Normalizing XPages Web Development

Using modern tooling and a separation of data and design to create a new pattern of Domino web development.

Abstract

The XPages runtime is versatile and extends beyond just the components that come out of the box. While the core XPages/Domino platform has not changed materially since it's initial release, the way this platform is being used certainly is evolving rapidly. Learn the best ways to optimize your application development by leveraging the latest and greatest frameworks, libraries, and tools that the web has to offer. Many modern tools can plug into your Domino and XPages applications in a consistent fashion with industry web development practices. Join us for some challenges to our preconceptions, options and alternatives, and a couple of fancy demos.

Table of Contents

- Demo - Goal: Make People Go Blind Through Sheer Awesomeness
- Angular app
  - In NSF
  - Interface
  - Pulling Data from...
- Back-end / RESTful API
  - xe:restService
  - JSON content
- Automation and Tooling
  - Scaffolding / Tooling
  - Distributable vs Source
  - Testing

About Us

Shean McManus

Senior Consultant @ The PSC Group

Love front-end design and development, modern web tech and learning new about new ways to make great applications.

[@sheanpmcmanus](https://twitter.com/sheanpmcmanus)
spmcmanus.net
spmcmanusblog.wordpress.com
+ Skype, LinkedIn, Facebook, Google etc

Eric McCormick

Web Developer @ The Boldt Company

with a passion for open source software, git, well-structured Java applications, Node, build automation, and other tooling to enhance a developer's workflow. In the front-end, I love Angular and vue.js.

[@edm00se](https://twitter.com/edm00se)
edm00se.io
github.com/edm00se

Target Audience

- Developers of Domino/XPages apps with at least a basic understanding of:
  - Notes/Domino API
  - Domino SSJS, Java basics (*can* avoid Java, shouldn't)
  - A desire to be more versatile
  - A desire to learn and grow!

Why this topic?

- Traditional Notes client development is dead
- Traditional Notes web development is dead
- Self-contained xPages development is limiting
- Integration, versatility, flexibility
- Use the best tools for the job
- Extend the life of legacy applications

Overview / Outline

- Re-normalizing / de-specializing XPages development

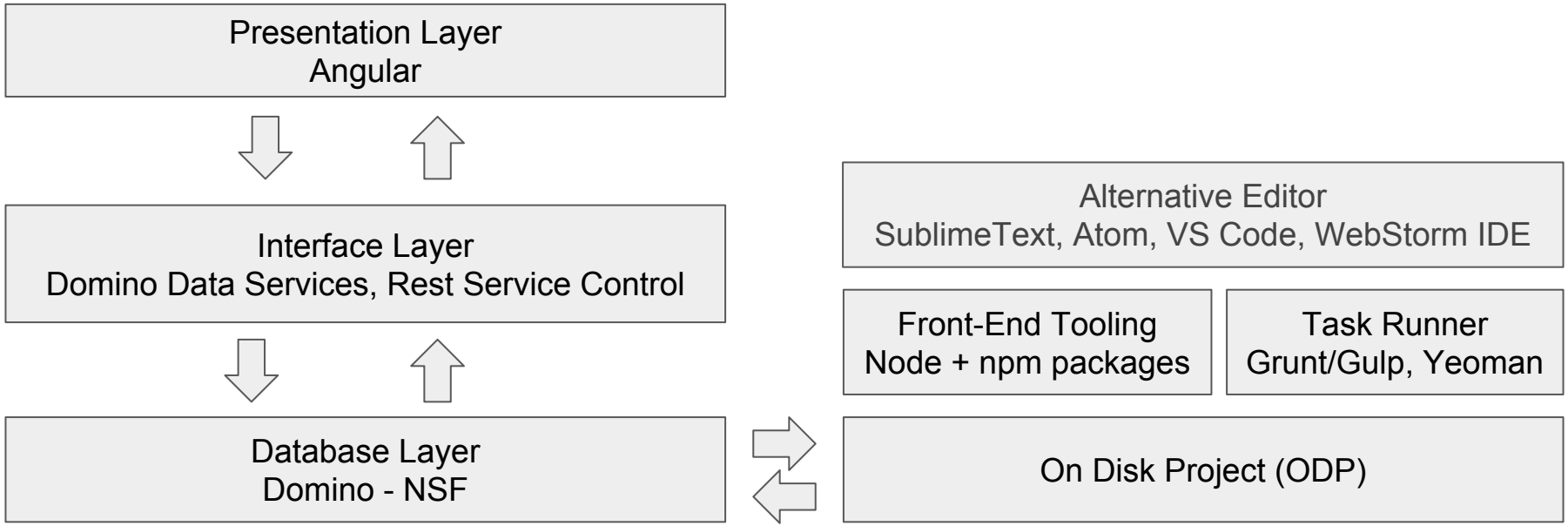
  (note: not to demonize XPages controls, but highlight the versatility of the XPages runtime)

- Focus on the flexibility of XPages as a platform
  - It can support many modes of development
  - Use tools and languages common to other well known modern platforms.

Demos!

Examples to illustrate what we will be talking about in greater detail

Demo Technical Stack



Advantages

- Easier to split apart development concerns
- An API can be consumed by either a client-side app (UI) or another system (properly credentialed, following documented requests)
- Use advanced front-end tooling
- Working on an HTTPServlet *only* requires someone with Java experience and ability to follow the *lotus.domino.** JavaDoc; working on an Angular/Backbone/Ember/vue.js app *only* requires that dev to know those tools
- Your/developer skill set can be more diverse than “just Domino+XPages”
- Front-end vs back-end developers can work on either side of the fence, respectively
- Future-proofs your application (to a degree)
- Future-proofs your development skill set
- Developer ecosystem is far larger, yielding significantly more help/forums/information
- Opening up your toolbox to non-domino tech also provides a far more extensive set of solution possibilities and components/libraries
- Breathes additional life into existing legacy applications without necessarily needing to rework the data model and back end. Or, leverage the domino web server to surface a front end from an existing non-domino data source.

Disadvantages

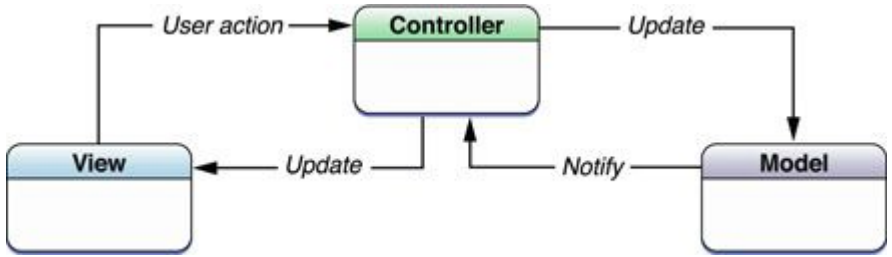
- “New” to many “traditional” Notes/XPages developers (but that margin is closing down)
- Small shops may need devs to be full-stack (I argue that *we already are*)
- Necessitates training, research and sandbox time that may be difficult for some companies to allow.
- Needs to promote a future thinking IT strategy.

MVC Design Pattern

- Developed for desktop software but now recommended for web frameworks
- Clear separation of presentation and application logic

- **M**odel - data and business logic
- **V**iew - presentation of data to the users in any supported format
- **C**ontroller - receives requests and calls necessary resources to carry them out
- Structured code is easier to maintain

- MVC supported frameworks have this basic structure already prepared



REST

- REST
  - REpresentational State Transfer
  - Architecture style or design concept for data communication
  - Used in the building of web services
  - Uses HTTP and it’s GET,POST,PUT,DELETE methods
  - Is stateless (no client context stored on the server between requests)

- RESTful
  - A platform and language independent service is RESTful if it conforms to all of the REST architectural properties.

RESTful Techniques to Use with Domino Data

Many paths can be taken

- LotusScript or Java agents
- view?readviewentries&outputformat=json
- XAgent
- Domino Data Service (exposes full, direct CRUD operations against your database for all your accessible users; likely recommend one of the others!)
- xe:restService
  - via SSJS or Java (can transport anything, but focus is generally the ubiquitous JSON format)
  - *xe:viewJsonService*
  - *CustomServiceBean*
- HTTPServlet(/DesignerFacesServlet)
- OSGi plugin deployed servlet(s)

Domino Data Services (DDS)

- A REST API that accesses databases on Domino servers. It is part of Domino Access Services.
- The Domino Data Service receives requests and sends responses using HTTP and HTTPS protocols with body content in JSON format.
- The Domino Data Service allows you to obtain information on databases, views, folders, and documents. You can update, add, and delete documents.
  Source: IBM Domino Help

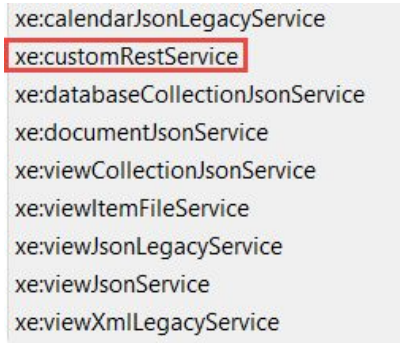
- Enable on Server and Database
- Access via URL: http://{{server}}/{{filename}}/api/data/collections/unid/{{view unid}}?open (for example)
- No coding necessary for the REST service
- Full CRUD operations are exposed to any Author access or above user, so take it into consideration!

XAgent

- Similar to the traditional call of ?OpenAgent
- Call an XPage that is set to not render
- Run your server side code
- More details:
  - Stephen Wissel: http://www.wissel.net/blog/d6plinks/shwl-7mgfbn
  - Chris Toohey: http://www.dominoguru.com/pages/domino_rest_xpages_part1.html

xe:restService

- Standard control to drop onto a page
- Use with many service types



- SSJS or Java
- Custom service bean
- JSON feeds

xe:customRestService

- When called, allows you to run server side code on the following REST calls:
  - DELETE - <xe:this.doDelete><![CDATA[#{javascript:print("doDelete")}]]></xe:this.doDelete>
  - GET - <xe:this.doGet><![CDATA[#{javascript:print("doGet!")}]]></xe:this.doGet>
  - PUT - <xe:this.doPut><![CDATA[#{javascript:print("doPut")}]]></xe:this.doPut>
  - POST - <xe:this.doPost><![CDATA[#{javascript:print("doPost")}]]></xe:this.doPost>

  OR

- Use a service bean (managed JAVA bean)
  - <xe:customRestService contentType="application/json"
    serviceBean="com.my.CustomServiceBean"></xe:customRestService>
  - Maps well to an HTTPServlet (DesignerFacesServlet) based approach

HTTP Servlet

- Fairly close to what other JEE developers would make with a (true) javax.servlet.http.HttpServlet
- Technically, the implementation here is a DesignerFacesServlet
  - Gives us access to FacesContext
  - Which gives us access to user's Session
- All app code is contained within the NSF
- You can (optionally) use something like the Google GSON JAR to handle:
  - Conversion of data (Java object) to JSON for response
  - Reflecting between JSON string and Java object
    - Any POJO
    - Map
  - Many non-XPages/Domino developers are quite familiar with GSON

HTTP Servlet Setup

Requires a little setup, but easy once established:

- Edit of java.pol(icy) to grant permissions (ClassLoader related)
- Addition of allowed methods PUT and DELETE (via Internet site or notes.ini value of HTTPEnableMethods=PUT,DELETE)
- Inclusion of the lwpd.domino.adapter.jar JAR in your project build path (it's already there, just not included by default)
- A specific file (META-INF/services/com.ibm.xsp.adapter.servletFactory) to connect the NSF to your endpoints
- Endpoints are established in your ServletFactory (com.ibm.designer.runtime.domino.adapter.IServletFactory)

OSGi Plugin

The “JEE way”, it requires a working ability to create and deploy an OSGi plugin / update site.

- Lots of people have shown off the working pieces of this process
  - Toby Samples has a 4-part JAX-RS series, as he calls it THE way to do REST on Domino
  - Paul Withers, as part of his From XPages to Web App series, including Vaadin and CrossWorlds
  - Jesse Gallagher, who goes into more of the related topics, in his “That Java Thing” series
- Java skills are far more portable beyond any single platform, so a little learning can go a long way to benefit your apps today and those of tomorrow

Intro to Front End MVC Frameworks

- What is a framework?
  - An end-to-end solution that controls the coordination and sequencing of application activities.

- Delivery of assets
  - Local copies
  - CDN - Content Delivery Network

Popular Front-End (MV*) Frameworks

- Angular
  - Open-source, maintained by Google
  - Supports MVC and MVVM architectures
  - Dynamic page content through two-way data binding
    - automatic synchronization of data between model and view components
  - Examples: YouTube on PS3, MSNBC.com, Plunker, Weather Channel
- Backbone
  - RESTful JSON interface
  - MVP architecture
  - Lightweight - only dependency is underscore.js
  - Examples: Airbnb, USA Today, Hulu, Pinterest, Digg
- Ember
  - Open-source, MVCs, two-way data binding
  - Discourse, Vine, Live Nation, Nordstrom, Chipotle
- Knockout, KendoUI and Others

Task Runners: Overview

Provide automated tasks, consistently, with every build. Tasks require some minimal configuration for automated gain in the realm of:

- Optimized assets, with tasks/sub-tasks of:
- Consistency between developers (enforced JS code formatting)
- Builds as trivial operations (for on-demand, or deployment via CI/CD)
- Can add pre-processing to a development workflow with minimal effort
    - TypeScript or CoffeeScript
    - SASS or LESS
- Tests as standard (or test-driven development)

Task Runners: Our Demo

Our optimized demo incorporates:

- An AngularJS + Bootstrap-ified front-end app
- An application scaffolded out via Yeoman and generator-angular (installed from npm)
- Dependent front-end libraries managed/installed via Bower (our demo includes this install as part of `npm install`)
- Configure once, use always… automated!

- Build optimization of CSS, JS, and HTML partials
- JS Linting (formatting, error checking)
- Image optimization
- Reduction of network calls for page load:
  - 1x vendor CSS file
  - 1x vendor JS file
  - 1x app CSS
  - 1x app JS
  - HTML partials templated into the app JS
- JS test scripts

Installing Our Demo App

Our optimized demo incorporates:

- Clone the repository `git clone https://edm00se@bitbucket.org/edm00se/beer-debt-mk2.git`
- Install dependencies from npm `npm install` + `bower install`
- Perform the build `grunt`
- Import the On Disk Project (ODP) in Designer's Package Explorer (or Navigator)
- Right-click the ODP and Associate with New/Existing NSF
- Done!*

*Note: Domino Data Services will not work correctly from local web preview

Task Runners: Our Demo App Workflow

Steps for a normal workflow with our setup:

- Create new back-end code, if needed (new service, RESTful endpoint, backing logic change)
- Create/Update front-end code/logic in editor of choice (SublimeText, WebStorm, Atom, VS Code, etc.)
    - Add a route with html partial, js controller, and js test script all added automagically w/ generator-angular (plugs into existing app and routing definition, etc.) by `yo angular:route <myNewRouteName>`
- Perform new build to update the ODP's copy of your front-end app, via `grunt`
- Wait for DDE to import and sync the changes to your NSF
- Rejoice!

* Alternatively, you can use the command `grunt serve` to keep a watch task on your source files, triggering a new build automatically when it detects a file save event (which will, with DDE preferences set correctly, automatically bring them into your NSF).

Summary

- Separating Domino data from the design allows developers to make use of MVC frameworks for the client side development.
- REST architecture is a modern standard for data interfaces between client and server
- Modern tooling allows for automating application builds, deployments and testing.
- Domino is a versatile web development platform capable of supporting a wide range of development patterns, tools and techniques.