Prototyping a Future UI

OBSOLETE: This page is left here as a reference for the time being, but it has been replaced/superseded by the new DSpace UI Prototype Challenge.

UI Platform Needs

The following is a list of features/needs/use cases which we feel would make a good User Interface / User Interface framework. Since not all of these features/needs would have the same importance, we've categorized some as "required", "recommended" or "optional".

Any prototyped UI platform must keep in mind these needs. Prototypes will be judged based on how well they can achieve these needs overall

1. **Open Source (*required*):** Obviously. Also we need to avoid GPL and similar licenses which are incompatible with BSD license in use by DSpace.
2. **Complements DSpace's "just works" goals (*required*):** in keeping with Goal 5 of the Strategic Plan for Technology (*DSpace will support low-cost, hosted solutions and deployments (by featuring an easy, "just works" setup)*), a UI framework should in no way hinder the ability to easily install/setup DSpace. This means that if a UI framework has its own dependencies or software prerequisites they should be easy to bundle/install (at least no more difficult than the current DSpace installation process, and ideally easier).
3. **Complements the Roadmap *(required)***: A UI framework should be analyzed in how it may achieve other higher priority items on the RoadMap. More specifically, a UI framework should not make it more difficult to achieve specific, high priority Roadmap UI-level features. Some examples include: the ability to move configurations into the Admin UI, and manage themes in the Admin UI.
4. **Search Engine Optimization (SEO) friendy (*required*):** A UI platform must allow for indexing by search engines (e.g. Google, Google Scholar and similar). This means that UI frameworks which build the majority of the page using client-side javascript would be unacceptable (as Google would be unable to index the contents). But, frameworks using server side technologies (even server side javascript) would be acceptable, as search engines would be able to easily index the final HTML.
5. **Ease of Branding/Theming (*required*):**A User Interface should be easy for institutions of all sizes to brand or theme. This means even smaller institutions (without a full time DSpace developer) should be able to theme or brand DSpace with some amount of ease. At a minimum, things like the header/footer/color scheme and basic layout should be simple to modify or customize. Ideally, the UI would support third-party themes (e.g. Bootstrap themes from http://bootswatch.com/ or similar) which can be easily applied to the UI to change its entire look and feel.
    a. **DCAT feedback 2015-03-10:** We see it as a substantial feature/requirement to be able to apply **different** themes to different sections of DSpace (collections, communities). It would be great if "some amount of ease" would be more tightly defined as: Configurable within the user interface itself and does not require a rebuild or restart of the system, especially when we're talking about basic theme config changes.
    b. In addition to general "look and feel" theming, it should be possible to easily configure basic functionality, such as the order and selection of metadata fields to display on simple and full item pages. Such customisation belongs at a configuration level, and does not need to be intermingled with design concerns.
6. **Responsive Web Design (*required*) :** a UI should be responsive and mobile-friendly, adapting to the size of various devices.
    a. Bootstrap support (*recommended*): Ideally, the UI would support Bootstrap, since it is one of the most widely used and supported responsive frameworks
7. **HTML5 Support (*required*):** a UI should be able to support HTML5. Ideally, it is built primarily with HTML5 in mind, rather than only supporting some aspects of HTML5.
8. **Separation of Concerns (*required*):** a UI should be built with the idea of "separation of concerns". For example, the UI framework should include NO business logic or Database query logic, etc. It should also have no knowledge of the underlying storage framework (e.g. Database schemas, file storage locations, etc). A UI should communicate with DSpace through one of two means: (1) via the DSpace Java API (but any missing business logic may need to be added to that Java API), OR (2) via the REST API (and other similar layers, e.g. Solr). It would NEVER communicate directly with the database or other underlying storage layers.
9. **Standard way of dealing with Internationalization (i18n) or translations (*required*)**: DSpace has multiple international language communities who each manage their own set of translations for the interfaces. Migration from the current way of managing these translations to the new framework should be possible. Contribution of new translations should not be more difficult than it is today.
10. **Rapid Development support / Developer friendly (*highly recommended*):** a UI should be easy to develop against and improve upon. Ideally in a popular technology or language. Local developers should not need to go through extensive training to work with the UI.
11. **Large, active development community *(highly recommended)***: a UI framework should display promise for long term sustainability. Does it have an active, diverse development community (with ongoing releases and a roadmap)? Does it provide good documentation and online training resources? Is there a third-party development ecosystem (e.g. tips/tools/modules from third parties) or is the framework controlled by only one or two organizations?
12. **Faceted/Filtered Search/Browse friendly (*highly recommended*):** a UI should easily integrate with a faceted/filtered search engine/server (such as Solr or Elastic Search) or a generic API which can communicate with said faceted/filtered search engine (e.g. Discovery, Blacklight).
13. **Java-friendly (*recommended*):** DSpace's underlying framework & API is Java, and will remain Java. There are *no plans* to completely rewrite DSpace. However, this does NOT mean the UI needs to also be written in Java (especially if the UI primarily communicates via REST API or similar). However, it may be best that the UI technology is "Java-friendly" or in a language that is similar to or based off Java (e.g. Javascript, Groovy, even Ruby is similar enough). This would ensure existing developer resources could easily switch between Java (at the API level) and the UI language.
14. **Flexible URL Structure (*recommended*)**: It may be too limiting to work with a UI framework that imposes a very specific, limiting URL path structure. We also do need to anticipate that, if a UI platform necessitates a new URL path for objects (Communities, Collections, Items), we will need to find a way to auto-redirect existing URL paths to the new location (ideally within the UI platform itself, and not requiring manually adding many Apache or Tomcat level redirects).

UI Platform Prototype Requirements

A single prototype should be achievable within a 2-4 weeks of active development (ideally closer to 2 weeks). We've tried to scope these requirements to a bare minimum so that you need not sink a lot of development time into a prototype.
Prototypes should be considered "throw-away" experiments/proof-of-concepts. There is no guarrantee any prototype code will be used.

Keep in mind that all prototypes should be considered potentially "throw-away" code. We do not recommend trying to build all the functionality of DSpace, as it would be a waste of your time if it is not the chosen platform. The goal of these prototypes is merely to provide a "proof of concept" of how the platform could be used, and to get enough familiarity with the platform to determine how well it may (or may not) be able to achieve other features on the RoadMap.
Be willing to abandon a prototype if it's not going well. Just give us feedback on what issues you encountered

Because these prototypes should be considered "throw-away", we do not recommend working longer than necessary to provide a "proof of concept" of the UI platform. Therefore, if you hit a major roadblock on a platform, you are more than welcome to abandon it and simply describe the issues you encountered with that platform.

Functional in the UI Prototype

A prototype only needs to implement "required" functionality. If it helps in analyzing the platform, prototypes may add additional functionality. But, keep in mind that the number of features implemented will in no way affect the final UI decision.

1. **Browse Content (*required*) :** MUST be able to display the content (Communities, Collections, Items) of an existing DSpace database (i.e. You may assume the database is pre-filled with content already).
2. **Authentication (*required*)**: MUST allow for someone to successfully authenticate into the system (using a *single* authentication method, of your choice). They should be able to see their name appear as having successfully authenticated, but need not be given extra permissions.
    a. While you should NOT build in additional authentication methods, you must describe how you would do so in this UI platform
3. **Searching Content (*optional*):** Not required to be implemented, so this could just be described in words. How would this platform achieve searching content in DSpace (via Discovery or Solr directly)?
4. **Basic Editing (*optional*)**: Not required to be implemented, so this could just be described in words. How would you achieve editing of metadata or files within this framework?

Describe how you would achieve

For the UI platform to be fully analyzed, we ask that you describe how you *would achieve* the following RoadMap features in this UI platform. (You need not actually achieve any of these features in the prototype itself, just describe how you think they might be achieved by developers in the future.)

1. **Authorization:** Would this UI platform be able to use the existing DSpace authorization scheme, or is there other means for AuthZ in this platform?
2. **Internationalization:** How does this UI platform support/handle Internationalization?
3. **Theming:** How does this UI platform support/handle theming? Is it possible to theme portions of the site (e.g. individual Communities/Collections) differently? Is there any mechanism for selecting between multiple themes (e.g. shipping DSpace will multiple themes, but selecting the one you'd like to use)? Is there any mechanism for dynamically changing a theme (even just header/footer/color scheme)? See also "Theme management in Admin UI" feature in RoadMap.
4. **Configurations in the Admin UI:** Does this UI platform support the idea of dynamic configuration (without requiring a Tomcat restart/reload or similar)? If so, any ideas/resources for how to achieve this within this platform (e.g. would configs need to be added to the database, or is there another mechanism for dynamically loading/changing configurations)? See also "Configurations in Admin User Interface" feature in RoadMap
5. **Configurable Deposit Process:** How would one reconfigure deposit screens in this UI platform? Would it be easy to tweak the UI templates to add new fields into the deposit process? Any ideas for how difficult adding custom metadata fields or reordering "steps" would be? What about enabling/disabling Creative Commons licensing and/or Embargo functionality? (Keep in mind we are NOT requiring a new UI to use the existing Configurable Submission APIs/configs to do this, but that is still an option available. So, please describe whether or not you'd use the existing Configurable Submission APIs, and, if so, how you think that would work in this UI platform.) See also Submission User Interface documentation.
6. **Configurable Workflow Process:** Similar to the previous question, how would you anticipate allowing for tweaks to the Workflow Approval process? Would this Platform just use the existing APIs in DSpace for configuring this, or is there any other means available? See also Configurable Workflow documentation, as well as the "Single Approval Workflow system" and "Enhance Approval Workflow capabilities" in RoadMap

UI Platform Prototyping Projects

Any UI platforms/frameworks are fair game, but keep in mind that they will be judged based on the "Platform Needs" and "Prototype Requirements" listed above.

UI Platform	Open Source License (s)	Technology Resources	Prototype Links / Notes	Prototyping Team / Volunteers
What UI platform /framework are you prototyping?	*What OS license(s) does it use?*	*Add links to resources /documentation for any technologies that this prototype uses*	*Link to either a public wiki page describing your prototype, or to a public README in GitHub. Keep in mind, your codebase **must** also be public from the beginning of the prototype.*	*List your team members*

Spring Boot + Thymeleaf (Note: *Currently a rough experiment / proof of concept*)	• Apache 2 (Spring Boot, Thymeleaf) • MIT (Bootstrap)	• Spring Boot (UI framework) • Thymeleaf template engine (builds the HTML) • Bootstrap (for CSS styling /theming)	• Codebase: https://github.com/tdonohue/DSpace/tree/spring-boot-ui/dspace-ui • NOTE: This is a very rough experiment at this point (not a full prototype yet). Its purpose is just to "prove" that Spring Boot can be used to build a UI on top of the existing DSpace API. The Thymeleaf template engine could easily be swapped for any other template engine supported by Spring Boot.	TBD
REST + Ruby on Rails		• DSpace REST • http://rubyonrails.org/	https://github.com/peterdietz/dspace-rest-rails http://dspace-rails.herokuapp.com/	Peter Dietz
REST + Play!		• DSpace REST • https://www.playframework.com/	https://github.com/peterdietz/dspace-rest-play http://dspace-rest-client-play.herokuapp.com/	Peter Dietz