"Advanced" Web Technologies in LCLS/PCDS (personal quest)

Igor Gaponenko
SLAC National Accelerator Laboratory, USA

SLAC/PPA Computing Department Meeting
January 7, 2011

What is in this talk

- Use case for Web at PCDS
- Technologies we use
- New trends in Web (as we see it)

...and what is not

- **In general, 99.9...% of existing technologies isn't covered. So, don't expect the full picture for the "advanced"! The Web world is huge and beyond our limits to capture it in full!**

- **Our choice was driven by the following factors:**
  - Needs of experiments and our development schedule
  - "What many/most others are using"
  - "What's going to be still usable and supported tomorrow"
  - Personal taste and professional experience

- **Specifically, we do NOT use or plan on using:**
  - Any commercial tools, from Microsoft, IBM, ORACLE, etc.
  - Any Java-based techs (Tomcat, J2EE, Java Applets, etc.)

- **Also, this talk will NOT cover LCLS/PCDS OFFLINE**

USE CASE FOR LCLS/PCDS

How it all started...

- **Roughly 5 months before LCLS had the first experiment (AMO Commissioning, Sep 30, 2009) the following surprising (to us ☺) question was asked:**
  - Which Electronic Logbook we're going to use?

- **After some research of existing products we came to a conclusion that no single implementation would meet our requirements. Hence:**
  - We decided to implement our own
  - And we had to decide on candidate technologies (discussed later)
  - Since none of us had any serious experience in Web development we hired a contractor to work on the project (the idea failed. Interested why?)

- **The first usable version was ready ~1 month before the first experiment started. Here is the very first message posted:**
  - https://pswww.slac.stanford.edu/apps-dev/logbook/index.php?action=select_message&id=14

Web Applications & Services

And this is what we have today

- **Home grown applications:**
  - Experiments Registry: https://psdev.slac.stanford.edu/apps/regdb/
  - Authorization Database: https://psdev.slac.stanford.edu/apps/authdb/
  - File Catalog: https://psdev.slac.stanford.edu/apps/explorer/
  - Electronic LogBook: https://psdev.slac.stanford.edu/apps/logbook/
  - LDAP group management:
  - Experiment switch:
  - HDF5 translation management

Quick demo of each will be provided as this talk will progress…

- **Services:**
  - Experiments Registry
  - Authorization
  - File Catalog (programmatic interface to iRODS)
  - E-log printer (to post new messages from non-Web based applications)
  - And many more…(most are not documented)

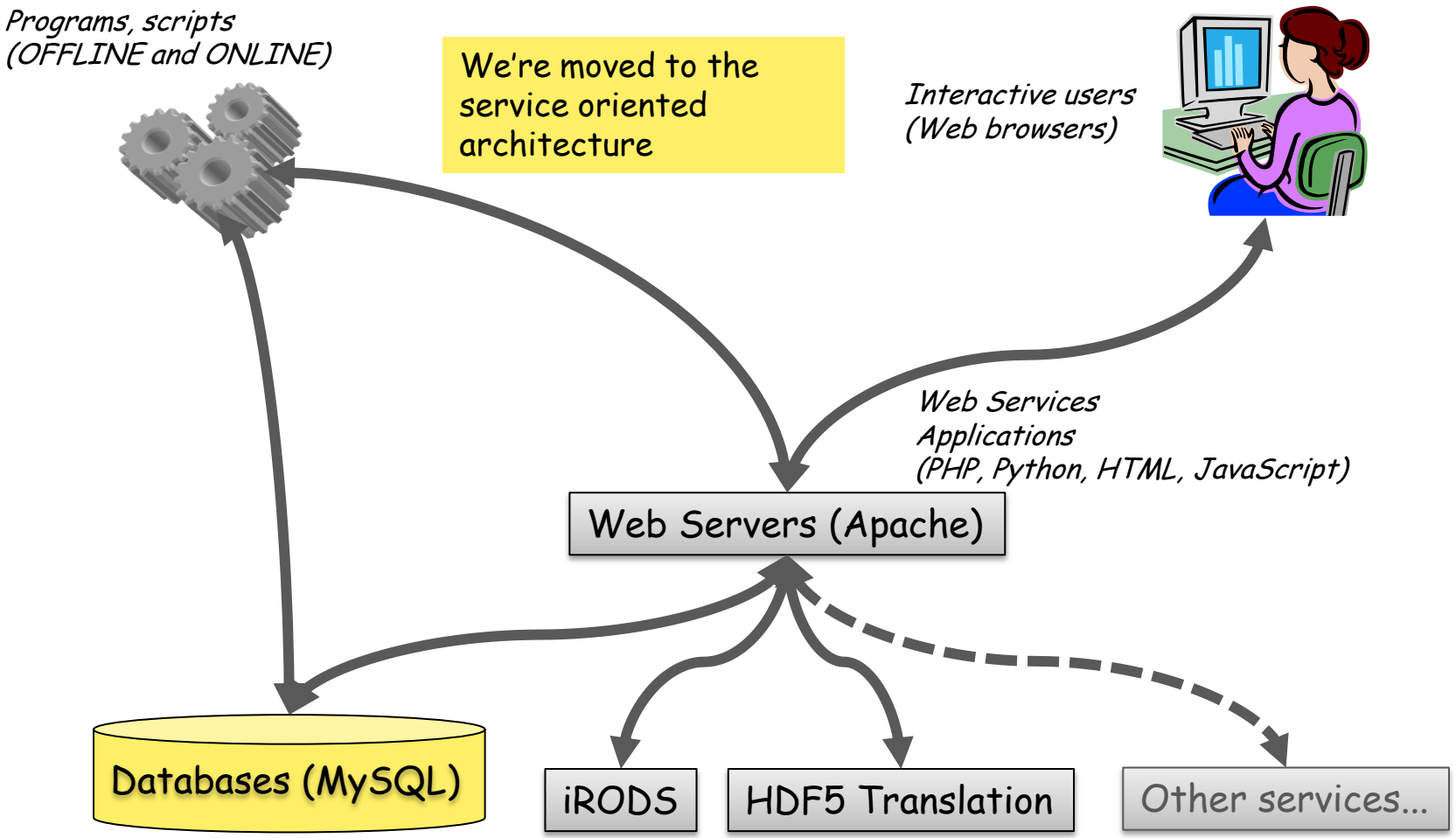
- **Third-party applications**
  - TRAC: https://pswww.slac.stanford.edu/trac/controls
  - BuildBot: http://psdev.slac.stanford.edu:8010/

Databases & Web

Important to mention before proceeding with the rest

- **Databases are the cornerstone of the OFFLINE system**
- **Some limited use by ONLINE as well:**
  - to report various information about runs, raw data files
  - request registration/configuration records for experiments, etc.
- **What we store:**
  - Experiments registration & configuration info
  - Authorization records (will be explained on another slide of this talk)
  - Electronic Logbook data (everything, including images and any documents)
  - File Manager catalog & metadata info
  - TRAC data
  - XTC-to-HDF5 translation status, history
  - Etc.
- **Most of this information is made available/used by Web applications**

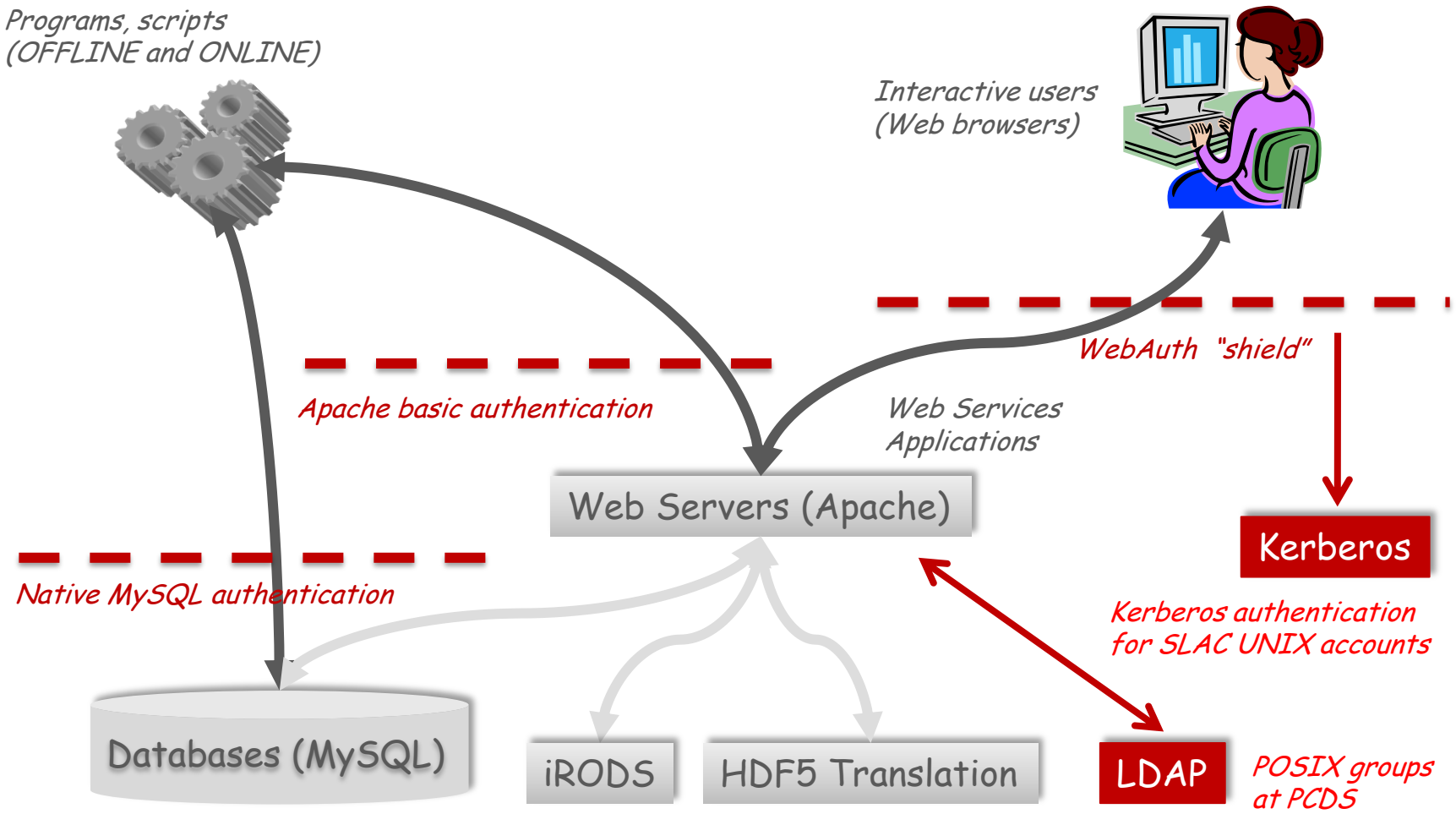
Today's Architecture



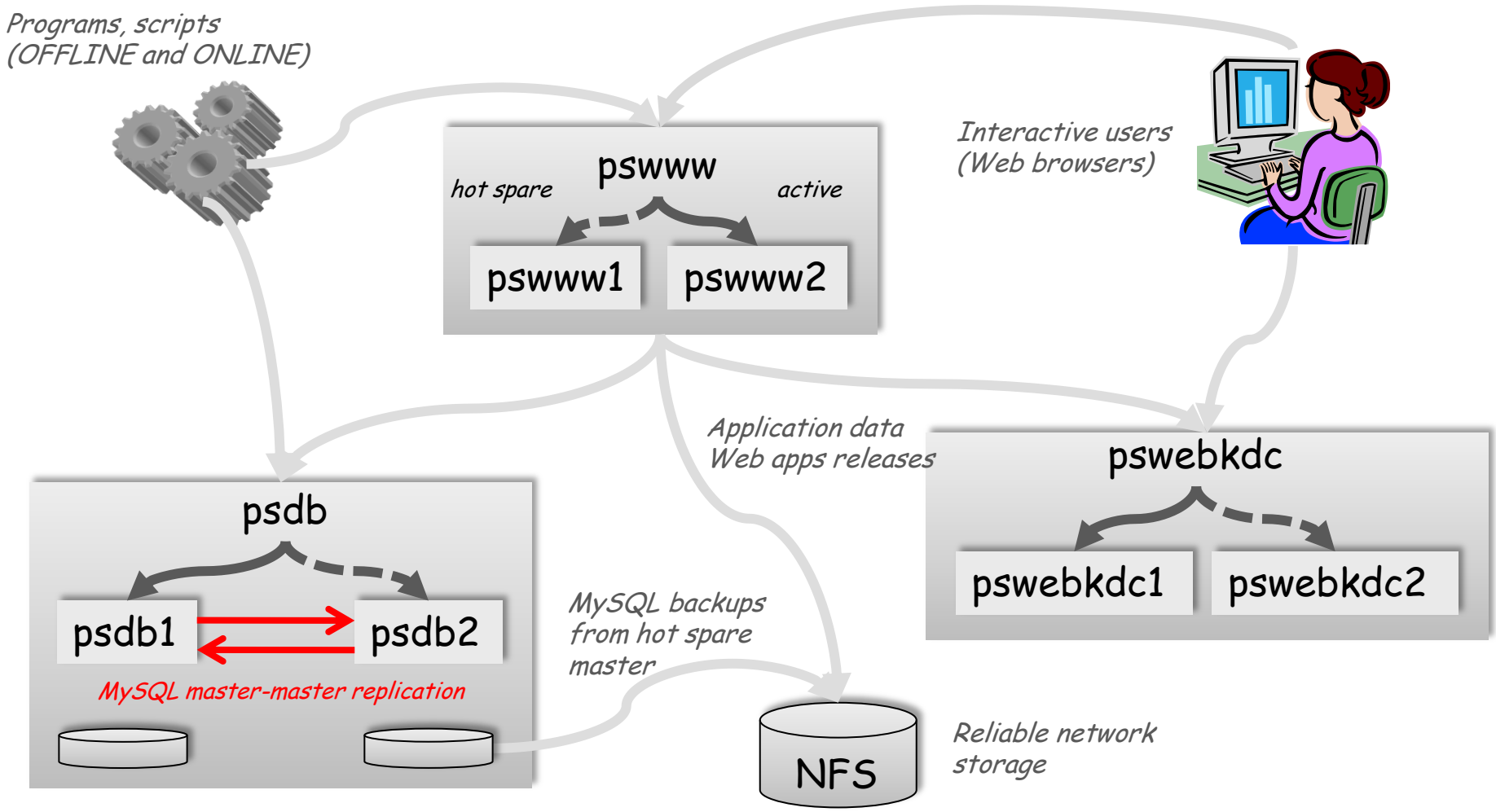
Security

- **Authentication**
  - WebKDC (show a highlighted version of the Architecture with Kerberos & LDAP added).
    - => Each user must have a valid SLAC UNIX account!!!
  - Apache basic authentication for Web services (limited use)
- **Authorization**
  - Role based model combining:
    - (user, application+role, experiment)
  - Each role has a set of privileges
  - Authorization is made for individual users or POSIX groups
  - Authorization records are stored in a database
  - Applications and services can access it
- **Example:**
  - "user **gapon** is an **e-log Editor** of LCLS experiment **amo14410**"
  - "**e-log Editor** is allowed to **view**, **post**, **edit** or **delete** messages, **manage shifts**"
- **Authorization manager:**
  - https://pswww.slac.stanford.edu/apps-dev/authdb/

Security of Web apps



24/7 (reliable infrastructure)



TECHNOLOGIES WE USE

The problem of choice

- **Initial battle (1.5+ year ago) was over Java vs PHP ☺**
  - The later won, even though, none of us really had any serious experience with PHP beyond "Hello World!"

- **It turned out PHP was "as easy as Perl…or C++". More serious problem was what to do with GUI:**
  - A quote from someone: *"…if you're lucky you may get a chance to hire a good Web designer…"* (not a developer!!!). We weren't.
  - A somewhat risky approach was chosen (see the next slide for more)

- **Other considerations**
  - Stay with "basic" technologies and not involve any complex framework or development tools which would require a steep learning curve. Another reason for that layered technologies introduce extra dependencies into an architecture, and they also make it more fragile and less stable in the long run.
  - Commercial products were ruled out from the very beginning
- **Later, we expanded a set of technologies and added Python, installed third-party applications and tools**

The Big Idea (for UI)

- **A contemporary Web browser is actually a small operating system**
  - It can run complex applications (JavaScript)
  - JavaScript supports asynchronous processes (via timers)
  - Applications can communicate with Web services to load data
  - JavaScript can modify DOM (Document Object Model) – what the browser displays and how it displays it
- **A computer on which a browser is run has plenty of resources to do more than just displaying a static HTML page or an image**

- => **HENCE:**
  - Move GUI generation into a browser
  - Use Web server as a data source and a service provider
- **Benefits:**
  - (Much) faster and more dynamic user interfaces
  - Separate data from presentation
  - Less data to transfer

This trend further expands with HTML5. See the end of the talk. Also recall the demo run at the beginning of the presentation.

JavaScript

- **Scripting language supported by all browsers**
- **It's been around since 1994(?)**
- **Standard exists: ECMA-262**
  - http://en.wikipedia.org/wiki/ECMAScript

- **Prototype language: true OO in a sense of being literally object-oriented (not class-oriented). There are no classes in the language, only objects.**
- **Dynamic types**
- **Full access to DOM (Document Object Model)**
  - The main perpose of JavaScript is to manipulate DOM
- **Known Issues/criticism:**
  - No threads, no byte-code compilation, some browser compatibility problems
  - Inconsistent, hard to debug (100% true!), no modular structure
- **Demo (source code of any Web apps):**
  - https://pswww.slac.stanford.edu/apps-dev/logbook
  - view-source:https://pswww.slac.stanford.edu/apps-dev/logbook/
  - Open Chrome Tools and watch changes in DOM while using the application

AJAX

- **The real power of JavaScript came with AJAX (around 2006):**
  - **A**synchronious **J**avaScript **A**nd **XML**
  - http://en.wikipedia.org/wiki/Ajax_(programming)
- **This is actually a group of technologies. Quite often used within a framework:**
  - http://en.wikipedia.org/wiki/List_of_Ajax_frameworks
- **With Ajax, web applications (JavaScript programs run within browsers) can retrieve data from the server asynchronously in the background without interfering with the display and behavior of the existing page. Data is usually retrieved using the *XMLHttpRequest* object. Despite the name, the use of XML is not needed (we use JSON), and the requests need not be asynchronous.**

- **Demos (browsing iRODS catalogs, Google Finance?):**
  - https://pswww.slac.stanford.edu/apps-dev/explorer/

Yahoo! User Interface Lib.

For GUI

- **"The YUI Library is a set of utilities and controls, written with JavaScript and CSS, for building richly interactive web applications using techniques such as DOM scripting, DHTML and AJAX. YUI is available under a BSD license and is free for all uses"**
  - http://developer.yahoo.com/yui/
- **My personal inspiration to try it was Yahoo! Mail**
- **We still use version 2.8.x. It has a rich set of widgets and tools:**
  - http://developer.yahoo.com/yui/2/
- **Version 3 is available. It's not as complete though.**
  - My personal impression – developers either "run out of steam" or a better UI caught their attention (JQuery UI will be discussed later in this talk)
- **Issues (came up after 1+ year of using):**
  - Too complex to use and to develop the code; the hierarchy of classes is too complicated (even for the Web UI development)
  - Forces a particular way of designing UI; not really a toolkit/library but a complete framework
- **Demo: any Web app from PCDS**

PHP

Most of the server-side code in our Web apps is written PHP
We use version 5.3.x

- **A general purpose object-oriented scripting language:**
  - http://www.php.net/
  - http://en.wikipedia.org/wiki/PHP
- **Very popular:**
  - (According to Wiklipedia) In April 2007: 75% of all Web servers served PHP generated pages
- **The syntax is close to C and Perl**
- **Huge library of modules and tools cover everything one may need**
- **Initially used as a server side programming language to produce dynamic Web pages. Can be used for anything else though.**

- **Apache module "mod_php" runs an interpreter for files ending with .php**
- **Very fast (for the scripting language)**
  - Byte-code caching supported by Apache (deployed recently, significant speed up of apps: x10)
- **Numerious frameworks exists:**
  - http://en.wikipedia.org/wiki/Comparison_of_web_application_frameworks#PHP_2
  - We are NOT using ANY of those!

- **Demo:**
  - view-source:https://pswww.slac.stanford.edu/apps-dev/tests/php_demo.php.txt
  - https://pswww.slac.stanford.edu/apps-dev/tests/php_demo.php

Python: WSGI and Pylons

- **Used primarily by Andy Salnikov for Python based Web services**

- **WSGI – Web Server Gateway Interface**
  - http://en.wikipedia.org/wiki/Web_Server_Gateway_Interface
  - http://wsgi.org/wsgi/
  - http://www.python.org/dev/peps/pep-0333/

- **Pylons - is an open source web application framework written in Python. It makes extensive use of the Web Server Gateway Interface standard to promote re-usability and to separate functionality into distinct modules. It is strongly influenced by Ruby on Rails.**
  - http://en.wikipedia.org/wiki/Pylons_%28web_framework%29
  - http://pylonshq.com/
- **Services were designed based on RESTful principle in mind:**
  - Operations are mapped to RESTful resources and HTTP methods (POST, GET, etc.). Resources are identified with URLs.
  - http://en.wikipedia.org/wiki/Representational_State_Transfer#RESTful_web_services
  - See our documentation at:
  - https://confluence.slac.stanford.edu/display/PCDS/Web+Services

New Developments

- **Technologies for next generation applications:**
  - JQuery: http://jquery.com/ ("Write less do more!")
  - JQuery UI: http://jqueryui.com/

- **Data Portal**
  - The previous approach was application centric; the UI was too convoluted for regular/occasional users
  - Based on 1 year of operations we decided to shift the gear to experiment-centric UI
  - Re-factor UI to use JQuery instead of YUI
- **Shifting toward the service-oriented architecture**
- **To improve communication with our users (in cooperation with OCIO):**
  - Forum (FUDForum or phpBB)
  - Confluence integration with Crowd (LDAP + Kerberos)

NEW TRENDS IN WEB

HTML 5

- **The next generation Web technology which would finally turn a browser into a portable operating system**
- **Supported by advanced browsers (Chrome, Firefox 4.)**
- **See demo at:**
  - http://slides.html5rocks.com/#landing-slide

WebGL, etc.

- **Bringing OpenGL 2D/3D graphics into HTML & JavaScript**
- **Utilizing local GPU resources**
- **It's a standard directly supported by the new generation of browsers**
  - No add-ons, plug-ins, etc. needed
- **Promises:**
  - New kinds of computer games
  - Rich visualization (think about event display for SuperB or Fermi)

- **More info from the standardization organization:**
  - http://www.khronos.org/
    - Also note the related technology **COLLADA** – XML based schema for transporting 3D assets between applications

- **See demos:**
  - http://www.khronos.org/webgl/wiki/Demo_Repository
  - http://www.ninepointfive.org/