Overview of the Fluid Project

Colin Clark
Inclusive Software Architect,
Adaptive Technology Resource Centre

Responding to Challenges

- Responding to the need to improve UI in community source projects
  - Accessibilty, usability, internationalization, customization, individual needs
- Create both technologies and processes
- Enable skilled design contributions
- Share user interface components
- User interfaces that can flex to accommodate institutional and individual needs

How?

- Create common skinning tools
- Build technologies to support flexible, reusable UI components
- Create exemplary UI components
- Share and adapt components across applications
- Support the design and testing process

Why is this Architecture Different?

- Provides a consistent model for UI components across applications
- Establishes a single API for configuring components
- Provides a consistent way of specifying site-wide customizations such as skins
- Decouples UI from application logic
- Enables easy switching of components to meet diverse user needs

Technical Challenges

- Extremely diverse range of presentation technologies: Java, XSLT, and PHP
- Bridging the platform and language gap
- Enable rich and accessible experience
- Support incremental adoption

What is a UI Component?

- A reusable bundle of UI real estate:
  - HTML markup or template
  - Controller logic
  - Metadata describing the role and states of the component
  - Configurable properties (“bindings”)
- Can be composed of other components
- In our framework, a component is a DHTML widget managed by a JavaScript & AJAX container

Core Architecture

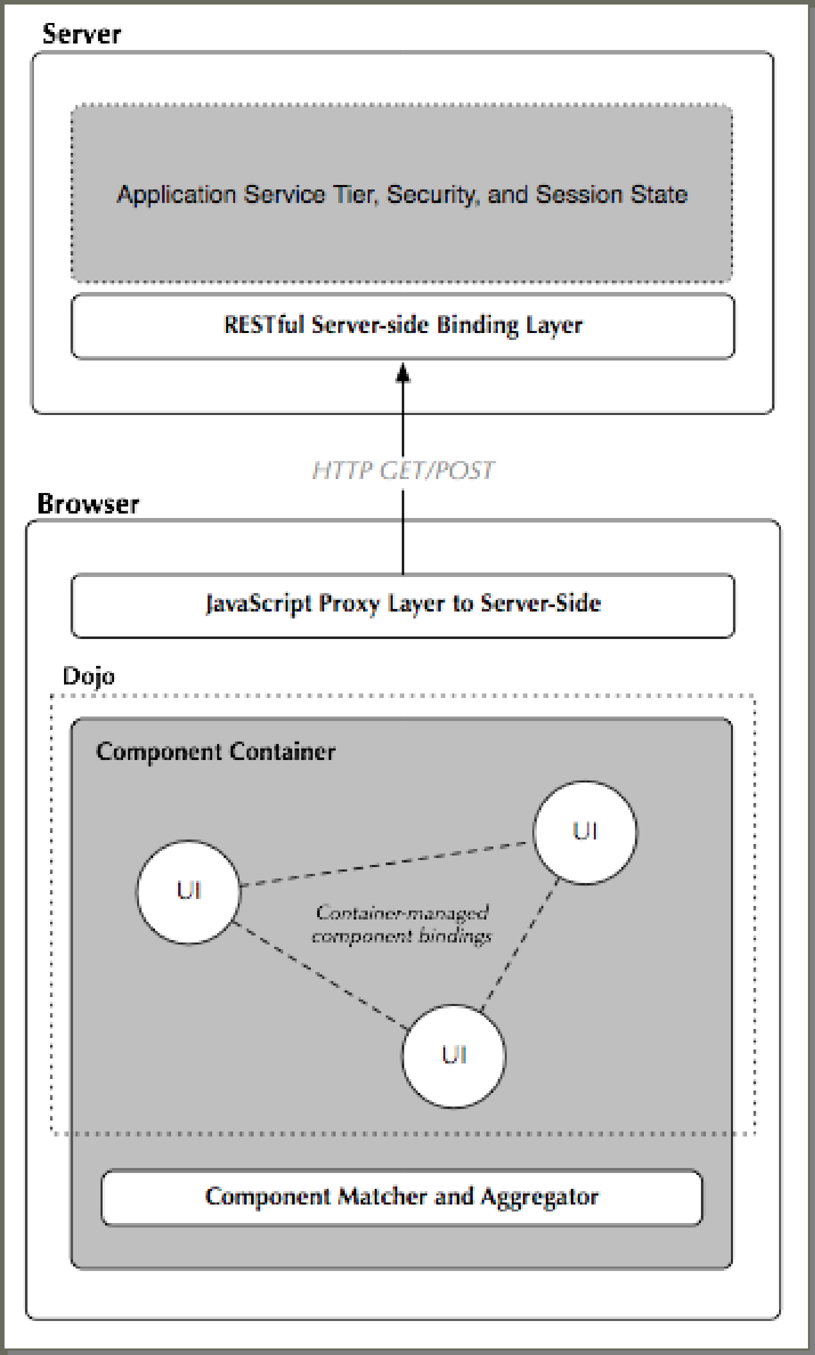
- Cross-application skinning system
- Personalized run-time styling
- Component framework
- Repository of shared components
- Semantics and specifications
- Integration

Skinning System

- General way to create skins that work across applications
- Customization and branding at configuration time
- Extensible
- Doesn’t require the Component Framework

Component Framework

- Component model and APIs
  - JavaScript, CSS, HTML
- Component container
  - JavaScript, AJAX toolkit integration
- Server-side binding layer
  - REST-based specification + implementation
- Runtime Transformation Engine


Server
Application Service Tier, Security, and Session State
RESTful Server-side Binding Layer
HTTP GET/POST
Browser
JavaScript Proxy Layer to Server-Side
Dojo
Component Container
UI
UI
Container-managed component bindings
UI
Component Matcher and Aggregator

Component Repository

- Means for sharing components publicly
- Source of high-quality, reusable components
- Web-based, RESTful
- Secure, credible, and well-maintained

Integration

- Early and often
- Testing harness at first, project integration as soon as possible
- Requires regular collaboration with partner projects
- Litmus test of project usefulness

What’s Next?

- Proposal due mid-December
  - Goal: define a focused scope based on clear resources
- Involve the uPortal community:
  - Background documentation
  - More conversations with developers and UE people