A Modernized Developer's Workflow with Domino/XPages

Eric McCormick

**MWLUG 2014**

Audience

Who Am I?

- developer
- husband
- father
- veteran
- former EMT
- blogger
- ...more than I have space for

IBM**CHAMPION**

[Eric McCormick .io](#)

[@edm00se](#)

blog: [edm00se.io](#)

Who Am I?

Web Developer at   **BOLDT.** | BEYOND **125** YEARS

– a fourth-generation, family-owned professional
construction services firm, in operation since 1889

my focus usually revolves around:

– document control

– productivity and performance metrics

– system integration

– process automation

Agenda

- What and Why
- How? (in the App)
  - Split front-end and back-end (RESTful APIs)
    - HttpServlet
    - OSGi plugin (JAX-RS)
    - XPage w/ xe:customRestService (Java or SSJS)
    - XAgent
  - Back-end primary logic
  - Front-end heavy app
- Observations of a corporate developer
- How? (Outside the App)
- Demos

What and Why?

- Modern web apps
  - using all the fancy new toys
- existing Domino server
  - XPages runtime
    - all the Java server stack power
  - XPages components
  - user / security model
- performance optimizations
  - lots of development tooling
- eases concerns of:
  - testing
  - documentation
  - deployment
- eases sanity

App Concerns: Splitting the App Logic

where does the application logic lie?
- business logic
- workflow logic

server-side logic:
primary logic
- enforceable rules
- notifications
- (additional) server events

client-side logic:
- how the "form" operates
- specifics to UI/UX, or the device (e.g.- camera, barcode scanner)

App Concerns: Back-End

- "best way" of establishing RESTful (or other) APIs:
  - OSGi plugin driven
  - NSF driven
    - HTTPServlets (rather, extending DesignerFacesServlet)
    - xe:restService
      - xe:customServiceBean
      - other (even Domino's SSJS)
    - XAgent
- adding tooling:
  - OSGi plugins mean we can use "real" Eclipse for development
    - can use things like JAX-RS (which the rest of the JEE world uses!)
  - for those willing/abel to make an edit to the Domino servers java.pol(icy) file:
    - GSON
    - otherwise the com.ibm.commons.util.io.json way

App Concerns: Front-End Concepts

- you can use all the fancy toys
- you "deploy" your built app to your ODP's WebContent/ path
- properly structured, you can use back-end mocks for DDE-less preview and edits
- some tooling (angular-cli, webpack, etc.) automates this

App Concerns: Front-End Implementation

- use all the things!!1
- build for deployment
- profit

caveats:
- know what you're doing
- **don't obliterate <u>WebContent/WEB-INF/</u>**

App Concerns: Front-End Deployment

Files changed (1)

`+4` `−1` **package.json**


```
      package.json          Coverage error   Side-by-side diff   View file   Comment   ...

   ...

 8   8        "lint": "tslint \"src/**/*.ts\"",
 9   9        "test": "ng test",
10  10        "pree2e": "webdriver-manager update",
11       −    "e2e": "protractor"
    11   +    "e2e": "protractor",
    12   +    "clean": "del 'ODP/WebContent/*' '!ODP/WebContent/WEB-INF'",
    13   +    "build": "npm run clean && ng build —prod —o ODP/WebContent/"
12  14      },
13  15      "private": true,
14  16      "dependencies": {

   ...

21  23        "@angular/platform-browser-dynamic": "2.0.0-rc.5",
22  24        "@angular/router": "3.0.0-rc.1",
23  25        "core-js": "^2.4.0",
    26   +    "del-cli": "^0.2.0",
24  27        "moment": "^2.14.1",
25  28        "ng2-bootstrap": "^1.0.24",
26  29        "reflect-metadata": "0.1.3",

   ...
```

Observations of This Corporate Developer

- scm for everything
- if you spend on a task, automate it
  - to the extent available
- build scripts should be:
  - minimal effort (one-line commands are easy)
  - configured as non-invasively as possible
- "anyone" should be able to pull the latest:
  - code
  - build
  - submit a feature branch / PR
- CI / CD considerations:
  - headless DDE builds
  - more?

Why a "Developer's Workflow?"

- Automation is King
- Focus on the task at hand
- If the steps are reproducible, they should be automated/ scripted/etc.
- You can waste your time configuring and setting up
- Or you can get to work

http://www.commitstrip.com/en/2015/06/22/can-we-automate-everything/
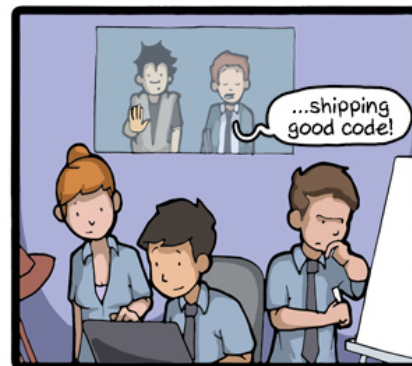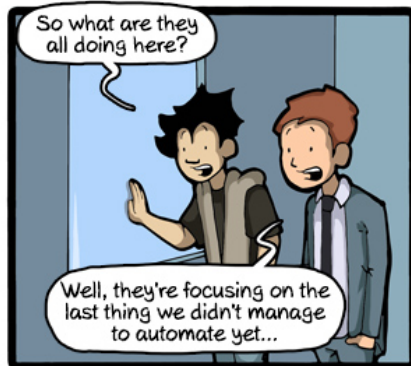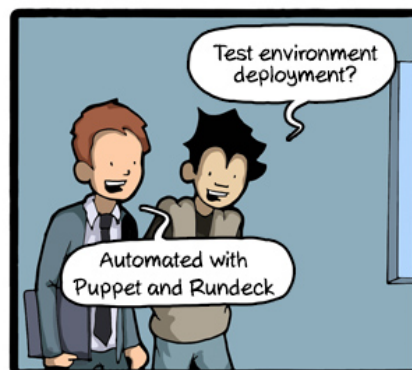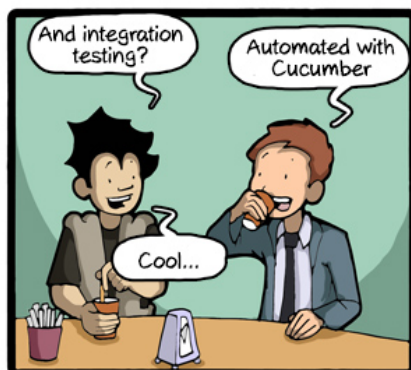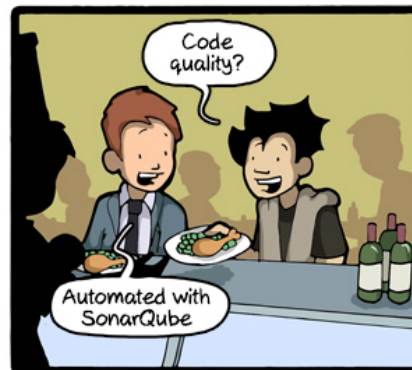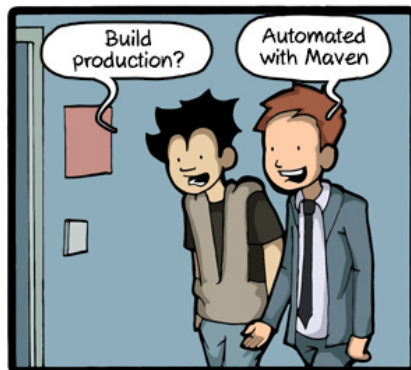
Beyond the App

- GitLab has CI capabilities baked in
- running GitLab in a docker container makes upgrades amazingly simple
- installing the GitLab CI runner on Windows easy
  - much easier than the setup required for setting up Jenkins for headless DDE builds (a la Cameron Gregor)
  - can set PowerShell as the default shell environment
  - the runner is registered w/ the main GitLab env, can "tag" (e.g.- Windows, DDE, PS)
  - can limit # of simultaneous executions (aka- only one DDE build at a time)
- the possibilities are endless

Demos

Choose Your Own Adventure!

Back-End

- [generator-xsp](#)
- [HTTPServlets](#), [xe:restService](#)

Front-End

- [generator-webapp](#)
- [generator-angular](#)
- [TypeScript](#) setup (simplified, DIY)
- [angular-cli](#) (Angular 2)
- [vue-cli](#) ([templates](#), w/ [browserify](#) or [webpack](#))

Scaffolding an App's ODP (generator-xsp)

Scaffolding an App's Front-End 1/2

Scaffolding an App's Front-End 2/2

- change the dev/build scripts to:
  - output to ODP/WebContent/
  - not obliterate your ODP/WebContent/WEB-INF/

Q&A

Coming to my blog soon…

- more on generator-xsp
- extending beyond my task runner series
  - to include use of webpack w/ front-end apps on Domino/XPages

Thank You!