Jonathan Roland

WEB 420 RESTful APIs

Instructor: Prof. Krasso

11/20/2020

Discussion 6.1

The seven step design procedure is uses known project requirements and outlines how they will be converted into an API.

1. **List the Semantic Descriptors**

This is the process of listing all information a client will want to receive or send to your API. Hierarchies and data objects will present themselves at this stage. One organization technique to highlight these structures is to draw boxes around each distinct level of data.

1. **Draw a State Diagram**

As we have learned previously, state diagrams describe the flow of the application state. Using the representations generated by step 1, we can lay out the flow of the application through intuition and logic. Boxes that contain other boxes hint to data relationships and links.

1. **Reconcile Names**

This is the part where you give human readable names to all your magic strings. You can of course create your own, but consider that there is a vast sea of profiles available to match all or part of your API’s needs. By re-using registered IANA profiles, the client has a shorter path to understanding the options it is presented with.

1. **Choose a Media Type**

Examine your state diagram and protocol semantics and choose how you want to represent them. To inform this choice, consider if your application resembles any common patterns like Collection+JSON. If it does not neatly fit an existing pattern, then a more generic hypermedia language like HTML is probably the best bet.

1. **Write a Profile**

While you may be able to match your application to an existing profile, any custom profiles need to be documented as well. You will need to include the correct Content-Type header to inform your clients parsing needs, but you will also need to include links to both externally referenced and internally customized profiles.

1. **Implementation**

Quite simply, this is the development phase. Be sure to do steps 1-5 thoroughly before you begin implementation.

1. **Publication**

Now that you have a shiny new API, people need to know where to find it and where to read documentation. Communicate to your users your URL, and provide hosting and access to your documentation (including machine readable documentation). If you have created a new profile, consider registering it with IANA, ALPS, or anything else that applies.

Publication is rarely the last step. As we used to say in the Marines, no plan survives contact with the enemy. Users certainly are not your enemy, but design oversights, project creep, and shifting external standards and markets can be. You will very likely need to make adjustments and changes to your API at some point. For many companies, this presents an instant problem because their clients have developed infrastructure that expects version 1.0 exactly. Moving forward with changes on your own will alienate at least some users.

In my opinion, partitioning your URL from the start is an excellent way to address this problem. A client that knows and is happy with someApi/v1 will never prevent you from publishing someApi/v2 and /v3. However, a client that wishes to adapt to and use new features can read your documentation and choose to communicate with /v2 et cetera as they like. This is a simple option up front that allows the publisher freedom of growth without alienating legacy version clients. If you choose this approach, be sure to be descriptive and visible wherever you are billboarding your API.

Reference:

Richardson, L., Amundsen, M., & Ruby, S. (2013). *RESTful Web APIs: Services for a Changing World* (1st ed.). O’Reilly Media.