REST API DESIGN RULEBOOK pdf

1: REST API Design Rulebook by Mark Masse | eBay

In today's market, where rival web services compete for attention, a well-designed REST API is a must-have feature. This concise book presents a set of API design rules, drawn primarily from.

All of those will contain many redundant actions. The URL should only contain resources nouns not actions or verbs. Then what is the correct way? But the question is how do we tell the server about the actions to be performed on companies resource viz. The paths should contain the plural form of resources and the HTTP method should define the kind of action to be performed on the resource. The important HTTP methods are as follows: GET method requests data from the resource and should not produce any side effect. POST method requests the server to create a resource in the database, mostly when a web form is submitted. POST is non-idempotent which means multiple requests will have different effects. PUT is idempotent which means multiple requests will have the same effects. There are few other methods which we will discuss in another post. HTTP status codes are bunch of standardized codes which has various explanations in various scenarios. The server should always return the right status code. The following are the important categorization of HTTP codes: If there is any error, like if employee 2 does not exist in the database, then the response code would be not be of 2xx Success Category but around 4xx Client Error category. And hence there is no need to transfer the same data again. Mostly if the server is undergoing maintenance. If the request body or response type is JSON then please follow camelCase to maintain the consistency. There will be no new set of APIs to handle these actions. Filtering For filtering the dataset, we can pass various options through query params. If there is any major breaking update, we can name the new set of APIs as v2 or v1. I would love to know your views on the pointers mentioned above. Please leave a comment, and let me know!

REST API DESIGN RULEBOOK pdf

2: REST API Design Rulebook by Mark Masse

In today's market, where rival web services compete for attention, a well-designed REST API is a must-have feature. This concise book presents a set of API design rules, drawn.

Hyphens - should be used to improve the readability of URIs Rule: Lowercase letters should be preferred in URI paths Rule: Consistent subdomain names should be used for your APIs Rule: A singular noun should be used for document names Rule: A plural noun should be used for collection names Rule: A plural noun should be used for store names Rule: A verb or verb phrase should be used for controller names Rule: Variable path segments may be substituted with identity-based values Rule: The query component of a URI may be used to filter collections or stores Rule: GET must be used to retrieve a representation of a resource Rule: HEAD should be used to retrieve response headers Rule: PUT must be used to both insert and update a stored resource Rule: PUT must be used to update mutable resources Rule: POST must be used to create a new resource in a collection Rule: POST must be used to execute controllers Rule: Content-Type must be used Rule: Content-Length should be used Rule: Last-Modified should be used in responses Rule: ETag should be used in responses Rule: Stores must support conditional PUT requests Rule: Location must be used to specify the URI of a newly created resource Rule: Cache-Control, Expires, and Date response headers should be used to encourage caching Rule: Cache-Control, Expires, and Pragma response headers may be used to discourage caching Rule: Caching should be encouraged Rule: Expiration caching headers may optionally be used with 3xx and 4xx responses Rule: Application-specific media types should be used Rule: Media type negotiation should be supported when multiple representations are available Rule: Media type selection using a query parameter may be supported Representation Design Rule: JSON should be supported for resource representation Rule: JSON must be well-formed Rule: XML and other formats may optionally be used for resource representation Rule: Additional envelopes must not be created Hypermedia Representation Rule: A consistent form should be used to represent links Rule: A consistent form should be used to represent link relations Rule: A consistent form should be used to advertise links Rule: A self link should be included in response message body representations Rule: A consistent form should be used to represent media type formats Rule: A consistent form should be used to represent media type schemas Error Representation Rule: A consistent form should be used to represent errors Rule: A consistent form should be used to represent error responses Rule: Consistent error types should be used for common error conditions Client Concerns Rule: New URIs should be used to introduce new concepts Rule: Schemas should be used to manage representational form versions Rule: Entity tags should be used to manage representational state versions Security Rule: OAuth may be used to protect resources Rule: The query component of a URI should be used to support partial responses Rule: Programmers and their organizations benefit from consistency Posted by.

3: 7 Rules for REST API URI Design

Along with rules for URI design and HTTP use, you'll learn guidelines for media types and representational forms. REST APIs are ubiquitous, but few of them follow a consistent design methodology. Using these simple rules, you will design web service APIs that adhere to recognized web standards.

Thank you to all the developers who have used Stormpath. Scalability â€” not necessarily its performance, yet rather how easy it is for RESTful APIs to adapt and grow and be plugged into other systems. Independency â€” with a REST API you can deploy or scale down specific parts of the application, without having to shut down the entire application or an entire web server form. Human readable â€” it uses very simple grammar and language, so a human can easily read it, including folks just starting to get into software development. What makes REST design difficult? It has no standard governing body and therefore has no hard and fast design rules. In the case of Stormpath, resources would be accounts, groups, or directories. So we recommend that, for a given resource in your REST API, you write a method that takes the resource itself as an argument, and the method contains all the functionality needed for said resource. How else do you keep things coarse grained? You work with collection and instance resources. A collection resource is what it sounds like â€” basically a folder containing similar resources. An instance resource is a singular instance of its parent resource. Why would you want to use POST to both create and update a resource? This is important because if for every update you make you are also sending over fields that are not updating, then your data plan is consuming more than it needs to. Because per HTTP specification, PUT is idempotent, meaning it has to have all its properties included or the result will not be the same. For example, if you first create an application without specifying a description, and then in a fourth call you send in the description, the state may have been different in between, therefore breaking the idempotency mandate. Have REST API documents link to other documents based on the notion of a media type A media type is a specification of a data format and a set of parsing rules associated with that specification. Likewise, as the server, return back a content-type header that notes how the data is actually being returned. So make sure to send through accept headers specifying what media type you want if on the client side , send through content-type headers telling the client what data format you are returning if on server side , and take advantage of customizing your own media types to make your rest API more flexible for your clients. Like what you see?

4: The Fundamentals of REST API Design

Description. In today's market, where rival web services compete for attention, a well-designed REST API is a must-have feature. This concise book presents a set of API design rules, drawn primarily from best practices that stick close to the Web's REST architectural style.

HTTP defines a set of standard headers, some of which provide information about a requested resource. Other headers indicate something about the representation carried by the message. Finally, a few headers serve as directives to control intermediary caches. The value of this header is a specially formatted text string known as a media type, which is the subject of Media Types. Content-Length should be used The Content-Length header gives the size of the entity-body in bytes. In responses, this header is important for two reasons. First, a client can know whether it has read the correct number of bytes from the connection. Second, a client can make a HEAD request to find out how large the entity-body is, without downloading it. Last-Modified should be used in responses The Last-Modified header applies to response messages only. The value of this response header is a timestamp that indicates the last time that something happened to alter the representational state of the resource. This header should always be supplied in response to GET requests. This header should always be sent in response to GET requests. Warning Generating an ETag from a machine-specific value is a bad idea. If client 2 decides to update the stored data, it may retry its request to include the If-Match header. Caches can be anywhere. When serving a representation, include a Cache-Control header with a max-age value in seconds equal to the freshness lifetime. The value is a time at which the API generated the representation plus the freshness lifetime. Including this header helps clients compute the freshness lifetime as the difference between the values of the Expires and Date headers. Tue, 15 Nov Thu, 01 Dec In this case, also add the Pragma: Caching should be encouraged The no-cache directive will prevent any cache from serving cached responses. Using a small value of max-age as opposed to adding no-cache directive helps clients fetch cached copies for at least a short while without significantly impacting freshness. Although POST is cacheable, most caches treat this method as non-cacheable. You need not set expiration headers on other methods. Implement clients and servers such that they do not fail when they do not find expected custom headers. If the information you are conveying through a custom HTTP header is important for the correct interpretation of the request or response, include that information in the body of the request or response or the URI used for the request. Avoid custom headers for such usages. Parameter names are case-insensitive. When more than one parameter is specified, their ordering is insignificant. The two examples below demonstrate a Content-Type header value that references a media type with a single charset parameter: Some commonly used registered media types are listed below: Unlike their more common counterparts, vendor-specific media types impart application-specific metadata that makes a message more meaningful to the web component that receives it. Vendor-specific media types may also be registered with the IANA. In other words, client programs should hardcode as few API-specific details as possible. However, this is a worthwhile trade-off since this media type communicatesâ€”directly to clientsâ€”distinct and complementary bits of information regarding the content of a message. Instead, by using a format parameter with a URI value, the WRML media type directs client programs to a cacheable document that provides links to other documents related to the format. In the example above, the representation of the document referenced by the format parameter http: By providing this code, available for various programming languages and runtime environments, an API can programmatically teach its clients how to interoperate with its representation formats. The future-proof nature of this design may prove especially useful when a REST API wishes to adopt a new format that is not yet widely supported by its clients. This separation of concerns allows multiple representation formats to be negotiated by clients and supported by REST APIs with relative ease. Media Type Schema Versioning The different versions of a given schema should be organized as different schema documents, with distinct URIs. The example below shows the URI of a schema document that details the fields and links of a soccer Player resource type: A schema document URI that ends with a number permanently identifies a specific version of the schema. Therefore the latest version of a schema is always modeled by two separate resources which

conceptually overlap while the numbered version is also the current one. This overlap results in the two distinct resources, with two separate URIs, consistently having the same state representation. Media type negotiation should be supported when multiple representations are available Allow clients to negotiate for a given format and schema by submitting an Accept header with the desired media type. Using media type negotiation clients can select the schema version that will work best for them. Media type selection using a query parameter may be supported To enable simple links and easy debugging, REST APIs may support media type selection via a query parameter named accept with a value format that mirrors that of the Accept HTTP request header. The virtual file extension approach binds the resource and its representation together, implying that they are one and the same. Therefore it should be used with careful consideration.

5: Hafizur's blog: REST API Design Rulebook

In today's market, where rival web services compete for attention, a well-designed REST API is a must-have feature. This concise book presents a set of API design rules, drawn primarily from best practices that stick close to the Web's REST architectural style. Along with rules for URI design.

6: REST API Design Rulebook - pdf - Free IT eBooks Download

REST API Design Rulebook and millions of other books are available for Amazon Kindle. Learn more Enter your mobile number or email address below and we'll send you a link to download the free Kindle App.

7: REST API Design Rulebook - PDF eBook Free Download

In today's market, where rival web services compete for attention, a well-designed REST API is a must-have feature. This concise book presents a set of API design rules, drawn primarily from best practices that stick close to the Web's REST architectural style.

8: Metadata Design - REST API Design Rulebook [Book]

RESTful API Designing guidelines â€" The best practices. Facebook, Google, Github, Netflix and few other tech giants have given a chance to the developers and products to consume their data through APIs, and became a platform for them.

9: REST API Design Rulebook PDF Download Free |

RESTful APIs are difficult to design because REST is an architectural style, and not a specification. It has no standard governing body and therefore has no hard and fast design rules. What REST does have is an interpretation of how HTTP protocol works, which allows for lots of different approaches for designing a REST API.

REST API DESIGN RULEBOOK pdf

Negro Soldiers and Sailors Memorial The Artists Guide to Selecting Colors Readings in Islamic financial services The Little Book of Vicarage Wisdom 5 The Brainpower Principle 65 Rural and social marketing notes Rodales Successful Organic Gardening Sea otters and nearshore benthic communities S.A. Levin Appendix : notes on Mississippi soldiers and politicians mentioned in the letters. Not Pretty, but Precious and Other Short Stories Mel Bay The Art of Hawaiian Steel Guitar Great realizations Notes for army medical officers Dastar nama of Khushhal Khan Khattak Handbook of Medical Psychiatry Mini gps tracker manual Rencontres 2 (Rencontres) Ceramic Powder Science (Advances in Ceramics, Vol 21) Poems. By Will M. Carleton. S media.metro.net riding_metro riders_guide images tap_sr_app. My fair concubine jeannie lin Teaching cues for sport skills for secondary school students VIII. SPECIAL PROBLEMS OF AVIATION MEDICINE IN EUROPE. . . 617 From Calcutta to the snowy range Colonial counterflow : from orientalism to Buddhism Mark Lussier The life of Sir William Petty, 1623-1687 The domains of art Rankin attachments price list The sinner by petra hammesfahr Porsche, Past and Present Boeije h 2010 analysis in qualitative research Fine structure analysis of genes Setting high standards for everyone Propaganda (Living History) 3.3 The Study Population ./t. 19 Animal farm lesson plans Elements of differential calculus The Insects and Arachnids of Canada: The Horseflies Deer Flies of Canada Alaska: Diptera Sex Manual Over 30 D. Biographical sketches.