Project: RESTful Elaboration

Story: RESTful Services Spike

As an Architect

I want to: Understand the RESTful Web Services pattern

So that I can understand how it could benefit a software solutions provider

---

Project Goals:

The whole point of the project is to answer the following questions:

- ~~Is REST really easier than SOAP/WSDL~~
- ~~How should we Build RESTful services (Maven2)?~~
- ~~How should we Test RESTful services(Jersey JUnit Test, embedded, selenium)?~~
- ~~How should we automate testing in CI (Hudson?)?~~
- ~~How should we Marshall/Unmarshall XML objects (JaxB?)?~~
- ~~How should we Persist resource data (JPA from JEE6)?~~
- ~~How can we control the data definitions (XSD?)~~
- ~~How do we Create, Read, Update & Delete Item resources (CRUD)~~
- What is WADL, how is it used, and is it necessary?
- Can we apply the same SOA Design Principals when using REST?
- What difference do we think using REST could make to our velocity?
- Is there an equivalent to the handler-chain for message validation in rest?
- How can we support service versioning?
- How do we make REST services discoverable?

Acceptance Criteria:

~~Test Driven Development~~
~~RESTful Web service created for 'Items' resource (AKA Product)~~
Complies with goals and principals of RESTful Web Services

Assumptions & Constraints:

~~Stick to Application/XML mimetype for Consumes and Produces~~
~~Use JPA for persistence~~

~~Use non-proprietary open source tools (eclipse, maven2, Glassfish, JEE, Hypersonic).~~
Plan as an agile project (VersionOne)
~~Keep it simple and short~~
~~Treat Client as unimportant (but building the service and testing it is)?~~
~~Version Controlled (Unfuddle)~~
~~Locally hosted only - no need for online servers or infrastructure~~
~~Locally Built only - No CI required~~

Project Approach:

1. Baby steps oriented project (small success's that build on each other)
2. Small 30min tasks wherever possible
3. Planned with Releases containing 2 Iterations
4. Vertically sliced where practical (but within 2)
5. Keep project requirements simple & short (small stories & limit the backlog)
6. Collaborative (use an open source approach, many collaborators)

Item attributes:

We'll keep the task simple by limiting the Item definition to a simple XSD containing...

- ~~Id~~
- Name
- ~~Description~~
- ImageName
- LastUpdated (Date)
- URI
- ~~Price~~
- ~~Cost~~
- Optional attributes
- At least one Array like attribute

I'd like to embrace RESTful practices and get a thorough understaning of what a Resource Oriented Architecture (ROA) would look like.

Getting Started

To get started with builds and tests...

1. Install Maven (check *mvn -version* works on the cmdline)
2. Install TortoiseSVN
3. Checkout project with Tortoise from the Unfuddle SVN repository (http://restfulitemservice.unfuddle.com/svn/restfulitemservice_svnrepo/)
4. Navigate to the folder with the project's pom.xml file in it in the command line window and type ***mvn clean test*** (the project should download all the dependencies build all the code and run all the JUnit tests successfully)

To Build/Deploy/Test/IntegrationTest the service on an embedded Glassfish server

1. type ***mvn clean verify*** into the cmd window

To work on the code...

1. Install glassfish tools bundle for eclipse
2. type *mvn clean eclipse:clean* to clean the project and remove any old eclipse settings ready for importing
3. type *mvn eclipse:eclipse* (sets up the project for importing into eclipse)
4. In Eclipse import the project (project type is 'maven' (under general))
5. You can now work on the code, execute tests etc.
6. Notice the right click 'Maven' context item in the project browser, and maven targets under 'Run As'