TechTarget | TheServerSide.com

E-guide

PRO+ Content

Representational State Transfer (REST) Challenges

In this e-guide

In this e-guide:

For those who are tasked with developing public APIs, the current industry thinking is to deliver those APIs as REST-based interfaces. The virtues of REST are numerous: REST is ubiquitous over a web-based network, it is interoperable with both Java and .NET frameworks, they can be invoked easily by single page interface technologies like AngularJS and Ember, and of course, they can create fast and efficient web-based apps by allowing browsers to call back end services directly through asynchronous JavaScript calls.

But, as popular as REST-based services are, there are challenges that can complicate their use. Ahead, we look at some of the most challenging aspects of REST-based development and see how industry experts are addressing those challenges.

In this e-guide

First, we look at some of the challenges developers encounter when integrating REST with modular, OSGi-based apps. Expert Holger Staudacher, of EclipseSource, discusses the work that is being done to address these concerns, especially in terms of accessing the HTTP protocol from within an OSGi bundle.

And from the micro-domain of modularized OSGi apps, we move to the world of cloud computing, where massive distribution and globalization of processing power and memory can make managing REST-based APIs a problem, especially in terms of security.

And finally, if some of the problems REST-based protocols pose are too onerous for a given project to overcome, we review some alternatives to REST that companies like Uber are integrating into their real-time apps. After all, REST is a great solution, except for the times when it isn't, and when it isn't, it's good to know that alternatives do indeed exist.

In this e-guide

Conquering the challenge of integrating JAX-RS (REST) with OSGi modularity

Jason Tee, TheServerSide.com

In the open source Java community, the Eclipse Foundation has a long history of supporting developers with new and better ways to create and deliver applications. In the past three or four years, the Java community has taken great strides in two key areas that have made server side development significantly easier: RESTful web services and modularization through OSGi. But now the big challenge is what happens when we want to bring these two technologies together? This is the latest challenge the enterprise community is trying to address.

What does a RESTful web service implementation like Apache CXF or Apache Wink have to do with OSGi? Well, until recently, it's been difficult to work with JAX-RS in a way that's deeply integrated with OSGi. This disconnect has left developers struggling to address issues such as HttpService integration instead of spending their time focusing on effective RESTful API development.

> "People always struggle to connect OSGi concepts with JAX-RS concepts. We provide the glue."
> - Holger Staudacher, EclipseSource

In this e-guide

JAX-RS and OSGi integration

"We are mixing JAX-RS with OSGi on the service level. Because OSGI has a concept called service. You define a class and register it as a service and the framework instantiates it," said Holger Staudacher, a software developer and consultant at EclipseSource "If you register a service which is annotated with JAX-RS annotations, we pick it up and publish it also as a REST service. The benefit the user has then is that they can consume it in the OSGi instance directly, or via HTTP."

The OSGi JAX-RS connector that Staudacher is heavily involved in developing and maintaining via Eclipse is making things even simpler. "People always struggle to connect OSGi concepts with JAX-RS concepts. We provide the glue. What we encapsulate is the JAX-RS implementation. If you use the connector, you do not need to care about the implementation of JAX-RS. You can use Jersey, Restlet, whatever, you only need to know OSGi services and the JAX-RS API (mainly the annotations)." With connectors doing the heavy lifting of integration, developers may be able to concentrate more on building truly RESTful services. That's an important benefit.

In this e-guide

Struggling with OSGi and REST

It's an unfortunate fact that many development teams still haven't achieved an advanced understanding of the REST concept itself. As Holger pointed out, implementing remote procedure calls is not the same thing as implementing REST. One common pitfall he's noticed is that developers unfamiliar with the complete scope of REST often forgo hypermedia content.

Simon St. Laurent, the co-chair of Fluent 2014 and Senior Editor at O'Reilly, described the benefits of hypermedia for RESTful APIs, "Hypermedia changes the chunking. Instead of changing little pieces individually with lots of transactions all over the place, you're sending out a more complete document with links in it that act as options to the next place you want to go to." Hypermedia is becoming an essential part of the REST picture in an age where the documents being passed from computer to computer are larger than ever. In fact, it's the third step in the Richardson Maturity Model that developers must achieve if they wish to enter into the *Glory of REST*.

Taking REST and OSGi to the next level won't be simple. However, contributors at the Eclipse Foundation are hard at work coming up with open source APIs and other tools that are sure to help.

In this e-guide

How to secure REST API endpoints for cloud applications

Amy Reicher, Software quality and testing veteran

When designing APIs, developers must make good decisions about security design components that show who's using what API, when and for what purpose. This tip offers insights into the critical components for securing APIs.

Because security is a business requirement for any application deployed in the public cloud, API architects must make security a top priority when designing public APIs for others to use in client applications. Currently, most public APIs are based on representational state transfer (REST) with Javascript object notation (JSON) or extensible markup language (XML). The flexibility and open nature of REST, JSON and XML make secure design considerations essential to protect the service endpoints and data from threats.

When designing APIs, developers must make good decisions about security design components, such as authentication, authorization, monitoring and tracking, all functions that show which user is using what API, when and for what purpose.

In this e-guide

REST APIs, available over HTTP or HTTPS protocols, use JSON or XML for data formatting. REST performs these basic actions: GET (read or list), POST (create), PUT (update) and DELETE (delete).

In design, the first step to securing an API endpoint is to determine which exposed actions need to be secured. Secure design requires that architects consider authentication, authorization and protecting the session state. Adding security to an API design enables development resources to own and assist in controlling the API, along with IT or DevOps resources. Also, it allows IT and DevOps resources to focus on higher-level security concerns rather than tracking each public API the organization publishes.

When a developer or architect designs a REST API, it's imperative that the API be designed so that only authenticated and authorized users can perform actions.

Authentication and API keys

When a developer or architect designs a REST API, it's imperative that the API be designed so that only authenticated and authorized users can perform actions.

Hash-based message authentication code (HMAC) is an option that provides the server and the client each with a public and private key. The public key is

known, but the private key is known only to that server and that client. The client creates a unique HMAC, or hash, per request to the server by combing the request data and hashing that data, along with a private key and sending it as part of a request. The server receives the request and regenerates its own unique HMAC. The server compares the two HMACs, and, if they're equal, the client is trusted and the request is executed. This process is often called a secret handshake.

Another option is using open authorization (OAuth) within an API, along with a JSON Web token (JWT). Typically, OAuth specifies four roles: resource owner, client, resource server and authorization server. Each client using an API receives a unique API key embedded within a JWT for authentication.

Both options provide security on the API for authentication, and both options are complex to set up. The preference of the developer, architect or DevOps determines which option an organization chooses.

Authorization of public APIs

Authorization is achievable via several methods, but protecting the HTTP methods and whitelisting are preferred and available for monitoring with security intelligence systems.

In this e-guide

Protecting HTTP methods with a REST API means setting user privileges on the GET, POST, PUT and DELETE actions of REST. Not every API user needs access to each of these actions. Setting up authorization based on privilege level means that only the users authorized to use DELETE can actually remove data records. At a minimum, most users should have access to records or a list of information. Based on the API client terms or the organization's terms of use, some users may be allowed to create or update data records.

Authorization via privileged HTTP method access isn't applicable only to users; it may be applied to resource collections. Setting authorization based on privilege or a user's role is a relatively common, but effective layer to include in API endpoint security.

Whitelisting methods are another option. Whitelisting allows authorization based on a URL. The methods remain GET, POST, PUT and DELETE, but use of the methods is restricted based on the URL's presence in a whitelist, with an assigned set of actions allowed. If a URL is not in the list, then an error response code, such as 403 – Forbidden, is returned. It's advisable when using this method to ensure server and system information is removed from the standard error message so no internal system information is presented.

An advantage of using method whitelisting is it can be monitored via access logs and data tracked in logs or by security intelligence systems.

In this e-guide

Protecting session states

Most Web services and APIs are designed to be stateless, with a state blob being sent within a transaction. For a more secure design, consider using the API key to maintain client state if the API is using a server-side cache. It's a commonly used method in Web applications and provides additional security by preventing anti-replay. Anti-replay is where attackers cut and paste a blob to become an authorized user. In order to be effective, include a time-limited encryption key that is measured against the API key, date and time, and incoming IP address.

The key to effective security is layering combined with active monitoring. Layering provides one or more backup systems in case one is compromised. Data and access are still protected by the next layer. Monitoring activity with an API allows an organization to retain control of the API and ensure it's used in a manner consistent with its contract or terms of use and not as an access point into an organization's or API user's system.

An API architect must design API endpoint security from the beginning for the public API to provide the greatest value to an organization and its users, and deliver ownership of security within the development team. Even if an API is not used to connect to, or transfer confidential or common forms of protected data, API endpoints are access points and have been used as engines for hackers to

In this e-guide

gain unauthorized access to networks and data. Endpoint security provides protection for a business and its customers.

Next article

In this e-guide

Where are the REST alternatives for real-time applications?

George Lawton, TheServerSide.com

Developers face several challenges in building responsive mobile and browser applications that synchronize data across thousands or even millions of users or IoT devices. This used to only be an important consideration for massively multi-player games. But the success of Uber's application for displaying car location's to thousands of mobile users has encouraged many enterprises to consider how to leverage the same principles in new application.

Developers have grown comfortable with writing Web applications using RESTful interfaces to push data out to mobile applications. While this makes for easy development, these are not necessarily the most performant applications, since it relies on the chatty TCP protocol underneath, and battery eating publish subscribe mechanisms for pulling in updates. Developers end up writing a lot of glue code instead before sitting down to writing the interesting part of their application, said Matt DeBergalis, co-founder of The Meteor Project.

In this e-guide

The Importance of New Protocols

Developers are now starting to explore a variety of novel protocols that use UDP instead of TCP for pushing and pulling data between users and devices including WebSockets, MQTT, CoAP and DDP. At the same time, it is important to leverage a protocol that makes it easy for backend developers to expose real-time APIs in a simple standardized way.

> "As more enterprises build real-time APIs based on ad-hoc protocols, the inability to easily interoperate becomes a big challenge." - Slava Akhmechet, Founder of ReThinkDB

WebSockets make this technically possible but doesn't expose a good framework for real-time APIs in a way that REST does for non-real-time ones. Meteor's DDP specification is another good start, but has not been widely adopted yet. It provides a standard way for real-time applications to inter-operate, much like REST does for Web applications today. "As more enterprises build real-time APIs based on ad-hoc protocols, the inability to easily interoperate becomes a big challenge," said Slava Akhmechet founder of ReThinkDB.

Other protocols like CoAP and MQTT could grow in importance, particularly as enterprises begin weaving data from the IoT into real-time applications, said Ilan

In this e-guide

Sehayek, the CTO of Jitterbit, an agile cloud integration solution. They will not replace enterprise messaging standards like JMS, but they will be an area of focus for several verticals, especially manufacturing. There are simple ways to bridge CoAP or MQTT and REST so developers used to RESTful interfaces won't necessarily need to learn the ins and outs of these protocols immediately, unless they are focused on delivering optimal performance.

Building a real-time UI

In addition, there has been tremendous innovation on the front-end to make building real-time apps like these possible. Frameworks like Meteor, Angular, and React treat real-time use cases as first class citizens, and all major web browsers support technologies like WebSockets and long-lived HTTP connections to make pushing data to the browser in real-time fast and easy.

Sophisticated teams with big budgets and lots of time can implement this functionality on top of existing databases using a combination of traditional features such as views, triggers, and third-party queuing systems like RabbitMQ. That can work quite well, but is very challenging and expensive to build because this approach is very error-prone and takes a lot of development resources, said Akhmechet.

The biggest challenge with building these kinds of apps is in the data layer on the backend. All databases to date have been designed to respond to queries,

not to push data out to the back-end when something changes. Back-end engineers are forced to build complex infrastructures and sophisticated custom code to work around these limitations in the data layer. Much of this work requires custom code on the front-end, which really further complicates the development and deployment process.

Add a layer of abstraction with synchronized databases

New kinds of data infrastructure are emerging to tackle this problem like Firebase and RethinkDB. They push data to the application and take care of scalability issues, making real-time app development dramatically cheaper and far more accessible.

With Firebase the developer can just write data through a client API and respond to the changes. All the scalability and infrastructure challenges are taken care of by the Firebase service. RethinkDB is a newer open source contender. But it requires a back-end, since it cannot be accessed directly from the client. This makes it more difficult for front-end developers to get started.

"Real-time APIs for the web are relatively new. It will take some time for the industry to develop best practices and train engineers to ensure a robust healthy ecosystem," said Akhmechet

In this e-guide

Configuration changes are updated constantly in mobile application development as development companies try to keep up with demand. Mobile app testing, on the other hand, is already complicated by the number of paths needed to ensure a quality product. The process is burdened by testing requirements around screen size, platforms, connectivity and configuration.

With an already full plate, how does a mobile app testing team add in testing configuration changes in a release? Typically, configuration settings are not the highest priority changes, so testing them falls behind other feature testing commitments. You can try a couple of options to see what works for the mobile app testing team's schedule.

The first is random selection. For example, if there are 20 configuration changes pick the top 10 that are most important to clients, or that occur in the most-used features. Or list each by priority or importance and by the popularity of the associated feature and test your way down the list.

Testers also can divide the list between releases. Granted, not all features will be tested in the initial release, but a group can be tested each release until testing is completed. Create rotating test suites and mix them into your regression testing executions. Once each has been thoroughly tested, go back through and mix them at random or by feature and add them to the regression test along with their matching feature.

In this e-guide

Another logical option is to test them with the feature they control. If the testing numbers are too high rotate the feature and create small subsets of test suites that cover a piece of each feature. Switch out testing suites each regression cycle. Each testing suite contains a portion of every feature so in this way, full testing occurs over a series of regression testing events but all features are covered to some extent.

Next Article

In this e-guide

Getting more PRO+ exclusive content

This e-guide is made available to you, our member, through PRO+ Offers—a collection of free publications, training and special opportunities specifically gathered from our partners and across our network of sites.

PRO+ Offers is a free benefit only available to members of the TechTarget network of sites.

Take full advantage of your membership by visiting http://pro.techtarget.com/ProLP/

Images; Fotalia