oVirt.js PoC – Deep Dive

Vojtech Szöcs

Software Engineer at Red Hat

August 20, 2014

Topics covered

- oVirt.js PoC overview
- oVirt.js GWT wrapper
- Impact on oVirt GWT web apps
- Plans for near future

oVirt.js / summary

- JavaScript library to work with oVirt via REST API
  - essentially a JS binding utilizing REST concepts
- Works in web browsers out-of-the-box
  - allows extension to achieve portability (Node.js etc.)
- Dynamic discovery of Engine capabilities (HATEOAS)
  - resources, resource collections, resource actions
- PoC published on users and devel list

oVirt.js / hello DataCenter

```
// Obtain DataCenter resource collection.
// Resource collection is a handle that groups specific resource objects.
var dcCol = ovirt.api.datacenters;

// Obtain "add DataCenter" operation.
// Operation is a reusable RESTful HTTP request abstraction.
// You can modify and run the same operation multiple times, if necessary.
var addOp = dcCol.add({ name: 'test-dc', local: false });

// Attach result handler and run the operation.
// This fires off HTTP POST /api/datacenters
function dcAdded (dc) {
    console.log('Added: ' + dc.data().name);
}
addOp.success(dcAdded).run();
```

oVirt.js / API examples, part 1

```
// Operations on resource collection.
vmCol.list();     // list all VMs : GET    /api/vms
vmCol.get(id);    // get VM by ID : GET    /api/vms/{id}
vmCol.add(data);  // add new VM   : POST   /api/vms

// Operations on resource.
vm.update();      // update VM    : PUT    /api/vms/{id}
vm.delete();      // delete VM    : DELETE /api/vms/{id}

// Access and modify resource data.
var name = vm.data().name;
vm.data().name = 'test-vm';

// Bulk resource data update.
vm.data({ name: 'test-vm', stateless: false });
```

oVirt.js / API examples, part 2

```
// Nested resource collections.
var diskCol = vm.disks;

// Actions on resource.
var startOp = vm.start;

// Setting API (global) options.
ovirt.api.options({
    engineBaseUrl: 'http://127.0.0.1:8080',
    filterResults: false
});

// Setting operation options.
var listOp = vmCol.list();
listOp.options({ filterResults: true });
```

oVirt.js / design ideas, part 1

- Require ECMAScript 5 compliant environment
  - all modern browsers, including IE9 (*)
  - use ES5 shim for obsolete browsers

- Use Node.js to facilitate tools for JavaScript development
  - Grunt to automate common build tasks
  - Maven to integrate with Engine Java build workflow

- No additional runtime dependencies
  - inline everything into single JS file during the build

oVirt.js / design ideas, part 2

- Break the library into logical namespaces
  - services – used by API implementation
  - utilities – used to promote common code style
  - API – main entry point to oVirt.js functionality

- Try to write simple JavaScript, avoid bad parts
  - create objects via mixins
  - do not use “new” operator
  - avoid relying on object's prototype chain

oVirt.js / design ideas, part 3

- Establish environment that promotes testing (Karma)
- Consider using ES6 features today via Traceur

- Distribution channels
  - Engine i.e. HTTP GET /services/files/ovirt.js
  - popular web package managers (Bower, Jam etc.)
  - RPM repository hosted on Copr

- Source code management
  - initially part of ovirt-engine repository

GWT wrapper / summary

- GWT module providing Java API to oVirt.js via JSNI
  - bundles oVirt.js distribution and loads it if not detected
- Auto-generate code for specific RESTful objects
  - use Maven to fetch XSD / RSDL files as resources
- PoC published on devel list

GWT wrapper / hello DataCenter, part 1

```
// Obtain DataCenter resource collection.
ResourceCollection<DataCenter> dcCol = Sdk.get().api().getDataCenters();

// Create DataCenter data object from template.
DataCenterTemplate dcData = DataCenterTemplate.create(
    "test-dc", // name
    false);    // local

// Obtain "add DataCenter" operation.
Operation<DataCenter> addOp = dcCol.add(dcData);
```

GWT wrapper / hello DataCenter, part 2

```
// Create result handler.
SuccessCallback<DataCenter> dcAdded = new SuccessCallback<DataCenter>() {
    @Override
    public void onSuccess(DataCenter dc) {
        GWT.log("Added: " + dc.getName());
    }
};

// Attach result handler and run the operation.
addOp.success(dcAdded).run();
```

GWT wrapper / design ideas

- Check oVirt Java SDK code generators for reference

- Distribution channels
  - public Maven repository (Sonatype central etc.)

Impact on oVirt GWT web apps, part 1

- UiCommon currently works with backend business entities
- UiCommon currently invokes backend queries and actions

- Chance to improve frontend code
  - redefine UiCommon model's responsibility
  - simplify infra code (i.e. business logic in presenters)

Impact on oVirt GWT web apps, part 2

- Conceptual shift from using RPC to using REST

```
// RPC puts focus on operations.
// "runQuery" and "runAction" is essentially internal Engine backend API.
frontend.runQuery(GetAllDisksByVmId, params, callback);
frontend.runAction(AddDisk, params, callback);

// REST puts focus on entities.
// Following code is equivalent to running GetAllDisksByVmId query.
Sdk.get().api().getVms().get(id).success(vmCallback).run();
vm.getDisks().list().success(callback).run(); // inside vmCallback
```

Impact on oVirt GWT web apps, part 3

- Resource data aggregation, i.e. “get VM + all disks”
  - support by Engine REST API (Yoga etc.) – 1 request
  - support by oVirt.js API – 1+N requests

```
// Prefetch nested resource collections, not implemented in PoC.
var vmCol = ovirt.api.vms.options({ prefetch: 'disks' });
vmCol.get(vmId).success(vmCallback).run();

function vmCallback (vm) {
    var vmDiskArray = vm.disks.prefetched;
}
```

Impact on oVirt GWT web apps, part 4

- Addressing nested resources in single HTTP request

```
// Direct resource collection access, not implemented in PoC.
var diskCol = ovirt.api.collection('/vms/{vmId}/disks');
diskCol.options({ vmId: vmId });
diskCol.list().success(cb1).run();      // GET /vms/{vmId}/disks
diskCol.get(diskId).success(cb2).run(); // GET /vms/{vmId}/disks/{diskId}

// Operation pipelining, not implemented in PoC.
// Following code would require RSDL parsing at build or run time.
var vmCol = ovirt.api.vms.options({ pipeline: true });
var interOp = vmCol.get(vmId);          // intermediate
var termOp = interOp.disks.get(diskId); // terminal
termOp.success(callback).run();
```

Plans for near future

- Submit initial patch for ovirt-engine repository
  - include both oVirt.js and GWT wrapper projects
- First stable oVirt.js
  - set up automated build and test environment
  - finalize code and tests for essential features
- First stable GWT wrapper
  - implement code generation for all RESTful entities
  - create PoC – migrate select UiCommon model to REST

Thanks!

vszocs@redhat.com

vszocs at #ovirt (irc.oftc.net)