

Ideas on modular UI

Vojtech Szöcs

Software Engineer at Red Hat

August 27, 2014

GWT applications in general, p1

- compiled GWT application is monolithic by design
 - process all source code and generate permutations
 - specific permutation loaded by selector script
 - code splitting (fragments) on Java method level
 - no inherent concept of logical application module
- application structure from source code perspective
 - component level – Model-View-Presenter, etc.
 - module level – ?

GWT applications in general, p2

- compiling monolithic GWT permutations [browser x locale]
 - need to parse and process all source code
 - source AST kept in memory per each permutation
 - huge resource consumption with multiple locales
 - known issue – GWT compiler runs out of temp. file handles
 - RFE – cut off locale vector, implement run-time i18n support
- debug performance proportional to compile performance
 - GWT 3.0 should bring incremental (faster) compilation
 - see [this blog post](#) for ideas on future of GWT

GWT applications in oVirt

- UI plugin infrastructure
 - designed to (only) extend standard UI
 - plugins cannot interfere with standard UI
 - plugins unable to depend on other plugins
 - plugins (only) consume exposed API
 - one global API, no plugin-specific API
- standard UI itself is still monolithic
 - “must know GWT” barrier to contribute core functionality

Modular UI, p1

- introduce concept of UI module
 - encapsulate functionality into logical units
 - modules can depend on other modules
 - each module is loaded only once
 - each module supports async loading at run-time
 - allow combining multiple modules into logical chunks
- module implementation
 - ES5 + [AMD](#) – [RequireJS](#) etc.
 - ES6 + native module support – [Traceur](#) etc.
 - module bundles (JS, CSS, etc.) using [webpack](#)

Modular UI, p2

- anatomy of UI module
 - JavaScript code
 - web resources (CSS, HTML etc.)
- ideas for different logical types of modules
 - layout – define base page layout with API for extension
 - example: “WebAdmin base layout”
 - extension – extend UI via API defined by other modules
 - example: “Virtual Machine main tab + sub tabs”
 - resource – provide 3rd party JS libs, common widgets etc.
 - example: “[PatternFly](#) skin and widgets for oVirt”

Modular UI, p3

- impact on frontend development
 - modules are developed and built separately
 - JavaScript is the lowest common denominator
 - core modules written in GWT/Java (reuse existing code)
 - choice (freedom) to write modules in different ways
 - reduce lock-in for GWT as main frontend technology
 - module's granularity determined by its logical scope
 - in a typical case, module consumes API of other module(s) in order to implement logical extensions, i.e. “new main tab”
 - reduce frontend code complexity
 - implementing feature via module should be less complex than implementing feature within monolithic structure



Thanks!

vszocs@redhat.com

vszocs at #ovirt (irc.oftc.net)