FullStack.Cafe - Kill Your Tech Interview

Q1: What REST stands for? ☆

Topics: API Design REST & RESTful

Answer:

REST stands for *REpresentational State Transfer*. REST is web standards based architecture and uses HTTP Protocol for data communication. It revolves around resource where every component is a resource and a resource is accessed by a common interface using HTTP standard methods. REST was first introduced by Roy Fielding in 2000.

In REST architecture, a REST Server simply provides access to resources and REST client accesses and presents the resources. Here each resource is identified by URIs/ global IDs. REST uses various representations to represent a resource like text, JSON and XML. Now a days JSON is the most popular format being used in web services.

Q2: What is API Design? ☆

Topics: API Design

Answer:

API design is the process of building an intermediary interface for a system to system connection to expose data to application users and developers.

The fundamental API design influences how well users can consume it and the general user experience. This development process does not allow a single approach. Instead, it combines a series of guidelines to meet initial expectations and continue to work consistently. Application programming interface designers closely follow industry best practices, design patterns, API design principles, and user needs to develop software that presents excellent functionality.

Q3: What are the advantages of Web Services? ☆☆

Topics: API Design

Answer:

Some of the advantages of web services are:

- **Interoperability**: Web services are accessible over network and runs on HTTP/SOAP protocol and uses XML/JSON to transport data, hence it can be developed in any programming language. Web service can be written in java programming and client can be PHP and vice versa.
- **Reusability**: One web service can be used by many client applications at the same time.
- **Loose Coupling**: Web services client code is totally independent with server code, so we have achieved loose coupling in our application.
- **Easy to deploy and integrate**, just like web applications.
- **Multiple service versions** can be running at same time.

Q4: What are different types of Web Services? ☆☆

Topics: API Design

Answer:

There are two types of web services:

- **SOAP Web Services**: Runs on SOAP protocol and uses XML technology for sending data.
- **Restful Web Services**: It's an architectural style and runs on HTTP/HTTPS protocol almost all the time. REST is a stateless client-server architecture where web services are resources and can be identified by their URIs. Client applications can use HTTP GET/POST methods to invoke Restful web services.

Q5: What is SOAP? ☆☆

Topics: SOA API Design

Answer:

SOAP stands for Simple Object Access Protocol. SOAP is an XML based industry standard protocol for designing and developing web services. Since it's XML based, it's platform and language independent. So our server can be based on JAVA and client can be on .NET, PHP etc. and vice versa.

Q6: What are advantages of REST web services? ☆☆

Topics: API Design REST & RESTful

Answer:

Some of the advantages of REST web services are:

- Learning curve is easy since it works on HTTP protocol
- Supports multiple technologies for data transfer such as text, xml, json, image etc.
- No contract defined between server and client, so loosely coupled implementation.
- REST is a lightweight protocol
- REST methods can be tested easily over browser.

Q7: Mention whether you can use GET request instead of PUT to create a resource? ☆☆

Topics: API Design

Answer:

No, you are not supposed to use POST or GET. GET operations should only have view rights

Q8: Mention what are resources in a REST architecture? ☆☆

Topics: API Design REST & RESTful

Answer:

Resources are identified by logical URLs; it is the key element of a RESTful design. Unlike, SOAP web services in REST, you view the product data as a resource and this resource should contain all the required information.

Q9: Mention some key characteristics of REST? ☆☆

Topics: API Design REST & RESTful

Answer:

Some key characteristics of REST includes

- REST is stateless, therefore the SERVER has no state (or session data)
- With a well-applied REST API, the server could be restarted between two calls as every data is passed to the server
- Web service mostly uses POST method to make operations, whereas REST uses GET to access resources

Q10: What are the core components of a HTTP Request? ☆☆

Topics: API Design

Answer:

A HTTP Request has five major parts −

- **Verb** − Indicate HTTP methods such as GET, POST, DELETE, PUT etc.
- **URI** − Uniform Resource Identifier (URI) to identify the resource on server.
- **HTTP Version** − Indicate HTTP version, for example HTTP v1.1 .
- **Request Header** − Contains metadata for the HTTP Request message as key-value pairs. For example, client (or browser) type, format supported by client, format of message body, cache settings etc.
- **Request Body** − Message content or Resource representation.

Q11: What is cached response? ☆☆

Topics: API Design

Answer:

Caching refers to storing server response in client itself so that a client needs not to make server request for same resource again and again. A server response should have information about how a caching is to be done so that a client caches response for a period of time or never caches the server response.

Q12: Define what is SOA ☆☆

Topics: SOA API Design

Answer:

A Service Oriented Architecture (SOA) is basically defined as an architectural pattern consisting of services. Here application components provide services to the other components using communication protocol over the network. This communication involves data exchanging or some coordination activity between services.

Some of the key principles on which SOA is based are mentioned below

- The service contract should be standardized containing all the description of the services.
- There is loose coupling defining the less dependency between the web services and the client.
- It should follow Service Abstraction rule, which says the service should not expose the way functionality has been executed to the client application.
- Services should be reusable in order to work with various application types.
- Services should be stateless having the feature of discoverability.
- Services break big problems into little problems and allow diverse subscribers to use the services.

Q13: What is the use of Accept and Content-Type Headers in HTTP Request? ☆☆☆

Topics: API Design

Answer:

- **Accept headers** tells web service what kind of response client is accepting, so if a web service is capable of sending response in XML and JSON format and client sends Accept header as `application/xml` then XML response will be sent. For Accept header `application/json`, server will send the JSON response.

- **Content-Type header** is used to tell server what is the format of data being sent in the request. If Content-Type header is `application/xml` then server will try to parse it as XML data. This header is useful in HTTP Post and Put requests.

Q14: Mention what is the difference between PUT and POST? ☆☆☆

Topics: API Design

Answer:

PUT puts a file or resource at a particular URI and exactly at that URI. If there is already a file or resource at that URI, PUT changes that file or resource. If there is no resource or file there, PUT makes one

POST sends data to a particular URI and expects the resource at that URI to deal with the request. The web server at this point can decide what to do with the data in the context of specified resource

PUT is idempotent meaning, invoking it any number of times will not have an impact on resources.

However, *POST is not idempotent*, meaning if you invoke POST multiple times it keeps creating more resources

Q15: Mention what is the difference between RPC or document style web services? How you determine to which one to choose? ☆☆☆

Topics: API Design

Answer:

In document style web services, we can transport an XML message as part of SOAP request which is not possible in RPC style web service. Document style web service is most appropriate in some application where XML message behaves as document and content of that document can alter and intention of web service does not rely on the content of XML message.

Q16: What are the best practices to create a standard URI for a web service? ☆☆☆

Topics: API Design

Answer:

Following are important points to be considered while designing a URI:

- **Use Plural Noun** − Use plural noun to define resources. For example, we've used users to identify users as a resource.
- **Avoid using spaces** − Use underscore(_) or hyphen(-) when using a long resource name, for example, use authorized_users instead of authorized%20users.
- **Use lowercase letters** − Although URI is case-insensitive, it is good practice to keep url in lower case letters only.
- **Maintain Backward Compatibility** − As Web Service is a public service, a URI once made public should always be available. In case, URI gets updated, redirect the older URI to new URI using HTTP Status code, 300.
- **Use HTTP Verb** − Always use HTTP Verb like GET, PUT, and DELETE to do the operations on the resource. It is not good to use operations names in URI.

Q17: What is statelessness in RESTful Webservices? ☆☆☆

Topics: API Design REST & RESTful

Answer:

As per REST architecture, a RESTful web service should not keep a client state on server. This restriction is called *statelessness*. It is responsibility of the client to pass its context to server and then server can store this context to process client's further request. For example, session maintained by server is identified by session identifier passed by the client.

Q18: What is Payload? ☆☆☆

Topics: API Design

Answer:

The request data which is present in the body part of every HTTP message is referred as *Payload*. In Restful web service, the payload can only be passed to the recipient through POST method.

There is no limit of sending data as payload through POST method but the only concern is that more data with consuming more time and bandwidth. This may consume much of user’s time also.

Q19: What's the difference between REST & RESTful? ☆☆☆

Topics: API Design REST & RESTful

Answer:

- Representational state transfer (REST) is a style of software architecture. As described in a dissertation by Roy Fielding, REST is an "architectural style" that basically exploits the existing technology and protocols of the Web.

- RESTful is typically used to refer to web services implementing such an architecture.

Q20: WebSockets vs Rest API for real time data? Which to choose?

☆☆☆

Topics: API Design Software Architecture WebSockets REST & RESTful

Problem:

I need to constantly access a server to get real time data of financial instruments. The price is constantly changing so I need to request new prices every 0.5 seconds. Which kind if API would you recommend?

Solution:

The most efficient operation for what you're describing would be to use a webSocket connection between client and server and have the server send updated price information directly to the client over the webSocket ONLY when the price changes by some meaningful amount or when some minimum amount of time has elapsed and the price has changed.

Here's a comparison of the networking operations involved in sending a price change over an already open webSocket vs. making a REST call.

webSocket

1. Server sees that a price has changed and immediately sends a message to each client.
2. Client receives the message about new price.

Rest/Ajax

1. Client sets up a polling interval
2. Upon next polling interval trigger, client creates socket connection to server
3. Server receives request to open new socket
4. When connection is made with the server, client sends request for new pricing info to server
5. Server receives request for new pricing info and sends reply with new data (if any).
6. Client receives new pricing data
7. Client closes socket
8. Server receives socket close

As you can see there's a lot more going on in the Rest/Ajax call from a networking point of view because a new connection has to be established for every new call whereas the webSocket uses an already open call. In addition, in the webSocket cases, the server just sends the client new data when new data is available - the client doens't have to regularly request it.

A webSocket can also be faster and easier on your networking infrastructure simply because fewer network operations are involved to simply send a packet over an already open webSocket connection versus creating a new connection for each REST/Ajax call, sending new data, then closing the connection. How much of a difference/improvement this makes in your particular application would be something you'd have to measure to really know.

FullStack.Cafe - Kill Your Tech Interview

Q1: What are disadvantages of SOAP Web Services? ☆☆☆

Topics: SOA API Design

Answer:

Some of the disadvantages of SOAP protocol are:

- Only XML can be used, JSON and other lightweight formats are not supported.
- SOAP is based on the contract, so there is a tight coupling between client and server applications.
- SOAP is slow because payload is large for a simple string message, since it uses XML format.
- Anytime there is change in the server side contract, client stub classes need to be generated again. Can't be tested easily in browser

Q2: What is UDDI? ☆☆☆

Topics: SOA API Design

Answer:

UDDI is acronym for *Universal Description, Discovery and Integration*. UDDI is a directory of web services where client applications can lookup for web services. Web Services can register to the UDDI server and make them available to client applications.

Q3: What are disadvantages of REST web services? ☆☆☆

Topics: API Design REST & RESTful

Answer:

Some of the disadvantages of REST are:

- Since there is no contract defined between service and client, it has to be communicated through other means such as documentation or emails.
- Since it works on HTTP, there can't be asynchronous calls.
- Sessions can't be maintained.

Q4: What are different ways to test web services? ☆☆☆

Topics: API Design

Answer:

- SOAP web services can be tested programmatically by generating client stubs from WSDL or through software such as Soap UI.

- REST web services can be tested easily with program, `curl` commands and through browser extensions. Resources supporting GET method can be tested with browser itself, without any program.

Q5: How would you choose between SOAP and REST web services? ☆☆☆

Topics: SOA API Design

Answer:

Web Services work on client-server model and when it comes to choose between SOAP and REST, it all depends on project requirements. Let's look at some of the conditions affecting our choice:

- Do you know your web service clients beforehand? If Yes, then you can define a contract before implementation and SOAP seems better choice. But if you don't then REST seems better choice because you can provide sample request/response and test cases easily for client applications to use later on.
- How much time you have? For quick implementation REST is the best choice. You can create web service easily, test it through browser/curl and get ready for your clients. What kind of data format are supported? If only XML then you can go with SOAP but if you think about supporting JSON also in future then go with REST.

Q6: Mention what are the different application integration styles? ☆☆☆

Topics: API Design

Answer:

The different integration styles include

- Shared database
- Batch file transfer
- Invoking remote procedure (RPC)
- Swapping asynchronous messages over a message oriented middle-ware (MOM)

Q7: What are the best practices to design a resource representation? ☆☆☆

Topics: API Design

Answer:

Following are important points to be considered while designing a representation format of a resource in a RESTful web services −

- **Understandability** − Both Server and Client should be able to understand and utilize the representation format of the resource.
- **Completeness** − Format should be able to represent a resource completely. For example, a resource can contain another resource. Format should be able to represent simple as well as complex structures of resources.
- **Linkablity** − A resource can have a linkage to another resource, a format should be able to handles such situations.

Q8: What are the core components of a HTTP response? ☆☆☆

Topics: API Design

Answer:

A HTTP Response has four major parts −

- **Status/Response Code** − Indicate Server status for the requested resource. For example 404 means resource not found and 200 means response is ok.
- **HTTP Version** − Indicate HTTP version, for example HTTP v1.1 .
- **Response Header** − Contains metadata for the HTTP Response message as key-value pairs. For example, content length, content type, response date, server type etc.
- **Response Body** − Response message content or Resource representation.

Q9: What are the disadvantages of statelessness in RESTful Webservices? ☆☆☆

Topics: API Design REST & RESTful

Answer:

Following is the disadvantage of statelessness in RESTful web services:

- Web services need to get extra information in each request and then interpret to get the client's state in case client interactions are to be taken care of.

Q10: What are the best practices for caching? ☆☆☆

Topics: API Design

Answer:

Always keep static contents like images, css, JavaScript cacheable, with expiration date of 2 to 3 days. Never keep expiry date too high.

Dynamic contents should be cached for few hours only.

Q11: What is the purpose of HTTP Status Code? ☆☆☆

Topics: API Design

Answer:

HTTP Status code are standard codes and refers to predefined status of task done at server. For example, HTTP Status 404 states that requested resource is not present on server.

Consider following status codes:

- 200 - OK, shows success.
- **201** - CREATED, when a resource is successful created using POST or PUT request. Return link to newly created resource using location header.
- **304** - NOT MODIFIED, used to reduce network bandwidth usage in case of conditional GET requests. Response body should be empty. Headers should have date, location etc.
- 400 - BAD REQUEST, states that invalid input is provided e.g. validation error, missing data.
- **401** - FORBIDDEN, states that user is not having access to method being used for example, delete access without admin rights.
- 404 - NOT FOUND, states that method is not available.

- **409** - CONFLICT, states conflict situation while executing the method for example, adding duplicate entry.
- 500 - INTERNAL SERVER ERROR, states that server has thrown some exception while executing the method.

Q12: What are the primary security issues of web service? ☆☆☆

Topics: API Design

Answer:

To ensure reliable transactions and secure confidential information, web services requires very high level of security which can be only achieved through *Entrust Secure Transaction Platform.* Security issues for web services are broadly divided into three sections as described below

1) Confidentiality: A single web service can have multiple applications and their service path contains a potential weak link at its nodes. Whenever messages or say XML requests are sent by the client along with the service path to the server, they must be encrypted. Thus, maintaining the confidentiality of the communication is a must.

2) Authentication: Authentication is basically performed to verify the identity of the users as well as ensuring that the user using the web service has the right to use or not? Authentication is also done to track user's activity. There are several options that can be considered for this purpose

- Application level authentication
- HTTP digest and HTTP basic authentication
- Client certificates

3) Network Security: This is a serious issue which requires tools to filter web service traffic.

Q13: What is difference between SOA and Web Services? ☆☆☆☆

Topics: SOA API Design

Answer:

Service Oriented Architecture (SOA) is an architectural pattern where applications are designed in terms of services that can be accessed through communication protocol over network. SOA is a design pattern and doesn't go into implementation.

Web Services can be thought of as Services in SOAP architecture and providing means to implement SOA pattern.

Q14: What are the advantages of statelessness in RESTful Webservices? ☆☆☆☆

Topics: API Design REST & RESTful

Answer:

Following are the benefits of statelessness in RESTful web services:

- Web services can treat each method request independently.
- Web services need not to maintain client's previous interactions. It simplifies application design.
- As HTTP is itself a statelessness protocol, RESTful Web services work seamlessly with HTTP protocol.

Q15: What do you mean by idempotent operation? ☆☆☆☆

Topics: API Design

Answer:

Idempotent operations means their result will always same no matter how many times these operations are invoked.

Q16: Which type of Webservices methods are to be idempotent? ☆☆☆☆

Topics: API Design

Answer:

PUT and DELETE operations are idempotent

Q17: Which header of HTTP response provides control over caching? ☆☆☆☆

Topics: API Design

Answer:

`Cache-Control` is the primary header to control caching.

Q18: Explain Cache-control header ☆☆☆☆

Topics: API Design

Answer:

A standard Cache control header can help in attaining cache ability. Enlisted below is the brief description of various cache control header:

- **Public:** Resources that are marked as the public can be cached by any intermediate components between the client and server.
- **Private:** Resources that are marked as private can only be cached by the client.
- **No cache** means that particular resource cannot be cached and thus the whole process is stopped.

Q19: Explain element? ☆☆☆☆

Topics: API Design

Answer:

Definition element is described as the root of WSDL document which defines the name of the web service as well as act as a container for all the other elements.

Q20: Explain what is the API Gateway pattern ☆☆☆☆

Topics: Microservices API Design

Answer:

An **API Gateway** is a server that is the single entry point into the system. It is similar to the Facade pattern from object-oriented design. The API Gateway encapsulates the internal system architecture and provides an API that is tailored to each client. It might have other responsibilities such as authentication, monitoring, load balancing, caching, request shaping and management, and static response handling.

A major benefit of using an API Gateway is that it encapsulates the internal structure of the application. Rather than having to invoke specific services, clients simply talk to the gateway.

FullStack.Cafe - Kill Your Tech Interview

Q1: What are the best practices to be followed while designing a secure RESTful web service? ☆☆☆☆☆

Topics: API Design REST & RESTful

Answer:

As RESTful web services work with HTTP URLs Paths so it is very important to safeguard a RESTful web service in the same manner as a website is be secured. Following are the best practices to be followed while designing a RESTful web service:

- **Validation** − Validate all inputs on the server. Protect your server against SQL or NoSQL injection attacks.
- **Session based authentication** − Use session based authentication to authenticate a user whenever a request is made to a Web Service method.
- **No sensitive data in URL** − Never use username, password or session token in URL , these values should be passed to Web Service via POST method.
- **Restriction on Method execution** − Allow restricted use of methods like GET, POST, DELETE. GET method should not be able to delete data.
- **Validate Malformed XML/JSON** − Check for well formed input passed to a web service method.
- **Throw generic Error Messages** − A web service method should use HTTP error messages like 403 to show access forbidden etc.

Q2: Enlist some important constraints for RESTful web services ☆☆☆☆☆

Topics: API Design REST & RESTful

Answer:

Every constraint has positive as well as negative impacts and to produce an overall architecture, there should be the balance between both of them. Below mentioned are some important constraints for RESTful web service:

- There should be separate concerns for each server and client which will help to maintain the modularity within the application. This will also reduce the complexity and increase the scalability.
- The client-server communication should be stateless, which means no previous information is used and the complete execution is done in isolation. In cases of failure, it also helps the client to recover.
- In client-server communication, the HTTP response should be cacheable so that when required cached copy can be used which in turn enhances the scalability and performance of the server.
- The fourth constraint is the uniform interface which allows client-server interaction to be easily understood. This constraint is further divided into four sub-constraints as:
  - Resource Identification
  - Resource manipulation
  - Each message is easily understood and is self-descriptive.
  - Hypermedia, which is defined as the text with hyperlinks and when clicked it moves to another application state.
- Client-server communication should be done on a layered system and thus the client should only have knowledge about the intermediate level with which communication is being done.

Q3: What is Open API Initiative? ☆☆☆☆☆

Topics: API Design

Answer:

The **Open API Initiative** was created by an industry consortium to standardize REST API descriptions across vendors. As part of this initiative, the Swagger 2.0 specification was renamed the OpenAPI Specification (OAS) and brought under the Open API Initiative.

You may want to adopt OpenAPI for your web APIs. Some points to consider:

- The OpenAPI Specification comes with a set of opinionated guidelines on how a REST API should be designed. That has advantages for interoperability, but requires more care when designing your API to conform to the specification.
- OpenAPI promotes a contract-first approach, rather than an implementation-first approach. Contract-first means you design the API contract (the interface) first and then write code that implements the contract.
- Tools like Swagger can generate client libraries or documentation from API contracts. For example, see ASP.NET Web API Help Pages using Swagger.

Q4: Name some best practices for better RESTful API design ☆☆☆☆☆

Topics: API Design REST & RESTful

Answer:

- Use nouns and HTTP methods but no verbs

```
GET /cars
POST /cars
DELETE /cars/:id
instead
GET /getAllCars
POST /createNewCar
GET /deleteAllRedCars
```

- GET method and query parameters should not alter the state
- Use plural nouns

```
/cars instead of /car
/users instead of /user
/products instead of /product
/settings instead of /setting
```

- Use sub-resources for relations

```
GET /cars/711/drivers/ Returns a list of drivers for car 711
GET /cars/711/drivers/4 Returns driver #4 for car 711
```

- Use HTTP headers for serialisation formats

```
Content-Type defines the request format.
Accept defines a list of acceptable response formats.
```

- Use **HATEOAS** - Hypermedia as the Engine of Application State is a principle that hypertext links should be used to create a better navigation through the API.

```
{
  "id": 711,
  "manufacturer": "bmw",
  "model": "X5",
  "seats": 5,
  "drivers": [{
      "id": "23",
      "name": "Stefan Jauker",
      "links": [{
          "rel": "self",
          "href": "/api/v1/drivers/23"
      }]
  }]
}
```

- Use appropriate HTTP response status codes
- **2xx** (Success category)
- **3xx** (Redirection Category)
- **4xx** (Client Error Category)
- **5xx** (Server Error Category)
- Provide filtering, sorting, field selection and paging for collections

```
GET /cars?color=red Returns a list of red cars
GET /cars?seats<=2 Returns a list of cars with a maximum of 2 seats
```

- Version your API

```
/blog/api/v1
```

- Use error payloads - All exceptions should be mapped in an error payload.

```
{
  "errors": [{
      "userMessage": "Sorry, the requested resource does not exist",
      "internalMessage": "No car found in the database",
      "code": 34,
      "more info": "http://dev.mwaysolutions.com/blog/api/v1/errors/12345"
  }]
}
```

- Allow overriding HTTP method

  In certain situations (for example, when the service or its consumers are behind an overzealous corporate firewall, or if the main consumer is a web page), only the `GET` and `POST` HTTP methods might be available.

In such a case, it is possible to emulate the missing verbs by passing a custom header in the requests. To support a RESTful API with these limitations, the API needs a way to override the HTTP method and use the custom HTTP Header `X-HTTP-Method-Override` to map the request to an appropriate API method.

Q5: Explain the difference between WCF, Web API, WCF REST and Web Service? ☆☆☆☆☆

Topics: API Design ASP.NET Web API REST & RESTful

Answer:

The .Net framework has a number of technologies that allow you to create HTTP services such as Web Service, WCF and now Web API. There are a lot of articles over the internet which may describe to whom you should use.

Web Service

- It is based on SOAP and return data in XML form.
- It support only HTTP protocol.
- It is not open source but can be consumed by any client that understands xml.
- It can be hosted only on IIS.

WCF

- It is also based on SOAP and return data in XML form.
- It is the evolution of the web service(ASMX) and support various protocols like TCP, HTTP, HTTPS, Named Pipes, MSMQ.
- The main issue with WCF is, its tedious and extensive configuration.
- It is not open source but can be consumed by any client that understands xml.
- It can be hosted with in the applicaion or on IIS or using window service.

WCF Rest

- To use WCF as WCF Rest service you have to enable webHttpBindings.
- It support HTTP GET and POST verbs by [WebGet] and [WebInvoke] attributes respectively.
- To enable other HTTP verbs you have to do some configuration in IIS to accept request of that particular verb on .svc files
- Passing data through parameters using a WebGet needs configuration. The UriTemplate must be specified.
- It support XML, JSON and ATOM data format.

Web API

- This is the new framework for building HTTP services with easy and simple way.
- Web API is open source an ideal platform for building REST-ful services over the .NET Framework.
- Unlike WCF Rest service, it use the full feature of HTTP (like URIs, request/response headers, caching, versioning, various content formats)
- It also supports the MVC features such as routing, controllers, action results, filter, model binders, IOC container or dependency injection, unit testing

Q6: What is difference between OData and REST web services? ☆☆☆☆☆

Topics: API Design REST & RESTful

Answer:

- REST - is an architecture of how to send messages over HTTP.
- OData (v4) - is a specific implementation of REST, really defines the content of the messages in different formats (currently I think is AtomPub and JSON). ODataV4 follows rest principles.

For example, asp.net people will mostly use WebApi controller to serialize/deserialize objects into JSON and have javascript do something with it. The point of Odata is being able to query directly from the URL with out-of-the-box options.