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
  - 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) compile
  - see this blog post for ideas on future GWT

GWT applications in oVirt

- UI plugin infrastructure
  - designed to (only) extend standard UI
  - plugins cannot interfere with standard UI
- standard UI itself is still monolithic

Modular UI, p1

- introduce concept of UI module
  - modules communicate through common interface
  - modules can declare dependencies on other modules
  - each module is loaded only once
  - each module supports async loading at run-time
    - allow combining multiple modules for initial page load
- 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 (HTML, CSS etc.)
- different logical types of modules
  - layout – define base page layout with API for extension
  - extension – extend UI via API defined by other modules
  - resource – provide 3rd party JS libs, common widgets etc.

Modular UI, p3

- impact on development
  - modules are built separately from each other
  - JavaScript is the lowest common denominator
  - modules can be written in JavaScript or anything that eventually becomes JavaScript (GWT etc.)
    - existing GWT code can be moved into different logical modules (virtualization, storage, networking, etc.)

Thanks!

vszocs@redhat.com

vszocs at #ovirt (irc.oftc.net)