Next Gen Web Architecture for the Cloud Era

Darryl Nelson
Chief Scientist, Raytheon

Saturn 2013
28 Apr - 3 May

Copyright © (2013) Raytheon
Agenda

• Existing Web Application Architecture
• SOFEA
• Lessons learned
Anyone interested in web technology who has a basic understanding of web applications and Service Oriented Architectures (SOA)
Focus

Data Sources → Big Data / Enterprise Infrastructure → Visualization, Dashboards, Reports
Arc of Web App Architecture History *

- Early 90's
  - CGI
  - Servlets
- Web 1.0
  - MVC Frameworks
  - Web Templating Engines
- Web 2.0
  - AJAX
- Today
  - Mobile
  - SOFEA

* NOT to scale!
Web Templating Engines

- Embedded code within static HTML elements
- Mix of static and dynamic HTML
- "Model 1" Architecture

Examples
- Java Server Pages (JSP)
- PHP
- Active Server Pages (ASP)
Web Templating Engines cont.

Web Template

```html
<html>
Hello,
<b>${db.name.102}</b>
</html>
```

Code

Markup

Web Template Engine

Web Browser

Hello, Bob

01 Ted
02 Susan
.
.
101 Joe
102 Bob

Data Store

Darryl Nelson

SATURN 2013
Copyright©(2012) Raytheon Company
MVC Frameworks

- **Model View Controller** pattern
- Server side framework
- “Model 2” Architecture
- Examples
  - ASP.NET MVC Framework (.Net)
  - Struts, Spring MVC (Java)
  - Ruby on Rails (Ruby)
  - Django (Python)
  - Grails (Groovy)
AJAX

- **Asynchronous JavaScript And XML**
- Dynamic content changes without reloading the entire page
  - interactive and dynamic web apps approaching rich client capability
- **HTML/CSS + DOM + XMLHttpRequest Object + JavaScript + JSON/XML**
3 Processes of Web Applications

1. **Application Download**
   Mobile code (JavaScript, HTML, Applets, Flash) download to the client (web browser)

2. **Presentation Flow**
   Dynamic visual rendering of the UI (screen changes, new screens, etc) in response to user input and data state changes

3. **Data Interchange**
   The exchange of data between two software components or tiers (search, updates, retrieval, etc)
Process Allocation for Web Templating Engines Frameworks

Application Download, Presentation Flow

Application
Browser

Network

Presentation Logic (web server)

Business Logic

Persistence

Data Interchange

Service Interface
Characteristics of Web Templating Engines and MVC Frameworks

• Tight coupling between presentation flow and data interchange (both in the web server)
  – Triggering a Presentation Flow in a web application always initiates a Data Interchange operation
  – Every Data Interchange operation results in a Presentation Flow operation

• Presentation flow and data interchange are orthogonal concerns that should be decoupled
  – Separate concerns
Today

web templating engines +
MVC frameworks +
a *sprinkling* of Ajax
SOA & Cloud

Web Apps

Mobile Devices

Machine-to-Machine

Cloud Stack

SOA Stack

INTERFACE
Endpoints

SERVICE
BATTERY

GOVERNANCE

SaaS

PaaS

IaaS

Data Schema

Policies

External Cloud
SOFEA

An architectural style for web applications in SOA (& Cloud) environments
- **Service Oriented Front End Architecture**  
  - Synonymous with “Single Page” Web Applications

- **Life above the Service Tier**  
  How to Build Application Front-ends in a Service-Oriented World  
  - Ganesh Prasad, Rajat Taneja, Vikrant Todankar

- **Architectural Style**  
  - Not an implementation

- Prasad, et al propose that the SOA revolution has left behind application front ends/UI’s
SOFEA is now...

Feasible because

1) Maturity of the SOA paradigm in theory and practice
2) Advancements in browser-based client technologies, especially JavaScript browser engines and AJAX toolkits

Necessary because

1) SOA is the defacto delivery mechanism for cloud-based services (Cloud and SOA are complementary technologies)
2) Diversity of client platforms
   - Growing dominance of Mobile clients
Legacy Enterprise Web Architecture

CLIENT

Web Browser

SERVER

Web Page Construction Logic

Business Logic and Persistence

Typical Enterprise Web Application Architecture

JSP, ASP, PHP, RoR, etc.
Process Allocation for SOFEA

Presentation Flow

Application Download

Network

Service Interface

Service

Persistence

Data Interchange

Presentation Logic

Application

Browser
3 Web Processes and SOFEA

Web Browser

Application Download

Download Server

Presentation Flow

Data Interchange

Business Logic and Persistence

Service Interface
SOFEA Principles

1. Application Download, Data Interchange, and Presentation Flow must be decoupled
   - No part of the client should be evoked, generated or templated from the server-side.
2. Presentation Flow is a client-side concern only
3. All communication with the application server should be using services (REST, SOAP, etc)
4. The MVC design pattern belongs in the client, not the server
SOFEA Lifecycle

1. Web Browser ➔ Download Server

   One Time Download
   Presentation Code
   (HTML, JavaScript)

2. Web Browser ➔ Business Logic and Persistence

   Data Only

Darryl Nelson

SATURN 2013
Copyright©(2013) Raytheon Company
MVC in the Web Browser

**MVC Pattern**

- **MODEL**
- **CONTROLLER**
- **VIEW**

**Great for Web Dev**

Web Server → CONTROLLER → VIEW → Web Browser
Benefits of SOFEA

• **Scalability**
  – Server has less work to do; no more presentation generation, just provide a services

• **Higher ROI for each LOC**
  – Expanded opportunity space due to the inherent reusable nature of SOA

• **Better user response**
  – Low latency == happy end users
  – After the app download, no presentation is transported over the wire, only business data

• **Natural fit into SOA and Cloud environments**

• **Organized programming model**
  – Client developers concentrate on the UI
  – Server-side developers concentrate on Services

• **Offline applications**
  – When the network crashes, decoupled client can dynamically switch out their model objects

• **Interoperability**
  – Easier integration with lower overhead from multiple platforms
  – Clients don’t care if services are Java, C#, Python, Cobol or a heterogeneous mix
SOFEA Client Implementation Archetype

Single HTML Page

JavaScript

DOM

HTML-5

Web Workers

MVC

REST Framework (JAX-RS)

HTTP

SPDY

Websockets

JSON

Web Server

Darryl Nelson

Copyright © (2013) Raytheon Company
Lessons Learned

• The web client is a “Priority 1” architecture tier, not an after thought
  – Object-Oriented Analysis and Design principles
  – Design Patterns
  – Continuous integration, performance testing, etc
  – Critical to expend significant engineering time and energy on the client architecture
• Use a mature client-side frameworks
  – Dojo, JQuery, AngularJS, etc
• The RESTful model is natural fit for SOFEA systems
• Architects & developers should “bake-in” asynchronicity between the server and client layers
• Leverage newer technologies where appropriate
  – HTML-5 Web Workers & Websockets
  – Google’s SPDY
• Start cross-browser compatibility testing early in the development cycle
  – Fight the “add IE support later” temptation
• SOFEA excellent choice for our customer’s bandwidth starved environments
  – Very low latency for those customer’s with average-good network pipes
Resources

• **Life Above the Service Tier**
  by Ganesh Prasad, Rajat Taneja and Vikrant Todankar

• **JavaScript Frameworks**
  – Dojo: http://dojotoolkit.org/
  – JQuery: http://jquery.com/
  – AngularJS: http://angularjs.org/
  – KnockoutJS: http://knockoutjs.com/

• **JavaScript Design Patterns Book**

• **SOA & Cloud**

• **Web Sockets**
  – http://www.websocket.org/

• **Google SPDY**
  – http://www.chromium.org/spdy
Darryl Nelson
Chief Scientist
Raytheon Intelligence and Information Services

Darryl.Nelson@Raytheon.com
Darryl.Nelson.Tech@Gmail.com
backup
SOFEA Implementation Examples

• Client
  – JavaScript: Dojo, JQuery, ExtJS, angularjs.org, knockoutjs.com, Twitter Bootstrap
  – Flex*
  – Silverlight*
  – Java Applets*

• Services
  – WS-* (SOAP/WSDL)
    • Axis, Weblogic, Websphere
  – REST 😊
    • Jersey, RESTEasy, RESTlets, Drop Wizard
Processing Request with Push Response Design Pattern

1. Send Request
2. Web App runs asynchronously from the request
3. Handle Request
4. Push Notification & Result Download

Adapted from J. Christensen, Using REST Web-Services and Cloud Computing to Create Next Generation Mobile Applications, ACM 978-60558-768-4/09/10