GETTING GROOVY WITH SNOMED CT – SOLVING PRACTICAL PROBLEMS WITH SCRIPTING IN SNOW OWL

Balazs Banfai, Brandon Ulrich
B2i Healthcare

What is Groovy?

Groovy is a dynamic programming language that builds upon the strengths of Java with a very flat learning curve for people familiar with object-oriented languages. It is easy to read and learn and has scripting and domain-specific language support which makes it ideal for providing scripting support for SNOMED CT tools written in Java.

Groovy highlights

- Runs on a Java VM
  - Call Groovy code from Java and to call Java code from Groovy.
  - Works with well-tested frameworks such as Spring
  - Groovy classes/Groovy scripts
- **Everything** is an object
  - -1.abs()
  - println "This is a string 321".*toSet().sort().join().reverse().replaceAll(" ", "");*
- Static or optional typing
  - str = “I’m a string”
- Collections are native to the language
  - code_systems = [‘SNOMED CT’, ‘ICD-10’, ‘ATC’]

B2i Healthcare

Groovy highlights - Closures

A *closure* is a piece of code wrapped up as an object.

```
def closure = { param -> println("hello ${param}") }
def closure = { println "hello " + it }
closure.call("world!") //hello world!

Closure envelope = { person -> new Letter(person).send() }
addressBook.each (envelope)
addressBook.each { new Letter(it).send() }

def value = [1, 2, 3].findAll { it > 1 }
assert value == [2, 3]

def service = new SnomedHierarchicalService()
def targetConcepts = service.getTargetConcepts(CLINICAL_FINDING_ID,
ASSOC_FINDING_ID)
targetConcepts.each {
        println "ID: $it.id label: $it.label”
}
```

B2i Healthcare

Groovy highlights - Markup

```
def service = new SnomedHierarchicalService()
def rootConcepts = service.rootConcepts
def rootNodes = service.getSubtypes(rootConcepts[0].id)
def writer = new StringWriter()
def xml = new MarkupBuilder(writer)
xml.rootConcepts() {
        rootNodes.each {
            def directChildren = service.getSubtypes(it.id)
             rootNode(id: it.id, label: it.label) {
                    directChildren.each { childNode(id: it.id, label: it.label) }
             }
        }
}
println writer.toString()
```

```
<rootConcepts>
 <rootNode id='78621006' label='Physical force'>
  <childNode id='80032004' label='Fire' />
  <childNode id='18213006' label='Electricity' />
      …
  <childNode id='125676002' label='Person' />
 </rootNode>
 <rootNode id='370115009' label='Special concept'>
  <childNode id='362955004' label='Inactive concept' />
  <childNode id='363743006' label='Navigational concept' />
 </rootNode>
 <rootNode id='363787002' label='Observable entity'>
  …
```

Healthcare

Groovy highlights - Dynamic Object Orientation

- Meta Object Protocol
- Builders
- ExpandoMetaClass

```
//extending a SNOMED CT concept representat
IComponent.metaClass.getToHtmlString {
 "<tr>\n<td>" + delegate.id + "</td>\n<td>" + delegate.label + "</td>\n</tr>\n"
}
println it.toHtmlString
```

- DSL support

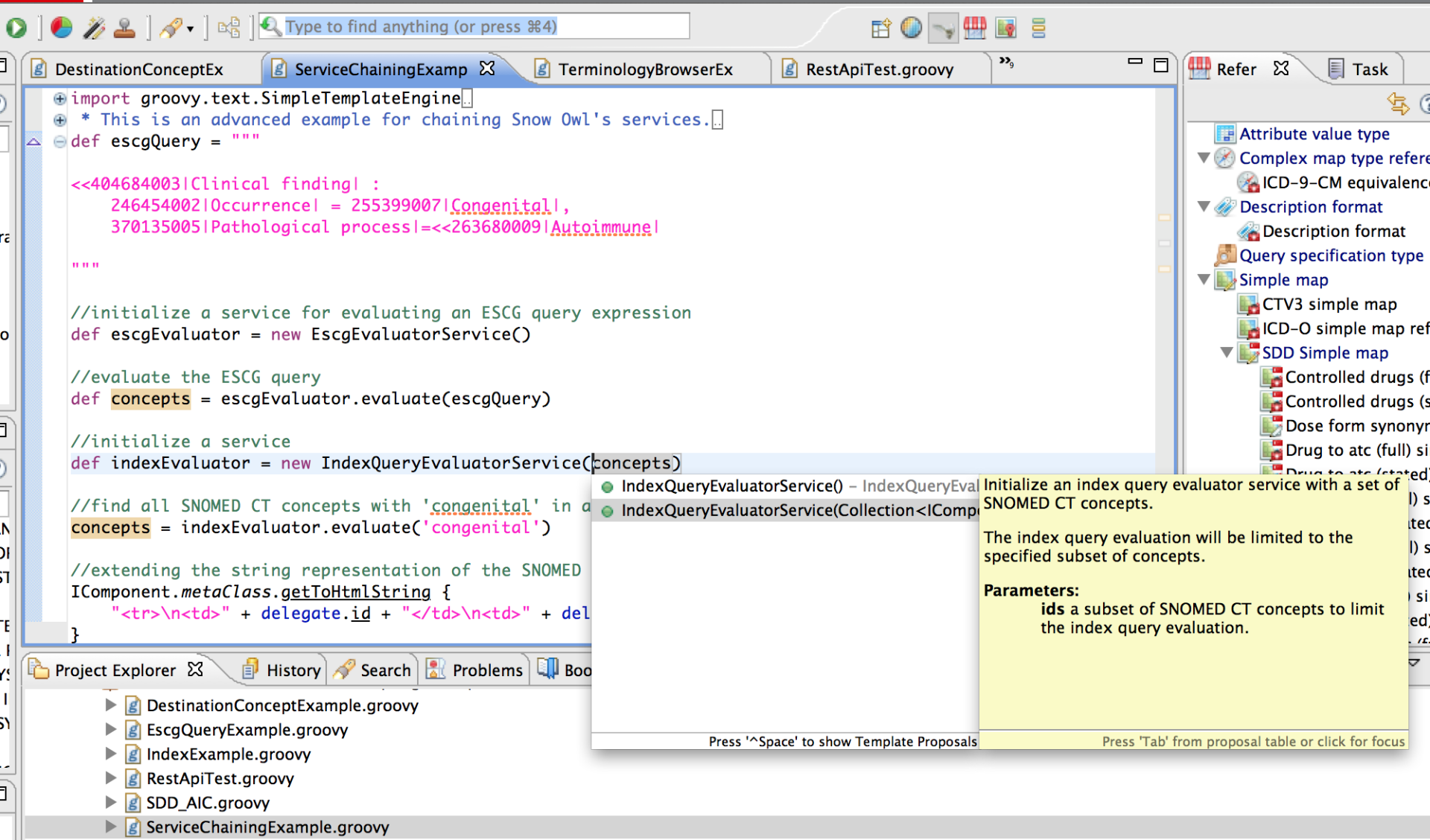
B2i Healthcare

Snow Owl's Groovy support

- Snow Owl provides and Integrated Development Environment (Groovy-Eclipse) that allows editing, compiling, running, and debugging Groovy scripts and classes from within Snow Owl.
- The integrated editor includes syntax highlighting, type inferencing, formatting, debugging, refactoring, auto-completion.
- Ideal for terminologist, medical informatics specialist or anyone interested in terminology management and authoring.

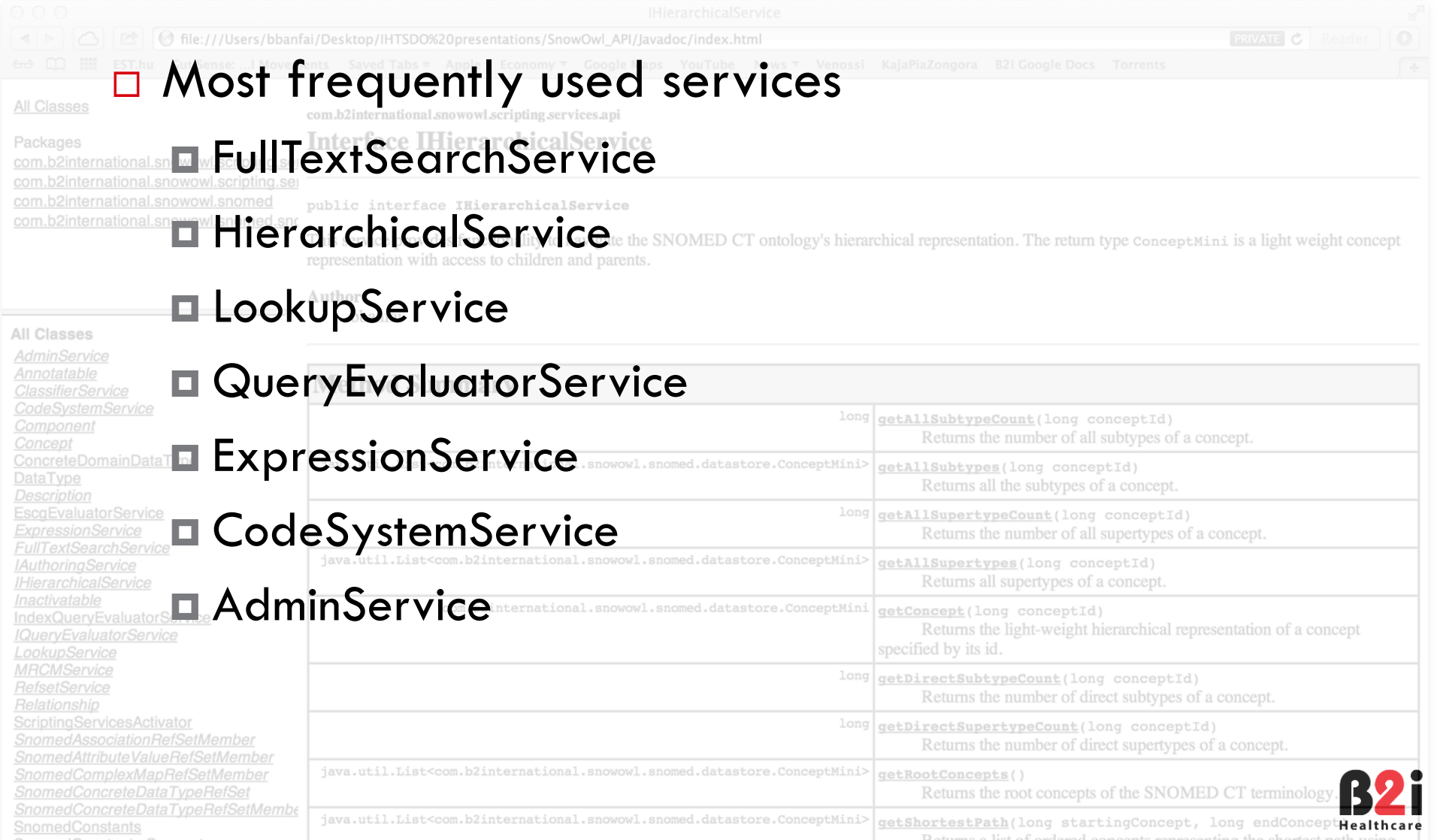
B2i Healthcare

Snow Owl's Groovy editor



Snow Owl service API

- Most frequently used services
  - FullTextSearchService
  - HierarchicalService
  - LookupService
  - QueryEvaluatorService
  - ExpressionService
  - CodeSystemService
  - AdminService



Candidates for scripting

- Ad-hoc queries
- Complex queries
- Ad-hoc reports in custom formats (text, markup)
- Artefact generation/updates
- Bulk terminology updates including the generation of terminology artefacts
- Terminology server access via REST protocol
  - Even from shell:

    ```
    #!/usr/bin/env
    groovy println "Hello, World!"
    ```
- DSL language development
- Combinations of the above with possible automatized execution

Concrete examples - Medicinal product descriptions

- List the SNOMED CT concept ID, PT, and Synonyms for
  - all descendants of 'Drug allergen or pseudoallergen'
  - not members in the Ingredients refset
  - tab separated table format.

ConceptId	PT	Synonym	Synonym	Synonym	Synonym
387121001	Clonidine hydrochloride				
96098007	Valaciclovir	Valacyclovir	Valacyclovir		
372511001	Benazepril				
373544004	Antazoline	Phenazoline			
6612003	Chloramphenicol Sodium Succinate				
88427007	Methyl acetylene	Propine	Allylene	1-Propyne	Propyne
38911000133101	Dapsone only				
372485004	Tiagabine				
404839003	Sodium Ibandronate	Ibandronate Sodium	Ibandronate sodium		
85603004	Triphenamyl				
….	…..				

Concrete examples – Ad-hoc report in html format

- Find all Clinical findings where the concept definition describes congenital origin and an autoimmune pathological process.
- Filter the results to concepts with any description that includes the word 'congenital'.
- Render the results in an HTML table

SNOMED CT ID	Preferred term
230672006	Congenital myasthenia
193216006	Congenital and developmental myasthenia
230677000	Congenital end-plate acetylcholinesterase deficiency
230673001	Congenital end-plate acetylcholine receptor deficiency

B2i Healthcare

Concrete examples - Medicinal product descriptions

Downstream users of NRC may have limitations in their systems as to the number of characters allowed within descriptions that are displayed. There was concern about the readability of these shorter descriptions for drugs that

- met a particular prescribing use case
- were **not** oral tablets
- contained less than 3 active ingredients

Therefore the customer wanted to output a list of preferred terms for these drugs along with the short description so that pharmacists could determine if there was potential for confusion when using the short names.

To answer this request, our customer first created a semantic query using the HL7 TermInfo standard to find drugs meeting a particular use case (medicinal product preparations) that did not have an oral tablet dose form. Next, the developer iterated through these results, discarding drugs that had more than 3 active ingredients. Finally, the developer created a file with a table of the preferred terms and short names.

Concrete examples – SNOMED CT DSL

- '386536003'.pt
- '386536003'.terms
- '386536003'.exist
- '126134000'.eachSynonym { println "ID: ${it.id} Term: ${it.label} »}

- '410607006'.subTypes.*each { println it.label }*

- def sortedBySizeSynonyms = '126134000'.synonyms.sort {it.label.*size()}*
- sortedBySizeSynonyms.*each { println it.label }*

- escgQuery.evaluate.filter('Congenital').toHtml

Future work

- Server-side scripting, scheduled automation
- Service injection into the runtime environment (no need to import and instantiate services)
- Continue to expose Snow Owl's functionality as a high-level API for scripting

Questions?

B2i Healthcare