Seaside

Dynamic Web Application Development
through Object-Orientation

Roadmap

- History

- Features

- Live Coding

- Seaside Today

- Discussion

History

- State of Web application development:

  - Server side or client side?

  - Many people were blocking javascript by default

    - Web Apps had to be server side

  - User-readable URLs were not a must

  - Ajax did not exist until 2005 (kickoff for Javascripts popularity)

History

- Avi Bryant wanted to build interactive Squeak Wiki

- First Release 0.9 in February 2002 by Avi Bryant & Julian Fitzell, Open Source with MIT License

- Inspired by Apple's WebObjects (Objective-C)

- Quick development, version 2.0 came October 2003

- Very modern: Interactive date picker

Seaside

- Web Framework for Smalltalk (Squeak, Pharo, Gemstone, etc.)

  - Purely Object Oriented

  - Works through Model View Controller

  - no pages: UI is a tree of individual, stateful and reusable component objects

- GLASS Stack:
  - GemStone (database and application server)
  - Linux (operating system)
  - Apache (web server)
  - Seaside (web framework)
  - Smalltalk (programming language)

Features

- Incredibly flexible developer tools:

  - Live debugging and code changing while user interacts with web app

  - debugger in browser (tobi nach Einstellung fragen)

  - no HTML, render page as Morph -> no browser needed

  - Reusable *components*

- hides synchronicity -> no callback hell

- CSS, Ajax, Comet & Javascript support

- Session for each user

Main Menu

- Categories
- Search
- Unreviewed
- Users
- Settings
- Logout

Search Tags

Desktop Job Quest Skills Trek Task Web

Web Trek Connect

detail | edit | move | delete

Job Quest

detail | edit | move | delete

Discovery Desktop

detail | edit | move | delete

Web Trek

detail | edit | move | delete

Comp Skills

detail | edit | move | delete

AIMS Task Builder

detail | edit | move | delete

Components

Features

- Incredibly flexible developer tools:

  - Live debugging and code changing while user interacts with web app

  - debugger in browser

  - no HTML, render page as Morph -> no browser needed

  - Reusable *components*

- hides synchronicity -> no callback hell

- CSS, Ajax, Comet & Javascript support

- Session for each user

Live Coding

Downsides

- Each session object can be multiple megabytes big

- If server crashes, all sessions are lost

    - Gemstone distributed but expensive

- Javascript not easily debuggable

- Very hard to implement human readable URLs

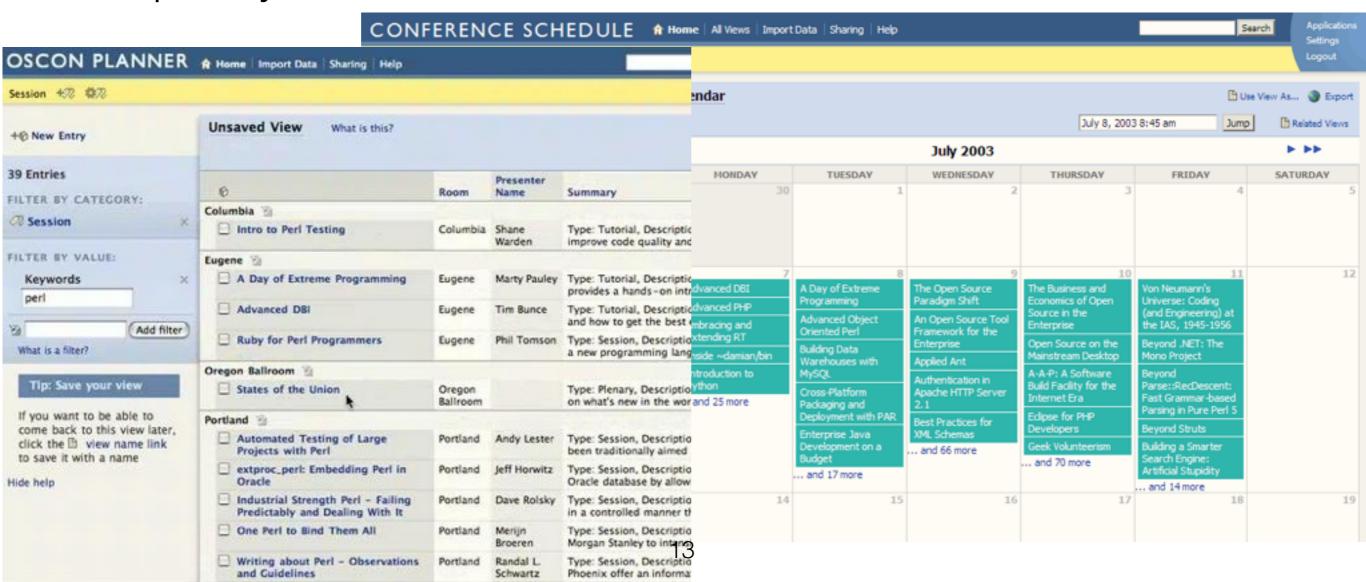- Not on par with today's demands of a fast dynamic webpage since its server side

Seaside Today

- Dabble DB was 'killer app' built by Avi Bryant himself

  - Was bought by twitter for real time analytics

- Still in use, example: *yesplan*

- Latest version 3.2 from May 2016

- Google Web Toolkit tried similar approach

Dabble DB

- Database management and analytics without SQL

- purely visual and interactive

Seaside Today

- Dabble DB was 'killer app' built by Avi Bryant himself

  - was bought by twitter for real time analytics

- still in use, example: *yesplan*

- latest version 3.2 from May 2016

- Google Web Toolkit tried similar approach

Discussion

- Why would people not use technologies such as Seaside or Google Web Toolkit?

- What do you think about it having such a monopoly, that people do not look for solutions anywhere else?