongplayAction

- `prepare()` to validate input
- `handle()` to save values

# ShowSongplayAction

- Used for URI
- Overload showNotice action
- prepare() validates input
- showContent() shows content

# Hands-on 6: Data encoding

# From Activity

- `saveNoticeFromActivity()`

# To Activity

- `activityObjectFromNotice()`

**Test it!**

# Further enhancements

- Song/Album/Artist streams
- Music clouds
- Recommendations

# Section 4: API

# Twitter-like API

- Copy of Twitter's API
- Oriented to plain text
- Social graph
- Pro: lots of libraries
- Con: just plain text

# AtomPub API

- Model social graph with activities
- Evan followed Brion
- Zach joined Dev group
- Richer activity-streams posting

# Step 7: Scrobbler

# StatusNet command-line

- continue using PHP
- get acquainted with more of StatusNet
- **Not** using internal data!

# AtomPubClient

- Build a “listen” activity
- Post it