Recipes 2.0: Building for Today and Tomorrow

Rion Dooley
Texas Advanced Computing Center
The University of Texas at Austin
Austin, US

Matthew R. Hanlon
Texas Advanced Computing Center
The University of Texas at Austin
Austin, US

***Abstract*—The history of science gateway development has, in many ways, been a story of the "Haves" vs. the "Have-nots." Large infrastructure projects led the way, building thick client portals to provide coherent interfaces to an incoherent environment. Contrast this with the way the modern web is designed using light, front end components and outsourcing much of the heavy lifting to a mash-up of REST APIs, and it is easy to see why modern web applications can be prototyped and refined into stable products in the time it previously took thick client portals to do an initial release. This paper argues that a "build for today" philosophy can lead to the rapid development of science gateways to serve the "Have-nots." Also presented is a set of responsive front end components built on top of the iPlant Foundation API that provide the boilerplate for rapid development of lightweight science gateways using only HTML, JavaScript, and CSS. Using these components, developers can easily stand up new gateways or quickly add new functionality to existing ones.**

Keywords— Science Gateway, REST, API, web service, AGAVE, HTML5, JavaScript, web

I. INTRODUCTION

The history of science gateway development has, in many ways, been a story of the "Haves" versus the "Have-nots." Large infrastructure projects led the way, building thick client portals to piece together incongruent service stacks and provide cohesion to an incoherent environment. The field was dominated so thoroughly by these heavyweight portals, that the terms portal and gateway became interchangeable. A gateway was no longer just a means of access, it was an ecosystem of moving parts that all had to be managed and maintained over time for the gateway to work. The concept of modular design became a relative term. If one could take a component out of one monolithic framework instance and add it to another monolithic framework instance, then the component was modular. Disregard the background processes, supporting services, and database that needed replication in order for the module to work. If the UI could be reused, the component was considered modular.

The resources required to build and maintain such portals made finding portals with long-term success rare. Whereas at one time portals were built as thick desktop clients, one of the reasons that portals gravitated from the desktop to the web was the ongoing cost of maintaining software on multiple operating systems. Even applications written entirely in Java require some platform-specific attention. That means multiple sets of unit tests, multiple testing environments, and most importantly, multiple investments of time writing system-specific code when the web only requires a single investment.

Despite the cost and complexity, projects that could afford to make the investment in a portal did so gladly because the end product was well worth the cost. Portals brought cohesion to complicated infrastructure and made computational science accessible to researchers without computer science degrees. They pulled the focus away from the machines and put it back onto the science.

Portals such as Cipres [1], GridChem [2], UltraScan [3], Galaxy [4], and NanoHub [5], just to name a few examples in the United States, continue to provide tremendous value to their user communities. The researchers using these portals have made discoveries leading to hundreds of published papers, multiple thesis and dissertations, and insights that would have taken significantly longer to realize, if at all. One cannot deny the value of such portals in today's scientific process.

The challenge portal-driven science faces is that for every scientist that has a portal like Cipres at their disposal there are hundreds more in the same domain who do not. No portal can meet the needs of everyone. Successful portals find their niche and focus on providing value to the researchers in that niche. Inherent in the design of a successful portal is the realization that it cannot and will not meet the needs of the vast majority of scientists who could otherwise derive value from similar tools. Thus, even within the highly technical research landscape there is a digital divide [6] between those who have advanced portal technology to facilitate their work and those who do not.

Exact numbers are difficult to obtain, but a rough approximation is possible. The National Science Foundation's Science and Engineering Indicator Report for 2012 (SEI) states that as of 2010, the US Science and Engineering (S&E) workforce is 6.65 million people [7]. Of them, 31% describe research and development (R&D) as a major work activity. If we consider only those with doctorates, 12% of those who describe R&D as a major work activity remain. This indicates that there are at least 247,000 PhD level workers in S&E actively conducting research in the US. Add to this the estimated 100,000 medical researchers in the US according to the Bureau of Labor and Statistics and we come to a lower bound of 347,000 for the number of researchers who could be impacted by portal technologies [8].

Looking again at SEI, we see a reasonably proportional investment in R&D across US S&E companies of roughly 6%.

Given that the Pareto Principal applies to revenue distribution among businesses, we can infer an 80/20 split among industrial researchers [9]. With 20% having access to the latest high technology tools to perform their research and 80% utilizing effective, but cost-restricted technologies. In academia, SEI shows that the top 100 spending universities spent 80% of the academic R&D money in the US. This is significantly more lopsided ratio, but as a lower bound, the Pareto Principal holds for academic research as well. Thus, it is reasonable to assume at least an 80/20 split between the haves and the have-not across US R&D in both sectors today, indicating there are at least 277,000 underserved researchers in the US alone.

II. Building for Tomorrow

How does one go about reaching the 277K scientists on the other side of the digital divide? Raising taxes to build 10,000 portals is not realistic. It also does not address the fundamentally deeper issue of utility. That portals provide value to their users is well documented [10][11][12]. What value they provide and at what cost are less well-documented questions. We look at 5 portals from the XSEDE Gateways Program [13] as short case studies.

Galaxy is an open, web-based platform for data intensive biomedical research. Scientists can download a copy of Galaxy for private use or they can use the hosted Galaxy instance, often called Galaxy Main. The Galaxy Main portal contains over 2500 application codes in its "Shed" that users can leverage for their work. Historically, the vast majority of users select a small number of codes that they use for all their work. In 2012, users ran over 100k jobs a month through Galaxy Main. In addition to application registration and job submission, Galaxy also supports visualization and data publication. Both are popular features, but neither is the primary focus of the portal. Does that mean that they were a waste of time? No. Galaxy Main serves over 28k users. There are many other features built into Galaxy, but the point of this observation is that as a portal, Galaxy casts a wide net and tries to provide something of value to every one its users. The price of doing so is added complexity, greater development costs, and a larger investment in supporting infrastructure to run the application. Initially funded by two awards totaling just under $1.4M in 2006 from the National Science Foundation (NSF), the additional use cases necessitated another round of funding totaling $1.1M from NSF. Recently, to support the expanding user community and support different resource utilization patterns, another round of funding totally $5.8M was obtained from the National Institute of Health to carry the project through 2018. Even for a successful portal like Galaxy Main, maintaining continuous funding and retaining talent are ongoing concerns.

GridChem is a desktop application supporting the computational chemistry community. Its mission is to enable computational and experimental scientists to do more computational chemistry by providing capability computing resources and services at their fingertips. To that end, the first release of GridChem provided federated identity management, job tracking, system monitoring, scheduling, enforcement of proprietary software license agreements, distributed account management, large data management, full experiment reproducibility, and integration with application codes installed on the user's local system. Many of the features took a significant amount of time to build which pushed back the first release of the software by nearly a year. However, after its first 3 years in production GridChem had enabled 500 plus researchers to publish over 60 papers and complete 6 dissertations. The software was used as a teaching tool in undergraduate chemistry classes at The Ohio State University, the University of Illinois, and the University of Kentucky to expose hundreds of students each semester to computational chemistry. The value of GridChem is obvious, however that value came at the up front cost of 6 man-years of development at a cost of $2.7M to provide enough features to simultaneously support undergraduate students and full professors alike. Further operation led to another $1M in funding to support workflow integration and expanded support for determining appropriate parameters for use in different experiments.

The Cipres Science Gateway is a public resource for inference of large phylogenetic trees. As of this writing, Cipres exposes 30 different tools for use on a preconfigured set of systems ranging from large shared compute clusters to private virtual machines. Users access these tools through a form-driven web interface. The process of developing Cipres included building multiple interfaces for each applications, job scheduling heuristics, data management, accounting systems, identity management, and integration with multiple infrastructure providers. These features took a significant amount of time, $4.5M in funding from NSF, and a very talented team of programmers to develop. The result of that work was a wildly successful portal. Cipres now serves over 700 users and has been used to run nearly 100k simulations burning over 15M compute hours. After 18 months in production, Cipres' usage was outgrowing its infrastructure. Due to the heavyweight nature of the infrastructure it took another year of development and $1.5M in funding from NSF to allow them to scale out to other systems and move away from a community account model. While growth is a common problem of success, this particular problem came at the end of the project's original funding. Had it not been for the talent and passion of the development team, Cipres would not have been able to address its growing pains and, as such, would have stalled until the next round of funding arrived.

NanoHub is web application built upon the Joomla CMS [14] and designed to support nanotechnology research and education. It provides over 270 simulation tools, 3800 seminars, tutorials, and teaching materials, 200 distinct user groups, and a mature workflow engine called Pegasus, which supports job execution across heterogeneous systems. Behind NanoHub lies a series of web services, command line tools, a full CMS, and an application-authoring tool. The portal as a

whole was built to support a large community and it does so very well. In 2010, NanoHub saw 10k users run 380k simulations. In 2011, 11k users ran 400k simulations. In 2012, 12k users ran 410k simulations. Clearly a lot of people are doing a lot of work and the growth is cumulative year over year. Such usage indicates that the portal is reaching a significant number of people, exposing them to some functionality, helping them accomplish a specific task, and a percentage are coming back year after year. The numbers are impressive, but the behavior is consistent with other portals. Users come in, find a few tools and/or features of value to them, and make a routine using those specific tools and/or features for the duration of their interaction with the portal.

Success comes at a price, and the price of building NanoHub was $14M from NSF. Sustaining NanoHub amid rapid growth has been an even more expensive activity. Their latest round of funding is $21.9M from NSF starting in 2013. To put that in perspective, NanoHub is a Joomla instance with a lot of custom plugins and some back-end services to support running nanotechnology simulations at an average rate of one simulation every 78 seconds. Looking at the CMS alone, the site receives 8.5 million hits a month. That is roughly half the traffic of edublogs.com, the leading educational blog provider in the world with 1.6 million blogs since 2005 [15]. Given comparable expenditures and team sizes between the two organizations, the cost of custom development and supporting the back-end infrastructure of NanoHub costs roughly 200% more than the total cost of running the website alone.

The Extreme Science and Engineering Discovery Environment (XSEDE) [16] is a National Science Foundation (NSF) funded national cyberinfrastructure (CI) that provides a set of large resources for scientific simulation and analysis. The XSEDE User Portal (XUP), led by TACC, is the primary interface for users to XSEDE. It provides user account management, project management, documentation, data management, and a myriad of other features to help users be productive on the XSEDE CI. It was built on the Liferay Portal platform [17], an enterprise open-source Java portal framework. The Liferay platform itself provides many features out of the box, including a content management system, wikis, calendaring, web forms, user forums, security and access control, and user notifications. Liferay also provides a plugin development platform for extending the portal with plugins and portlets. As an enterprise portal, Liferay is one of the leaders in the field, but there is a significant financial and human cost associated with its use. The cost of training, professional consulting, and enterprise support must be considered.

XUP has a very different focus than the previously mentioned science gateways, and it is more of a "destination portal" then "science gateway". But it is another example of a large, enterprise project that is designed to be a one-stop shop that provides users of XSEDE everything they need to be productive on XSEDE, excepting streamlined job execution. The development of the XUP is a continuation of the previous 5 years of development of the TeraGrid User Portal [18]. The initial cost of development for the first TeraGrid User Portal was on the order of $800k. Since then another $1.7M has been invested in the dedicated team of developers maintaining active development on XUP, adding features, addressing user issues, and providing the general maintenance required of a portal with over 12,000 registered users that supports a variety of user communities within the XSEDE organization, each with different needs.

In order to provide all of this functionality, the XUP, and to a lesser extend the XSEDE website, rely on a suite of services that provide the backend information services. These services include relational databases, non-relational "NoSQL" databases, SOAP and REST web services, flat file parsers, and many other services that interact directly with the resources in the XSEDE CI. The front end is built from many custom developed and specialized portlet applications, as well as out-of-the-box Liferay portlets. The system works because the development team has administrative access to the entire XSEDE infrastructure. They are able to obtain information that other gateways simply do not have access to. As a result, the portlets developed for XUP and the functionality they provide cannot easily be replicated simply by copying over the portlet code.

Each of the above portals is different in focus and function, but they are all successful science gateway projects and provide broad functionality. That functionality is often targeted at a small set of users who, for a given portal, will only ever use a subset of the features. The cost of these portals in terms of time and effort are all measured in multiple man-years and millions of dollars before they ever had a single user. They were designed to accommodate thousands of users when they went live and they made sure they could support a thousand users before they tried to support one. From their inception they were targeting long-term operational goals rather than short-term results. To be clear, there is nothing wrong with that, but it is an important distinction to make. The image of a successful science gateway promoted over the last decade was a portal built to support users of tomorrow rather than something that will get the results they need today.

III. BUILDING FOR TODAY

The reality for many scientists on the wrong side of the digital divide is that they do not need portals built for tomorrow; they need gateways built for today. They are content using their current workflows, but are willing to adopt technologies that make their workflows more efficient, more powerful, or less painful. They will gladly set down Outlook for Gmail, their departmental FTP server for Dropbox, and the server under their desk for a virtual machine on Amazon. These scientists are not pushing the boundaries of size and scale, but they are, in aggregate, performing the bulk of the science done today.

These scientists do not live in an enterprise world and their experimental processes are much less rigid than those of the organizations building the previously mentioned portals above. These scientists look for silver bullets, or the next best thing, to accelerate the time between proposal and discovery. And if a miracle doesn't come, simply squeezing an extra 5% out of their week would be a huge win for them.

Whether they realize it or not, these scientists have embraced the spirit of Agile [19] development that drives today's web ecosystem. In contrast to the monolithic deliverable approach historically taken by portal projects, today's web creates and innovates at a blazing pace. Working from incremental release to incremental release, actively engaging users, and obsessing over a results-first focus enables high quality sites and services to be created and refined into stable products in the time it takes most portals to make their first release. One notable example being an application called Burbn, which over a seven-month timeframe morphed from a web application to an iPhone app to a cross-platform application, then changed its focus and relaunched as Instagram [20]. A second example is the social bulletin board site Pinterest, which spent 3 months in development before launch, then constantly adapted to user feedback over the next year before expanding as an iPhone app and exploding into the giant of today [21]. A third example is a relatively new startup called GivePulse [22], which spent 4 months iterating over designs and features with local philanthropic organizations before publicly launching as a site enabling the promotion, matchmaking, and coordination of volunteers with events. While each of these examples gives launch timelines in terms of months, their feature development cycles were on the order of 1-2 weeks with updates and bug fixes pushed out daily.

In each of these examples, the product that went to market was markedly different from the project that was originally conceived. They survived due to their ability to leverage existing open source technologies, prototype ideas, and add small bits of functionality that they could present to their audience and find out if it had enough promise to invest more time into its continued development.

When attempting to serve the needs of the lower half of the digital divide, developers would do well to learn from Instagram, Pinterest, and GivePulse and take these lessons to heart. Start first by understanding that not every project can or should be the next big thing. Providing a tool that helps a researcher to see a problem in a different light and enables the discovery of a solution is a significant contribution in its own right. The gateway does not need to serve every conceivable user community to be a success.

While much of what one interacts with on the web is provided as a hosted service, i.e. Facebook, Gmail, DailyMile, etc., there is no reason that every gateway should be a hosted service. Most new desktop computers have more CPUs, memory, and disk available than the virtual machines powering the hosted services we rely upon. Furthermore, modern web browsers are constantly evolving with powerful new features both for the user and the developers of web applications. At the same time, as more browsers have adopted web standards put forth by the W3C [23] these features are more available for use natively in the browser without the need for polyfills such as Adobe Flash [24]. Some of these features can even leverage advanced capabilities of the underlying system such as GPU accelerated CSS rendering and animations. The latest CSS modules, such as transforms [25] and transitions [26] are even beginning to push the boundaries of 3D graphics. Combined, this makes the development of feature-rich, high-performance, and reliable web applications using only HTML, CSS, and JavaScript a reality.

By moving away from monolithic frameworks and large, server-side stacks to client-side applications built using only HTML, CSS, and JavaScript and leveraging RESTful APIs, one can rapidly develop powerful, targeted applications that can be quickly deployed, are highly scalable and "cloud-friendly."

One example is a tool created by Andre Mercer, an undergraduate student at the University of Arizona. Andre created a simple web page that submitted a request to the iPlant Foundation API to run a GeneSeqer job [27]. He spent an afternoon creating the page, then showed it to his supervisor, iterated a handful of times on the wording and default settings, then pushed it out into the group's website. Jon Duvick, a bioinformatician in a sister group saw the tool and decided to add it to his site as well as embedding it as part of his cloud-based annotation pipeline. Based on the success of the original tool, Andre is now adding data browsing via a jQuery [28] dialog box to the form so users can run analysis on files stored in the Cloud as well their desktop.

IV. Building on a Solid Foundation

One of the reasons that applications can be built with such light front ends is that they now rely upon a growing number of web-friendly APIs for much of the work. The API watchdog Programmable Web has tracked the growth and adoption of APIs since 2005 and has seen an explosion of new APIs in the last 2 years [29]. Much of this growth has been attributed to the fact that, "APIs are helping companies do business, with the tradeoff between adding an external dependency being out-shined by the ability to move faster building upon someone else's expertise [30]." In short, APIs allow companies to run lighter and move faster.

For new applications the abundance of APIs completely changes the established paradigm. API providers offering access to cloud storage, authentication, identity management, and Backend-as-a-Service (BaaS) [31] have redefined how applications are built. Things that used to take months to build and test are now leveraged as hosted services and integrated in

an afternoon. One well-known benefactor of building on the shoulders of other APIs is the communication platform provider Twilio [32]. From its inception Twilio has leveraged Amazon Web Services to handle spikes in demand and offload much of its compute load while focusing on the core part of their service, the communication platform.

Of the thousands of public APIs available today, and the hundreds targeted towards science, there are remarkably few that provide a generic platform for computational science. The SoapLab [33] project provides mechanisms for accessing SOAP services through a common interface, but it does not deal with federated identity, sharing, or access control. The NEWT project exposes HPC systems on the web, but is restricted in scope to systems and services at NERSC [34]. Recently, the CHAIN project has promoted an end-to-end solution for science gateway development based on open standards including JSR 168 and 268, OpenLDAP, SAGA, and PKSC-11 [35]. The framework is still relatively new at the time of this writing and as such, could not be included in the evaluation process leading up to the development of the solutions described in this paper. Based on early successes, CHAIN seems like an exciting project to watch going forward. The target audience and advertised use case, however, are more in line with traditional portal development than lightweight gateways creation. The gUSE project provides a mature web service framework for running workflows, storing data, and registering applications [36]. Further, it has existing integration with the WS-PGRADE portal to provide an out-of-the-box front end based on Liferay. As with CHAIN, the PGRADE and gUSE project timelines ran parallel to that of the work in this paper. Futhermore, the approach taken by gUSE to provide a SOAP-based service stack runs counter to the desire of current web developers to interact with REST services in an asynchronous manner.

In response to the dearth of platform APIs available for general science the iPlant Collaborative created the Foundation API [37]. The Foundation API is a RESTful Science-as-a-Service platform for building modern applications. It includes services that allow consumers to securely conduct science, manage data, and share and curate their work. Foundation exists as a hosted, multi-tenant cloud service that is freely available to the open science community. Version 1 of Foundation supports the following services.

- *Apps*: Allows users to register and discover scientific codes that can be run via the Jobs service. There are currently over 160 scientific codes both public and private that can be run across multiple high performance compute systems.
- *Auth:* token-based authentication service. Issues limited use tokens that can be restricted to a timeframe and number of uses and revoked when needed.
- *Data*: Acts as a Rosetta stone for biological data. Supports the conversion of data between known formats.
- *IO*: provides multiprotocol data movement and management.
- *Jobs*: Handles the end-to-end execution of registered applications on a heterogeneous set of systems ranging from HPC to raw VMs.
- *Monitor*: constantly monitors Foundation and its dependent services. Provides real-time and historical monitoring test results.
- *PostIt*: pre-authenticated URL shortening.
- *Profile*: search and view profiles of other users within the API.
- *Systems*: provides information about systems available from Foundation including status, stats, and accessibility.

Since its initial release in November 2011, the Foundation API has supported over 250 unique projects representing 10k scientists worldwide. Users burned nearly 9M SUs running over 10k jobs, leveraging 200 application codes installed on HPC systems at PSC, SDSC, and TACC. Version 2, due out prior to the publication of this paper, will add the following services as well as expanded support for system registration, federated identity management, additional execution platforms, and a more mature callback system.

- *Systems*: discovery and register storage, authentication, and execution systems for use throughout the API.
- *Transfer*: move data from anywhere to anywhere using multiple protocols.
- *Metadata*: create, search, and infer metadata about any resource (file, job, person, system, etc.) within the API.

By hiding all the heavy lifting of accessing systems, moving data, running simulations, and establishing relationships between people, data, and devices, consumers are freed up to focus on their science and developers are able to focus on innovation at the application layer rather than infrastructure at the system level.

V. Your Next Science Gateway

Turning back to Andre's GeneSeqr form, this tool is as basic an example of a science gateway as one can find, but it gets the job done. A scientist with remedial programming capabilities can stand up a static web page on their personal computer, a public web server, or on a CDN such as their public Dropbox folder, Amazon S3, or even a free Yahoo Sitebuilder page. When technology is that easy to adopt and reuse, the possibility for it to reach a broad audience increases dramatically. The question then becomes, how can we build tools to accomplish tasks requiring a bit more complexity and interaction and yet make them as simple to adopt and reuse as Andre's GeneSeqr form?

In recent years, a variety of toolkits and frameworks for developing modern web applications have emerged that aid in the development of lightweight, responsive, standards-driven, front-end components. These projects are open source, have very large user communities, and are supported by real companies such as Twitter (Hogan.js, Bootstrap) [38][39], DocumentCloud (Backbone.js, Underscore.js) [40][41], and Google (Yeoman.io) [42]. Furthermore, as HTML5 has come into its own and the development of single-page applications has become commonplace in the commercial web, it makes sense to begin using these technologies in science gateways.

With the goal of building tools that are simple to adopt and reuse in mind, we have developed a toolkit using these frameworks with the intention of doing for science gateways what jQuery did for JavaScript and Web 2.0. We leverage the iPlant Foundation API as a backend, and provide plugins for Backbone.js that allow a Backbone.js application to easily use the Foundation API without the developer needing detailed knowledge of its inner workings. These plugins provide implementations of the objects in the API as Backbone Models and Collections that can be readily used to build science gateways. This allows the gateway developer to focus more on gateway development and less on handling the web service calls to the backing API.

TABLE 1. THE FULL LIST OF FOUNDATION API BACKBONE.JS PLUGINS AND THE FUNCTIONALITY THEY PROVIDE.

Plugin Name	Functionality Provided
backbone-foundation	Core support for using Foundation API
backbone-foundation-apps	Application discovery and registration
backbone-foundation-data	Data transformation and staging
backbone-foundation-io	Data management and movement
backbone-foundation-jobs	Job submission and monitoring
backbone-foundation-profile	Identity management
backbone-foundation-systems	Resource discovery and monitoring
backbone-foundation-post-it	Pre-authenticated URL shortening

The Backbone-Foundation library is broken into separate plugins that can be included in an ad hoc manner based on the needs of the application. At the core is the main backbone-foundation.js file, which provides functionality for basic interaction with the Foundation API. By providing extensions of the default Backbone Model and Collection objects, the Foundation API can be used transparently through the standard Backbone API. Also in this plugin is an implementation of the Foundation Auth API and Model objects for authenticating and obtaining API tokens for authenticated use of the API. Finally, we include an Events object that can be used to manage API-aware events across the application.

Support for the remaining Foundation services is provided by the additional Backbone plugins listed in Table 1. Each plugin depends on the Backbone-Foundation core library to provide the API integration. The only other dependency of the Backbone-Foundation library is Backbone.js itself.

We selected Backbone as platform for multiple reasons. Backbone adheres closely to our build-for-today design philosophy. It is specifically designed for developing rich, yet lightweight client-side applications that utilize a RESTful API backend. Backbone applications follow the Model-View-Controller (MVC) design pattern making for code that is easy to develop, maintain, and extend. As a JavaScript application framework, it can be easily integrated into other environments and web platforms such as Liferay or Drupal. Finally, Backbone is a widely used and popular framework with an active user community and multiple examples of large-scale applications built on top of it. Two examples of large-scale Backbone users are the Khan Academy [43] and Coursera [44], both providers of massively open online courses (MOOC). One can imagine the benefits of having a computational science course with labs and homework that included hands-on access to a computational environment where students can gain experience using actual large-scale, high-performance computational systems. Using the Backbone-Foundation plugins we have developed, these MOOC providers could easily integrate the Foundation API into coursework offered through those sites.

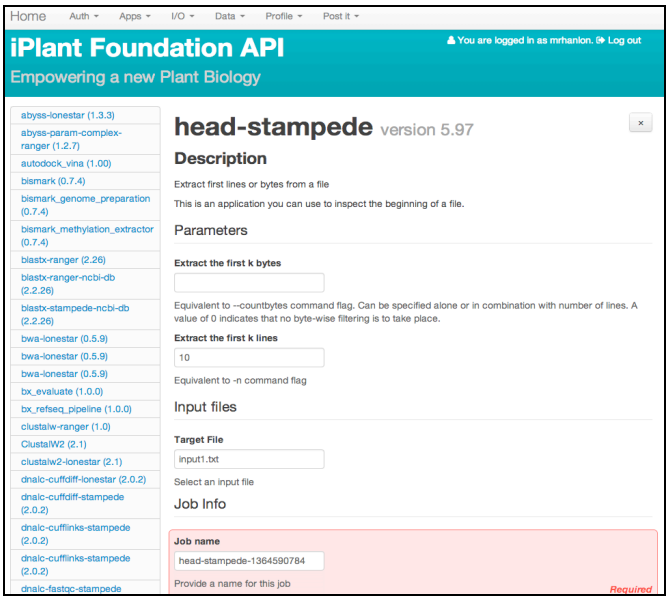

Figure 1. A standalone boilerplate gateway built using Backbone.js and the Backbone-Foundation plugins. This application leverages the iPlant Foundation API to provide authentication, data management, application discovery, and job submission with no backend other than the Foundation API.

We have also developed a complete Backbone application (Figure 1) as a boilerplate science gateway using the Backbone-Foundation library. The Backbone-Foundation library and Foundation API are white-label components that can be readily and easily used to develop your own science gateway. This application is built using Backbone for the application framework, Twitter Bootstrap for the front-end components and HTML structure, and has no backend other than the Foundation API and a web server to host the static assets (which could also be hosted out of the Foundation API).

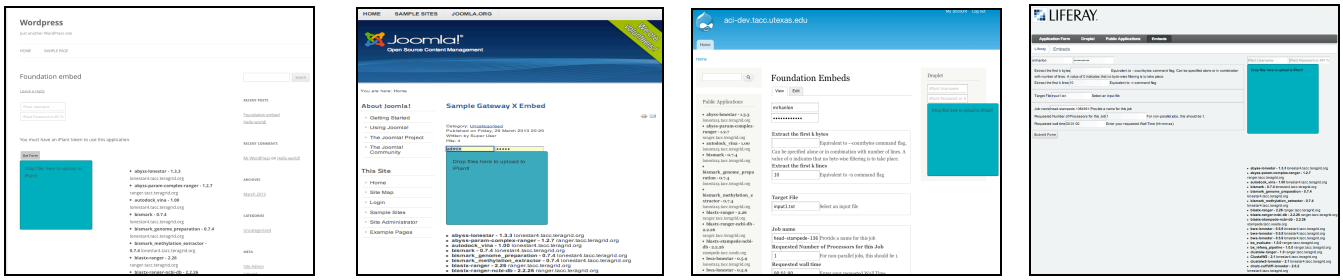

Figure 2. Embedding gateway widgets as a page in privately hosted CMS. From left to right: Wordpress, Drupal, Joomla, and Liferay sites.

The development of this boilerplate application took a single developer less than a month to complete and includes authentication, data management, application discovery, and job submission.

Lastly, we have developed a collection of embeddable "widgets" that provide discrete slices of functionality that can be used immediately to add advanced capabilities to any web page or existing portal or gateway with no more effort than adding a Twitter "Tweet this" or Facebook "Like" button.

These widgets are also built on top of the Backbone-Foundation plugin library. To include a widget in a page, the page author only needs to add a reference to the widgets script and a single div tag with the widget configuration contained in data attributes on the tag. The widgets script acts as a scout script to discover the widget div, determine the desired widget, and then inject the appropriate widget into the page.

Foundation widgets can be easily used in any HTML page and many CMS platforms such as Wordpress, Drupal, or Joomla (Figure 2). And because they leverage the Foundation API backend and don't require local server configuration to use, they can be used even on cloud-hosted sites such as Wordpress.com.

The widgets available at the time of this writing include a drag-and-drop file uploader, an application discovery widget, and a job execution widget. The uploader widget gives a drag-and-drop upload functionality using the HTML5 FileAPIs allowing users to drag files from their desktop into the web browser in order to upload to the iPlant Data Store. The application discovery widget allows embedding up-to-date lists of available iPlant applications into any page essentially providing an application catalog for browsing and searching applications. The job execution widget allows the embedding of an application-specific job submission form in any page.

VI. Getting from Today to Tomorrow

The discussion on building for today has been targeted at researchers developing new gateways up to this point. Previous sections have demonstrated how one can bootstrap an idea into a functional science gateway with a relatively short ramp-up using existing APIs and services like the Foundation API. However, these same development principles can benefit existing gateways and portals, enabling the rapid development and deployment of new features in a results-driven fashion no matter how established an existing portal may be.

Consider the example of the Liferay enterprise platform. Just as with the CMS platforms mentioned above, one can drop in a Foundation application using only HTML and JavaScript, and utilizing the Liferay Web Content Display portlet as shown in Figure 2. Or, if something more robust is needed, the application can be wrapped in a portlet and deployed it in the same way one would deploy any custom portlet.

Whether this functionality is packaged as content (HTML and JavaScript) or as a plugin, module, extension, or JSR 268 portlet for a specific platform, migrating the functionality from a lean prototype gateway using these tools, to an enterprise solution with all the bells and whistles is a trivial process. Deploying features built entirely on the front end is not a deliverable that consumes months of time and effort. On the contrary, it is more akin to migrating static content from one site to another.

Finally, as mentioned above, forward integration isn't limited to wrapping bits of functionality as pages. It is possible to embed custom widgets to provide one-off functionality such as activity streams, share buttons, data drop boxes, submission forms, and directory trees just to name a few.

The process of embedding a widget is the same as that of adding a page. However, for easier adoption, an AJAX driven widget generator is provided on the Foundation API developer's website to help users create widgets based on their unique constraints such as styling, default values, and restricted permissions.

VII. Conclusion

Science gateway development has historically been an enterprise effort. In recent years, the introduction of lightweight web technologies and REST APIs have changed the way modern applications are built. By leveraging the technologies of today and decoupling complex infrastructure from gateway front ends, developers can respond to change faster, innovate more quickly, prototype more easily, and drastically reduce their time to production. This paper presents a set of reusable, white labeled, front end components written

entirely in HTML, JavaScript, and CSS that leverage the Foundation API and enable just such a transformation. By utilizing the backbone-foundation plugins as fully functional, interchangeable components, both new and existing gateways can shift their attention from tedious integration to rapid innovation that can impact researchers today rather than tomorrow. Both the gateway components and the Foundation API are freely available for use today at https://foundation.iplantcollaborative.org.

ACKNOWLEDGMENT

The iPlant Collaborative is funded by a grant from the National Science Foundation Plant Cyberinfrastructure Program (#DBI-0735191). This work was also partially supported by a grant from the National Science Foundation Cybersecurity Program (#1127210).

REFERENCES

[1] Miller, M.A., Pfeiffer, W., and Schwartz, T. (2010) "Creating the CIPRES Science Gateway for inference of large phylogenetic trees" in Proceedings of the Gateway Computing Environments Workshop (GCE), 14 Nov. 2010, New Orleans, LA pp 1 - 8.

[2] Dooley, Rion, Kent Milfeld, Chona Guiang, Sudhakar Pamidighantam, and Gabrielle Allen. 2006. From proposal to production: Lessons learned developing the computational chemistry grid cyberinfrastructure. Journal of Grid Computing 4, no. 2: 195-208.

[3] Demeler, BORRIES. 2005. UltraScan: a comprehensive data analysis software package for analytical ultracentrifugation experiments. Modern analytical ultracentrifugation: Techniques and methods: 210-229.

[4] Goecks, Jeremy, Anton Nekrutenko, James Taylor, and T Galaxy Team. 2010. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. Genome Biol 11, no. 8: R86.

[5] Klimeck, Gerhard, Michael McLennan, Sean P Brophy, George B Adams, and Mark S Lundstrom. 2008. nanohub. org: Advancing education and research in nanotechnology. Computing in Science & Engineering 10, no. 5: 17-23.

[6] Norris, Pippa. 2003. Digital divide: Civic engagement, information poverty, and the Internet worldwide. Vol. 40. Cambridge: Cambridge University Press, September.

[7] 2012. Science and Engineering Indicators 2012. http://www.nsf.gov/statistics/seind12/start.htm.

[8] Bureau of Labor Statistics, U.S. Department of Labor, *Occupational Outlook Handbook, 2012-13 Edition*, Medical Scientists, on the Internet at http://www.bls.gov/ooh/life-physical-and-social-science/medical-scientists.htm (visited *May 10, 2013*).

[9] Fawcett, Henry. *Manual of political economy*. Macmillan, 1888.

[10] Wilkins-Diehr, Nancy, Dennis Gannon, Gerhard Klimeck, Scott Oster, and Sudhakar Pamidighantam. 2008. TeraGrid science gateways and their impact on science. Computer 41, no. 11: 32-41.

[11] Lawrence, Katherine A., and Nancy Wilkins-Diehr. "Roadmaps, not blueprints: paving the way to science gateway success." In Proceedings of the 1st Conference of the Extreme Science and Engineering Discovery Environment: Bridging from the eXtreme to the campus and beyond, p. 40. ACM, 2012.

[12] Wilkins-Diehr, Nancy, and Katherine A. Lawrence. "Opening science gateways to future success: The challenges of gateway sustainability." In Gateway Computing Environments Workshop (GCE), 2010, pp. 1-10. IEEE, 2010.

[13] "XSEDE | Overview." 2011. 29 Mar. 2013. https://www.xsede.org/gateways.

[14] 2005. Joomla! The CMS Trusted By Millions for their Websites. http://www.joomla.org/.

[15] 2005. Edublogs – education blogs for teachers, students and schools. http://edublogs.com/.

[16] "nsf.gov - National Science Foundation (NSF) News - XSEDE Project ..." 2011. 29 Mar. 2013. http://www.nsf.gov/news/news_summ.jsp?cntn_id=121181.

[17] 2002. Liferay.com: Enterprise open source portal and collaboration software. http://www.liferay.com/.

[18] Dahan, Maytal, Eric Roberts, and Jay Boisseau. 2007. TeraGrid User Portal v1. 0: Architecture, Design, and Technologies. International Workshop on Grid Computing Environments. November 28.

[19] Martin, Robert Cecil. 2003. Agile software development: principles, patterns, and practices. Prentice Hall PTR, September 1.

[20] "Instagram." 2009. 29 Mar. 2013. http://instagram.com/.

[21] "Pinterest." 2009. 29 Mar. 2013 .http://pinterest.com/.

[22] "GivePulse | Enabling Everyone to Volunteer." 2012. 22 Mar. 2013. https://www.givepulse.com/.

[23] "World Wide Web Consortium (W3C)." 29 Mar. 2013. http://www.w3.org/.

[24] 2006. Adobe - Flash Player. http://www.adobe.com/software/flash/about/.

[25] "CSS Transforms." 2012. 29 Mar. 2013. http://www.w3.org/TR/css3-transforms/.

[26] "CSS Transitions." 2009. 29 Mar. 2013. http://www.w3.org/TR/css3-transitions/.

[27] Schlueter, Shannon D, Qunfeng Dong, and Volker Brendel. "GeneSeqer@ PlantGDB: Gene structure prediction in plant genomes." Nucleic Acids Research 31.13 (2003): 3597-3600.

[28] "jQuery." 2006. 29 Mar. 2013. http://jquery.com/.

[29] "ProgrammableWeb - Mashups, APIs, and the Web as Platform." 2005. 29 Mar. 2013. http://www.programmableweb.com/.

[30] 2012. 8,000 APIs: Rise of the Enterprise - ProgrammableWeb.com. http://blog.programmableweb.com/2012/11/26/8000-apis-rise-of-the-enterprise/.

[31] "Backend as a service - Wikipedia, the free encyclopedia." 2012. 29 Mar. 2013. http://en.wikipedia.org/wiki/Backend_as_a_service.

[32] "Twilio Cloud Communications - APIs for Voice, VoIP and Text ..." 2005. 29 Mar. 2013. http://www.twilio.com/.

[33] 2007. Soaplab2. http://soaplab.sf.net/.

[34] Cholia, Shreyas, David Skinner, and Joshua Boverhof. 2010. NEWT: A RESTful service for building High Performance Computing web applications. Gateway Computing Environments Workshop (GCE), 2010. IEEE, November 14.

[35] CHAIN, (2010) Co-ordination & Harmonisation of Advanced e-Infrastructures EU FP7 project http://www.chain-project.eu) project

[36] Peter Kacsuk, Zoltan Farkas, Miklos Kozlovszky, Gabor Hermann, Akos Balasko, Krisztian Karoczkai and Istvan Marton: WS-PGRADE/gUSE Generic DCI Gateway Framework for a Large Variety of User Communities Journal of Grid Computing, Vol. 9, No. 4, pp 479-499, 2012.

[37] Dooley, Rion, Matthew Vaughn, Dan Stanzione, Steve Terry, and Edwin Skidmore. Software-as-a-Service: The iPlant Foundation API.

[38] 2013. Hogan.js - Twitter on GitHub. http://twitter.github.io/hogan.js/.

[39] 2011. Bootstrap. http://twitter.github.com/bootstrap/.

[40] 2011. Backbone.js. http://backbonejs.org/.

[41] 2008. Underscore.js. http://underscorejs.org/.

[42] 2012. Yeoman - Modern workflows for modern webapps. http://yeoman.io/.

[43] 2009. Khan Academy - Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Khan_Academy.

[44] 2012. Coursera. https://www.coursera.org/.