Evaluating REST architectures

Telecom SudParis 2019-2020: CCN

Presented by: KASHAMA Katanga Wa Muila, Yerik KASSYM
Outlines:

1. Terminology
2. **What gives us REST** and When to use?
3. Difference between Traditional HTTP, SOAP and REST Style
4. Main REST architecture style
5. Foundations(Constraints) of REST architectural Style
6. Client-Server constraint
7. Cacheable REST constraint
8. Types of cache
9. Difference between Stateful & Stateless
10. Uniform Interface constraint
11. Code on Demand constraint
12. Layered System constraint
13. REST Quality Attributes
14. Main REST Quality Attributes
15. Challenges of REST (Disadvantages)
16. Conclusion of challenges REST
17. Conclusion
18. References
1. Terminology

**REpresentational**: The resource (image, page, video, profile) is represented by the web server to the client in any format like HTML, Image, JSON, XML etc.

**State**: The state of the application (web site) on a client's computer changes as the client clicks from one link to the next. Ask the client clicks on the link, they request additional resources, and the application "state" changes.

**Transfer**: The transfer of resources from the web server to the client in a "representational" state which can be read by the client or implemented in the application program by the programmer. The transfer may also refer to the application state transfer as the client browses a web site.
2. What gives us REST and When to use?

- Scalability of interaction of system components (applications)
- Interface Commonness
- Independent implementation of components

**When to use**

- When there is a connection bandwidth limit
- If you need to cache requests
- If the system involves significant scaling
- In services using AJAX
3. Difference between Traditional HTTP, SOAP and REST Style

Different styles to implement Web Services

**Hypertext Transfer Protocol**
- is an application protocol which is used for distributed, collaborative and hypermedia information systems.
- widely regarded as the foundation of data communication for the World Wide Web.

**Simple Object Access Protocol**
- It is a protocol based on messages that is used to implement the messages layer of a Web Service
- Strongly typed proxies – WSDL;
- Supports HTTP/TCP/MSMQ/Named Pipes

**Representational state transfer**
- Services act as Resources, such as images or HTML documents
- Only works over HTTP, and REST calls are restricted to HTTP verbs (GET, POST, PUT, DELETE, etc.).
4. Main REST architecture style

(Example of simple interaction between server and client)

Client (web browser) -> Request (HTTP) -> Web Server -> Response (html, css...)

(Example of REST interaction between server and client)

Client (web browser) <-> data (json) <-> REST Server <-> data (json) <-> App (iOS Adr)
5. Foundations (Constraints) of REST architectural Style

REST architectural style defines six constraints which make any web services a RESTful API which are:
6. Client-Server constraint

- Client is a machine that we use to make request through the web
- A server is a program that listens to requests and responds to them
- The client-server model is a centralized web architecture that classify computers into two sections: requests and response provides
- And finally we have an alternative which is peer-to-peer model: all computers are functionally equal
7. Cacheable REST constraint

- Improve performance
- Reduces the number of requests from client: By expiration
- Reduces network bandwidth requirement: By validation

Two models:

1. Expiration
2. Validation

Three types of Cache:

- Private or client cache
- Gateway cache or server cache
- Proxy cache
8. Types of cache

**Private or client cache**

**Gateway cache or server cache**

**Proxy cache**
9. Difference between Stateful & Stateless

**Stateful:** Session is retained

- **Protocol:** TCP
  - Request
  - Response
  - Web Server 1
  - Stateful Session: ON

- **Protocol:** IP, HTTP
  - Request
  - Response
  - Web Server 2
  - Stateless Session: OFF
9.1 Difference between Stateful & Stateless

Stateless: Session is not retained

All the required infos to understand a request are needed within that request.
10. Uniform Interface constraint

- RESTful architectures must have **uniform interface** between all **clients** and **servers**
- **Server** must not require different ways of accessing data
- Gaining access to endpoints is the same for any machine trying to access to the information
Layered System. In REST, it is allowed to divide the system into a hierarchy of layers, but with the condition that each component can see the components of only the next layer itself.
12. Code on Demand constraint

Servers can temporarily extend or customize the functionality of a client by transferring executable code: for example, compiled components such as Java applets, or client-side scripts such as JavaScript.
13. REST Quality Attributes

General software quality attributes

**Functional QA**

- Maintainability
- Extensibility
- Testability
- Reliability
- Efficiency
- Robustness

**Non-Functional QA**

- Interoperability
- Easibility
- On time
- Readability
- Usability
- Within Budget
14. Main REST Quality Attributes
15. Challenges of REST (Disadvantages)

1. Decentralized authorities
2. Distributed server (still remains a challenge to be solved)
3. Statelessness and Isolation
4. REST calls are restricted to HTTP verbs (GET, POST, PUT, DELETE, etc.).
The main conclusion of the analysis of the existing RESTful transaction models is that *one model does not fit all*. RESTful transaction scenarios are diverse in many dimensions and no transaction model fulfills the requirements of every scenario. On the contrary, these models are designed to cover specific scenarios. However, there are still some scenarios that are not sufficiently supported by the current models.
17. Conclusion

- REST architectural pattern is basically lightweight in nature. So, when you have bandwidth constraints then prefer REST web service.
- Easy and fast to develop.
- Top sites like Twitter, Yahoo uses this pattern and most social networking sites like facebook.com uses REST web services.
- Mobile App development growing rapidly and for their server interaction, it uses this REST pattern as it is faster in processing request/response data.
18. References

2. https://jegatech.wordpress.com/2012/10/18/soap-vs-rest/
Questions
www.google.com/image/thanking