Rapid Application Development of Single Page Applications (SPAs) with ColdFusion & JavaScript

intercooler.js + Cf

CFSummit 2019
Mark Takata
Founder
Takata Technology Consulting / Pretty Good AI / AIMLAR.com

What we're going to learn

- What is RAD (Rapid Application Development) methodology?
- Setting up RAD workflows
- Why is CF an ideal RAD platform?
- Tools to facilitate RAD methods
- When to document
- Why SPAs? (Single Page Applications)
- An introduction to Intercooler.js
- Code samples
- Resources
- Questions

<cfinclude template=”introduction.cfm”>

- Full stack web developer & Business Analyst
- Started programming in ColdFusion in 1998
- Founder & Developer, Takata Technology Consulting, Pretty Good AI, AIMLAR.com, on contract with UC Davis Financial Aid.
- Background in C#, CF, JS, PHP, SQL and other letters of the alphabet
- West Sacramento, CA

  (which is west of Sacramento)

What is RAD?

What is Rapid Application Development?

- A type of Agile methodology
- Also known as “rapid application building”
- Less emphasis on planning, more on adaptive process
- Focus on prototyping over specification
- Feedback, iterative driven
- Designed to “fail fast” & refactor
- There are specific & general versions (we’re focusing on general)

Benefits of Rapid Application Development

- Improved code quality
- Less risk of failure
- Ease of hitting deadlines & budgets
- Products closer to client requirements
- Higher amount of open communication

Potential Drawbacks of Rapid Application Development

- Uncomfortable, unfamiliar approach
- CTRL-A Delete is tough for some devs
- Missing “non-interaction” requirements
- Requires active stakeholder participation
- “Hack & Test” can lead to poor overall architecture
- Focus on micro-transactional design can lead to poor scaling

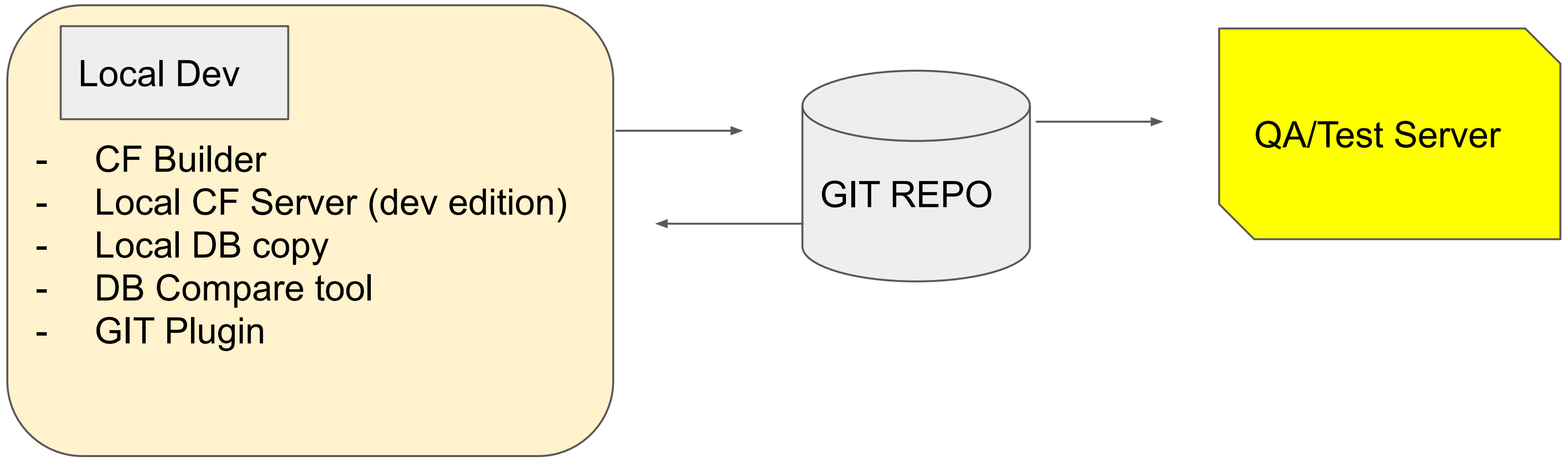
Best Fits For Rapid Application Development

- Small to medium projects
- Experienced developers (or juniors w/ strong mentorship/support)
- User interaction heavy applications
- Non life-critical systems

RAD Workflows

- Minimize “time to screen”, both locally & to your testers
  - Less steps. Less output options. Less potential for confusion.
- Use software management systems strategically
- CI can be your friend, if it is set up well
  - Speed is critical. Consider “round trip” time on changes
- Code, test, preview locally
- Comfort is paramount, but don’t forget safety
  - “Source of truth” needs to be in a repository, not local

RAD Workflows - Example



Coldfusion - A RAD Framework

Why is CF such a good RAD Framework?

- Designed for speed
  - Black box'y structures
  - Shortcuts
- Wide variety of coding methodologies
- JIT compilation speeds time-to-render
- Ability to use tag based CF lowers number of total lines
- Extremely good at building & exposing APIs

Tools for RAD Methodology

- CFBuilder
  - GIT plugin
  - FTP plugin (your mileage may vary)
  - Remote database access
- VS Code
- GIT Repository (Github, Azure Devops, etc)
  - VERY important for looking back
  - Commit often
- Dreamweaver
  - FTP, Git capability
- OneNote, Confluence or other documentation system

When to document

- Generally functional code > detailed documentation during RAD

BUT...

- If you overcome a major complication/hurdle, document it
    - Security, synch or timeout issues, etc
- Timesavers, “aha” moments
- Large exceptions to normal workflow (i.e. to solve a problem)

Once project moves to “endgame” or out of prototype phase, these can be used to help with requirement/functional specs for production and beyond.

Wait for laughter

Why Single Page Applications? (SPAs)

- Less page scaffolding
- When prototyping, the main issue with SPAs (state) is less important
- You can move from a SPA to a traditional site after prototyping, especially if you change JS frameworks
- User experience can be great (no page reloads)
- A lot of data can be loaded at once, so once cached it is fast

Why Not Single Page Applications? (SPAs)

- State management is not fun
  - Bookmarks, refreshes/back can be a bit of a thing to handle
- Not applicable for all site types, ideal for “tool” applications
- If you’re not using something like Intercooler to “load as needed”, initial application load can be a bear
- Not a common pattern for devs, so might take time to learn

What is Intercooler.JS

- Declarative JavaScript Framework
- Uses web standards such as AJAX, CSS, REST and JavaScript
- Can be added incrementally to existing architectures
- Can often require little to no actual JavaScript coding
- Works with any back-end technology
- Allow for Rapid Application Development
- Created by Sacramento legend Carson Gross
  (@carson_gross on the Twitters)
- Download & documentation at http://intercoolerjs.org/

Why Intercooler.JS

- Integrates well with Coldfusion
- Simple but powerful abilities
- Flexible, “use when you want”
- Very fast to integrate
- Framework is mature enough to use in production (used by UC ANR in Davis, CA)

How Intercooler.JS Works

- Overloads an object’s verb
- Replaces contents of the object with contents returned from remote

SO:
<a ic-post-to="/button_click">Click Me!</a> would replace “Click Me!” with whatever returned from /button_click/

Neat, right? But maybe not that useful...

How Intercooler.JS (cont)

You can also trigger from one object, and “target” another:
<div id=”ic__viewport”></div> ← Target

<button ic-get-from="/main_view/" ic-target=”ic__viewport”>Main</button>
<button ic-get-from="/about_view/" ic-target=”ic__viewport”>About</button>
<button ic-get-from="/contact_view/" ic-target=”ic__viewport”>Main</button>

WHOA! An SPA! Just like that...

Code Samples

Don’t mess up, Mark...

Resources

Intercooler: http://intercoolerjs.org/

Mark’s Twitter: @TheFatPanther

Github Repo: https://github.com/MarkRTakata/CFSummit2019

Mark’s Email: techfiend@gmail.com

Questions

(That is, if everyone isn’t just sitting there playing Poochmatch)