JSR 363

Third meeting

Agenda

- Spec doc: review, comments and changes
- Issue tracking (JAVA.NET or GITHUB?)
- RI and SE impl: items that need pruning, or how close are we to the final implementation
- Anatole's blog post about JSR363
- Public participation/hackatons

Spec Document

- Current version is located at https://github.com/unitsofmeasurement/unit-api/blob/master/src/main/asciidoc/jsr363.adoc
- Make it a PDF available to download at java.net?
- Continue to use asciidoc?
- Do we have a clear understanding about what are going to be in the RI, what already is and what are the extension points?

Comments

- Martin raised concern about including Measurement, Jean-Marie agreed, mentioned JSR275 problems
- Martin raised the current Measurement doesn’t work very well
- Werner suggested keeping Measurement simple, moving operations from Measurement to Quantity
- Martin agrees on keeping arithmetic in Quantity
- Martin suggests that Measurement only have getUnit()

Comments

- Martin and Werner to produce a link and html page for current Spec Doc
- we should raise a Jira issue about spec doc <> RI differences

Issue Tracking

- JIRA or GITHUB?
- We have open tickets in both, and to get better participation, we need to focus. So, let’s focus on JIRA?

Comments

- Keep everything in Jira
- Try and shutdown issue tracking in GitHub
- Otávio raised that Github would be best, Leo asked to try at least going Jira only

RI and SE implementation

- Current action points in JIRA
- Current action points in GITHUB
- Stripping

Anatole’s blog post

- Posted at http://javaremarkables.blogspot.com.br/2014/06/jsr-363-unit-of-measurement-api-first.html
- Comments by Anatole, Werner and Heiko
    - **Did we capture those as tickets????**

Anatole's blog post

- **I think the API requires some entry point, e.g. in form of a singleton to make it complete. Basically a programmer should be able to program against the API without having to know any details on the implementations.**

- I like the idea, and agree with him but…

  - Can we do this in a portable ME/SE way, using Service Loader? Would that break in ME without Service Loader? Can we detected the presence and register or not? And OSGi?

Comments

- Jean-Marie suggestion so that we'd be inspired by javolution on OSGi support with private/non-exported service factories for non-OSGi environment, and OSGi service loader support would be implemented by a bundle

- This would address Anatole's concern.

- We'd need a specific package for factories interfaces (public in OSGi) and a specific package for the static factory access methods (private in OSGi)

Anatole’s blog post

- **Quantity / Measurement reasoning**
- Werner states some reasoning behind the way artefacts are laid out. Should we provide an explanation page or enhance the Spec Doc?

Public participation

- Can we get SouJava to get someone (not Otávio) who's unfamiliar to try and contribute to the project as a test of our infrastructure?
    - Is it easy for outsiders to compile and run our project?
- Should we prepare a "how to contribute to this project" page?
- Again, what do we need from them?
    - Use cases?
    - Test cases?
    - ~~Actual RI/SE code? (this brings IP considerations)~~

Comments

- Bruno said that “how to contribute” page is nice to have
    - List all items we’d need, despite having a nice infrastructure
    - Present this list in events so that they can start in easy stuff
    - “Just list the things that need to be done”
- have a page up describing when/what the hackaton would be
    - have this in the registration page, ask for use cases - “do you have any units and measurements in your application? use units to present values, <list instead of open question>”