

Apelon

Writing DTS Applications

Apelon

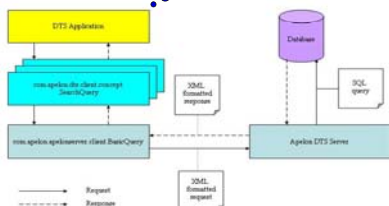
Writing DTS Applications

- This training provides a overview to writing DTS applications.
- It presents and expands on the standard DTS application "recipe".
- Additional support can be found in the JavaDocs and DTS Programming Tutorial:
Install_dir/Apelon/DTS/javadocs/com/apelon/dts/Install_dir/doc-files/tutorial.html

Apelon

Programming with DTS

We will develop here as a DTS Client



Apelon

Requirements

The following are needed prior to developing against the DTS Server.

- Java JSDK 1.5 (JDK & JVM)
- or .NET Framework 1.1.
- DTS Server
- Database to house DTS Schema
- Your favorite IDE

Apelon

Setup

- IDE Classpath should contain all files in:
[Install_dir/apelon/dts/lib](#)
- A specific SAX Parser must be used with the JVM, either by an invocation property:
[org.xml.sax.driver=org.apache.xerces.parsers.SAXParser](#)
or Runtime method:

```
System.setProperty("org.xml.sax.driver",  
    "org.apache.xerces.parsers.SAXParser");
```

Apelon

Gedanken Application

- Pedagogic purposes now, but will implement in next section
- Application is to display the SNOMED CT 'code' associated with an SNOMED CT Concept (found via its name)
- This code is in the *Code in Source* Property of the SCT Concept.

Apelon

The DTS Application Recipe

1. Connect to the DTS server
2. Instantiate the required query and helper classes
3. Get the desired objects
E.g., find the correct SNOMED Concept
4. Process/Display the output
E.g., print the SCT code

Apelon

Step 1

Connect to the DTS server – 3 ways

- [ServerConnectionSocket](#)
 - Standard user connection
 - Server host and port
- [ServerConnectionSecureSocket](#)
 - Higher security
 - Server host, port, username and password
- [ServerConnectionJDBC](#)
 - Simplified connection, bypasses sockets
 - Direct access to the db (no server required)
 - See JavaDoc for details

Apelon

Step 1

Connect to the DTS server:

[com.apelon.apelonserver.client.ServerConnectionSocket](#)

- throws `ApelonException` on failure

```
String dtsHost = "localhost";
int dtsPort = 6666;
ServerConnectionSocket connection =
    new ServerConnectionSocket(dtsHost, dtsPort);
```

Apelon

Step 2

Create the desired query class

- To find specific Concepts and fetch Concept data only need a basic query class; DTSTConceptQuery will do fine
- Would need a SearchQuery class to do actual searching of the knowledgebase

`com.apelon.dts.client.concept.DTSTConceptQuery`

```
DTSTConceptQuery conceptQuery =  
    DTSTConceptQuery.createInstance(connection);
```

Apelon

Step 2

Create the desired query and helper classes

- Query classes: objects that directly access the database
- Internal DTS "constants":
 - Mappings from desired attribute Names to internal attribute Types
 - ConceptAttributeSetDescriptors (fetched Concept context)

Apelon

Side Trip - Query Classes

- Query classes are action classes that mediate the transfer of information between the client and server
- ONLY Query classes have this function
- Query classes are abstract, they are instantiated via a `createInstance` method:

`com.apelon.dts.client.concept.DTSTConceptQuery`

```
DTSTConceptQuery conceptQuery =  
    DTSTConceptQuery.createInstance(connection);
```

Apelon

Side Trip - Namespaces

Get the Namespace ID

- Since concept names are only unique within a Namespace, we will need to limit our name lookup to "SNOMED CT"
- The DTS API uses integer ids and type objects rather than strings as keys to objects (like the SNOMED CT Namespace)
- We will see that this requires the ID of the Namespace
- To get a Namespace's Id, use a NamespaceQuery

[com.apelon.dts.client.namespace.NamespaceQuery](#)

```
NamespaceQuery spaceQuery =  
    NamespaceQuery.createInstance(connection);  
Namespace namespace =  
    spaceQuery.findNamespaceByName("SNOMED CT");  
int ctId = namespace.getId();
```

Apelon

Side Trip - Properties

Property

- A name-value pair associated with a concept
- A Property is an instance of a DTSPROPERTYTYPE
- Name is the Type, Value is an arbitrary string
- Use an subclass of BaseConceptQuery to get the type from a name (DTSCONCEPTQUERY works)

[com.apelon.dts.client.attribute.DTSPROPERTYTYPE](#)

```
String propName = "Code in Source";  
DTSPROPERTYTYPE propType =  
    conceptQuery.findPropertyTypeByName(propName, ctId)
```

Apelon

Side Trip - CASD

- DTS optimizes Concept retrieval by allowing the programmer to specify which Concept attributes are needed
- This is specified by a ConceptAttributeSetDescriptor (CASD) object:

[com.apelon.dts.client.concept.ConceptAttributeSetDescriptor](#)

- A CASD is a set of attribute types that is used as a "filter" on Concepts returned from the database
- Static ALL_ATTRIBUTES and NO_ATTRIBUTES are available
- CASDs *may* improve performance

Apelon

Side Trip - CASD

We add the desired PropertyType to our CASD

```
ConceptAttributeSetDescriptor casd =  
    new ConceptAttributeSetDescriptor("demo");  
casd.addPropertyType(propType);
```

Alternately:

```
casd.setAllPropertyTypes();
```

Or even:

```
casd = ConceptAttributeSetDescriptor.ALL_ATTRIBUTES;
```

Apelon

Step 3

Get the required concept

- Lookup the name in the selected Namespace
- Include an ConceptAttributeSetDescriptor which tells DTS which Concept Attributes to load

```
String ctName = "Pneumonia (disorder)";  
DTSConcept ctConcept =  
    conceptQuery.findConceptByName(ctName, ctId,  
    casd);
```

Apelon

Side Trip - Associations

ConceptAssociations

- Connect two Concepts with a named relationship
- Instances of an AssociationType
- Use an AssociationQuery to get the type from a name

[com.apelon.dts.client.association.*](#)

```
String mapping = "SNOMED CT to ICD-9-CM map";  
AssociationQuery assnQuery =  
    AssociationQuery.createInstance(connection);  
AssociationType mapType =  
    assnQuery.findAssociationTypeByName(mapping, ctId)
```

Apelon

Step 4

Display the output

- Fetch the properties (as specified in the CASD)
- If necessary, check for correct type
- Extract the value

```
DTSPProperty props[] = ctConcept.getFetchedProperties();
for (int i=0; i<props.length; i++) {
    if (props[i].getPropertyType().equals(propType)) {
        System.out.println("Code is " + props[i].getValue());
        return; // returns only one instance
    }
}
```

Note "Fetched"

Note no type specification

Apelon

Extra Credit - Search By Name Pattern

- Now we want to search by a name
'pattern' (wild-card)
- We need a new query class and we
need to set up search options

Apelon

Step 2 (By Pattern)

Create the desired query class

- Use the SearchQuery class
- Set up default search options (DTSSearchOptions)

[com.apelon.dts.client.concept.SearchQuery](#)
[com.apelon.dts.client.concept.DTSSearchOptions](#)

```
SearchQuery SearchQuery =
    SearchQuery.createInstance(connection);
. . .
DTSSearchOptions options =
    new DTSSearchOptions(10, ctId, casd);
```

Apelon

Step 3 (By Pattern)

Read the name and get the associated concept

- Find names that match the input *pattern* Returns an array of Concepts whose name matches the pattern

```
String ctName = "**pneumonia*";
DTSTConcept[] results = (DTSTConcept[])
    searchQuery.findConceptsWithNameMatching(ctName,options);
```

Apelon

Extra Credit - Search By Attribute

- Now we want to search by a Property (say *Code in Source*)
- We can use a different method of SearchQuery

Apelon

Step 2 (By Property)

Create the desired query class

- Use the SearchQuery class
- Set up default search options (DTSSearchOptions)

```
com.apelon.dts.client.concept.SearchQuery
com.apelon.dts.client.concept.DTSSearchOptions
```

```
SearchQuery searchQuery =
    SearchQuery.createInstance(connection);
. . .
DTSSearchOptions options =
    new DTSSearchOptions(10, ctId, casd);
```

Apelon

Step 3 (By Property)

Read the code and get the associated concept

- Find concepts whose *Code in Source* Property matches the input *pattern* (wild-carding is supported)
- Returns an array of Concepts

```
String ctCode = "222*"  
DTSTConcept[] results = (DTSTConcept[])  
    searchQuery.findConceptsWithPropertyMatching(propType,  
        ctCode,options);
```

Similar methods are available for Synonyms, Roles, and Associations.

Apelon

Extra Credit – Add a Local Property

- A common application is to add a local name, or code, to a formal, e.g. SNOMED, Concept
- We will add a 'MyCode' local Property to a SNOMED CT Concept

Apelon

Add a Local Property – Prework

- All of the below can be done via the API, but since this is only done once, it's easier to do using the DTS Editor (admin privileges required)
- Create a new Thesaurus Namespace, "MySpace"
- Create a new DTSPROPERTYType, "MyCode", in "MySpace" that attaches to Concepts

Apelon

Add a Local Property – Step 2

Get the Type for our Property:

```
DTSPROPERTYTYPE myPropType =  
    conceptQuery.findPropertyTypeByName(propName, mySpaceId)
```

Note that this is the
MySpace Namespace Id

Apelon

Add a Local Property – Step 4

Add the Property

- Create a DTSPROPERTY
- Add it to the Concept

```
DTSPROPERTY myProp =  
    new DTSPROPERTY(myPropType, "123345");  
  
ctConcept.addProperty(myProp);  
  
ctConcept =  
    conceptQuery.addProperty(ctConcept, myProp);
```

Need a Query Class to affect db!

Apelon

Other Query Classes - NavQuery

- NavQuery
 - Used to navigate Ontology or Thesaurus hierarchies

```
NavChildContext childCtx =  
    navQuery.getNavChildContext(concept, casd)  
or  
  
NavChildContext childCtx =  
    navQuery.getNavChildContext(concept, casd, assnType)
```

Then

```
ConceptChild[] kids = childCtx.getChildren();
```

ConceptChild is a subclass of DTSPROPERTY

Apelon

NavQuery

- Similar methods for parents:

```
NavParentContext parentCtx =  
    navQuery.getNavParentContext(concept, casd)
```

- Can also get roots:

```
ConceptChild[] roots =  
    navQuery.getConceptChildRoots(casd, nsId);
```

Apelon

OntylogClassQuery

- Specialized class for subsumption, aka “class” queries, in Ontylog Namespaces
- To get all descendants of a concept:

```
OntylogConcept[] cons =  
    classQuery.getSubConcepts(con, casd);
```

- To determine if a con1 is a descendant of con2:

```
boolean isDescendant =  
    classQuery.isSubConcept(con1, con2)
```

Apelon

Questions?



Apelon
