iTrust as a RESTful api

Kevin Price - kaprice4@ncsu.edu

Guoyang Di - gdi@ncsu.edu

Kwan Ho Tako Cheung - kcheung2@ncsu.edu

Aoyi Li - ali4@ncsu.edu

There are many tools in the great toolbox that is Software Engineering. One of these tools, REST, is a software architectural style that has grown in popularity since the introduction of SOAP. The REST style is built upon six constraints that must hold in order for a system to be considered RESTful. The following constraints are as follows: Uniform Interface, Stateless, Cacheable, Client-Server, Layered System, and Code on Demand [1]. All of these constraints put together provide a web service that is: Lightweight, Human Readable, and Easier to build. Whilst some of the drawbacks include: Point-to-Point communication, Lack of standards, and code that is Tied to HTTP [2]. The real question here is how iTrust relates to REST. Is iTrust a RESTful API? Why or why not? And if it's not, how can it be redesigned to be RESTful? Some of the constraints of REST are fairly straightforward: Client-Server, Uniform Interface, Code-on-demand, and Cacheable; these will be briefly mentioned. The more in-depth parts of REST in relation to iTrust: Layered System, and Stateless, will be talked about more.

Some of the restraints of REST are that a system must be: Client-Server, Uniform Interface, and Cacheable. iTrust is Client-Server. The client is coded via JSP, and sends requests to a SQL server. Responses and requests are sent via java "beans". iTrust is designed with MVC, which decouples User Interface and the Logic of the System; in this way, iTrust is implemented as Client-Server. iTrust has a Uniform Interface. A Uniform Interface has four constraints: Resource-Based, Manipulation of Resources through Representations, Self-Descriptive Messages, and Hypermedia as the Engine of Application State [1]. Because iTrust takes URI requests and sends responses back in XML, iTrust is a uniform interface. Any client can interact with iTrust, and get the same XML from the server. iTrust has code-on-demand with its use of JSP. JSP gives iTrust the ability to provide a consistent User Interface while simultaneously

having Java code processed on the server. Finally, iTrust is Cacheable. On the back-end of iTrust, temporary java objects are stored for later use, which reduces the need to continuously update, and therefore, request data from the Database. This gives iTrust a performance increase, and simplifies the code. These, however are not the only constraints for iTrust to be a RESTful system.

In order to become a full RESTful system, iTrust must meet the last two constraints of a RESTful api. That is, iTrust must be a Layered System, and iTrust must also be Stateless. iTrust meets the Layered System with ease. This is due to the fact that iTrust was designed around the MVC design pattern. iTrust provides a UI, built on JSP, a Controller, and a Model class built with java and SQL. In this way, the front-end and back-end of iTrust are broken up. Clients cannot tell whether they are connected to the end server or an intermediary. The multiple layers provide better design, as well as more security to users. Therefore, iTrust meets the requirements of a Layered-System. There is only one more constraint for iTrust to meet. If iTrust can meet the Stateless constraint, it will be a fully RESTful api.

Unfortunately, iTrust is not a RESTful api. iTrust is not stateless. iTrust changes state all the time because iTrust uses session variables to maintain the state at any given time. If iTrust was to become RESTful, it would have to be redesigned without the use of session variables. In which all requests would include the state to handle that request [1]. Most commonly, the URI is used to communicate the state during a request. Therefore, iTrust is not RESTful.

[1]P. Todd Fredrich, 'What is REST?', *Restapitutorial.com*, 2015. [Online]. Available: http://www.restapitutorial.com/lessons/whatisrest.html#. [Accessed: 08- Nov- 2015].

[2]D. Ogozalek, 'REST vs. SOAP | Business Integration Blog', *Extol.com*, 2009. [Online]. Available: http://www.extol.com/blog/?p=250. [Accessed: 08- Nov- 2015].