BIOINFORMATICS APPLICATIONS NOTE Vol. 24 no. 10 2008, pages 1321–1322
doi:10.1093/bioinformatics/btn122

Databases and ontologies

UniProtJAPI: a remote API for accessing UniProt data

Samuel Patient*, Daniela Wieser, Michael Kleen, Ernst Kretschmann, Maria Jesus Martin and Rolf Apweiler

The European Bioinformatics Institute, Hinxton, Cambridge CB10 1SD, UK

Received on December 6, 2007; revised on April 1, 2008; accepted on April 3, 2008

Advance Access publication April 4, 2008

Associate Editor: Martin Bishop

ABSTRACT

Summary: Programmatic access to the UniProt Knowledgebase (UniProtKB) is essential for many bioinformatics applications dealing with protein data. We have created a Java library named UniProtJAPI, which facilitates the integration of UniProt data into Java-based software applications. The library supports queries and similarity searches that return UniProtKB entries in the form of Java objects. These objects contain functional annotations or sequence information associated with a UniProt entry. Here, we briefly describe the UniProtJAPI and demonstrate its usage.

Availability: http://www.ebi.ac.uk/uniprot/remotingAPI

Contact: spatient@ebi.ac.uk

1 INTRODUCTION

The Universal Protein Resource (UniProt; The UniProt Consortium, 2008) provides a comprehensive and freely accessible central resource of protein sequences and functional annotation. UniProt data can be browsed online through http://www.uniprot.org. In addition, a number of services exist (Labarga *et al.*, 2007) to retrieve the data in various formats including XML, RDF, fasta and flat file. In order to process the information contained in these formats, a parser needs to be written that transforms the input into suitable data structures. Sometimes this can be done without further knowledge of the UniProt entry structure; the protein sequence, for example, can simply be parsed from the corresponding sequence line. Difficulties arise if splice variants of a protein sequence are required, since both the comment and feature sections of the entry need to be parsed at the same time and a deeper understanding of the UniProt structures is needed. Furthermore, data format changes within UniProt can lead to significant maintenance overhead.

Therefore, a Java application programming interface (UniProtJAPI) has been developed to provide remote access for Java applications processing UniProt and related data, granting users access to all four major components in UniProt. These are the UniProt Knowledgebase (UniProtKB), the UniProt Reference Clusters (UniRef), the UniProt Archive (UniParc) and the UniProt Metagenomic and Environmental Sequences (UniMES) database. Additionally, the UniProtJAPI has been extended to take into account information referenced in UniProtKB entries, for instance InterPro (Mulder *et al.*, 2007). This is a resource of protein families, domains and sites, containing a number of member databases that derive protein signatures. The UniProtJAPI provides this InterPro-derived information such as scores, and start and end positions of the signatures, for each UniProtKB entry. The UniProtJAPI provides the ability to perform text and sequence similarity search across all this data, allowing users to access a single database entry with a given accession number, or whole entry sets matching defined criteria.

2 CENTRAL DATA STRUCTURES

The UniProtJAPI represents protein data from the UniProtKB as Java objects, which enables programmatic retrieval of functional annotation and sequence information. The object structures resemble the flat file and XML format structures to facilitate access for those users already familiar with the UniProtKB formats.

The five central data structures of the UniProtJAPI are: *UniProtEntry*, *UniParcEntry*, *UniRefEntry*, *ProteinData* and *UniMesEntry*. A *UniProtEntry* object represents a UniProtKB entry and provides methods to access all of its information. For example, getDescription().getProteinName() returns the protein name associated with this entry. A *UniParcEntry* object models a UniParc non-redundant sequence while a *UniRefEntry* object is a UniRef sequence cluster of 100, 90 or 50 percentage identity. The relationship between the former objects is represented in a *ProteinData* object. The *ProteinData* object associated with a UniProtKB entry is accessible via the getUniProtEntry() method. The UniParc entry and the three levels of UniRef sequence clusters relating to the UniProtKB entry are also accessible using the getUniParcEntry() and getUniRefEntry() methods. Additional information is available from the getInterProMatches() method that returns a list of InterProMatch objects. These represent the sequence patterns for the UniProtKB entry. Lastly, a *UniMesEntry* object provides access to a UniMES sequence and its related sequence patterns. A graphical representation of the UniProtJAPI object model along with the library documentation is available online.

3 LICENSE AND GETTING STARTED

UniProtJAPI is based on open-source technologies and the software is under the Apache License, version 2. UniProt data

*To whom correspondence should be addressed.


```
1  public class ApplicationNoteDemo {
2
3  public static void main(String args[]) throws Exception {
4
5      ProteinDataQueryService service = new UniProtRemoteServiceFactory().getProteinDataQueryService();
6
7      String sequence = ">MES00005665499\n" +
8                        "MSNHGFAYFFTSYQSLSLDSSSPPPSPHPRAHASSRFPPRARAVASFHTSCKMARTKQTA\n" +
9                        "RKSTGGKAPRKQLATKAARKSAPATGGVKKPHRYRPGTVALREIRKYQKSTELLIRKLPF\n" +
10                       "QRLVREIAQDFKTDLRFQSSAVLALQEASEAYLVGLFEDTNLCAIHAKRVTIMPKDVQLA\n" +
11                       "RRIRGERA";
12
13     String jobid = service.submitBlast(new BlastInput(DatabaseOptions.UNIPROT_SP, sequence));
14
15     // Jobid used to check status
16     while (!(service.checkStatus(jobid) == JobStatus.DONE)) {
17        Thread.sleep(5000);// Sleep a bit before the next request
18     }
19     System.out.println("Blast job complete");
20
21     // The blast data contains the job information and the best hits
22     BlastData<ProteinData> blastResult = service.getResults(jobid);
23     List<BlastHit<ProteinData>> blastHits = blastResult.getBlastHits();
24     ProteinData bestHit = blastHits.get(0).getEntry();
25
26     System.out.print("Best hit: " + bestHit.getUniProtEntry().getPrimaryUniProtAccession().getValue() + " ");
27     System.out.print(bestHit.getUniProtEntry().getDescription().getProteinNames() + " ");
28     System.out.println(bestHit.getUniProtEntry().getOrganisms().get(0));
29
30     // Show all postranslational modifications on the sequence
31     Collection<ModResFeature> features = bestHit.getUniProtEntry().getFeatures(FeatureType.MOD_RES);
32
33     for (ModResFeature feature : features) {
34        System.out.println("From " + feature.getFeatureLocation().getStart() + " to " + feature.getFeatureLocation().getEnd() +
35              " " + feature.getFeatureDescription().getValue() + " " + feature.getFeatureStatus().getValue());
36     }
37
38     // Show all InterPro cross-references
39     List<InterPro> dbXReferences = bestHit.getUniProtEntry().getDatabaseCrossReferences(DatabaseType.INTERPRO);
40
41     Collection<DatabaseCrossReference> xref = bestHit.getUniProtEntry().getDatabaseCrossReferences();
42     for (DatabaseCrossReference databaseCrossReference : xref) {
43        System.out.println("databaseCrossReference = " + databaseCrossReference);
44     }
45
46     // Show all InterPro matches with positions and score
47     List<InterProMatch> matches = bestHit.getInterProMatches();
48
49     for (InterProMatch match : matches) {
50        System.out.println("Match: " + match.getMethodName().getValue() + " " + match.getMethodAccession() + " from " +
51              match.getFromPos() + " to " + match.getToPos() + " " + match.getScore());
52     }
53
54     EntryIterator<ProteinData> blastResultIterator = service.getEntryIterator(blastResult.getResultAsQuery());
55     System.out.println("Number of Blast hits " + blastResultIterator.getResultSize());
56
57
58     for (InterPro iprXref : dbXReferences) {
59        Query query = UniProtQueryBuilder.buildDatabaseCrossReferenceQuery(iprXref.toString());
60        EntryIterator<ProteinData> resultSet = service.getEntryIterator(query);
61        System.out.println("InterPro  " + iprXref + " contains " + resultSet.getResultSize() + " members");
62        // Create intersection between Blast result
63        blastResultIterator = service.getEntryIterator(blastResultIterator, SetOperation.And, resultSet);
64     }
65     System.out.println("Intersection contains " + blastResultIterator.getResultSize());
66  }
67 }
```

Contains Blast results, UniProt entry objects, additional data from InterPro

Best hit is a characterized protein: "P84239"

Prints database cross-references from EMBL, SMR, InterPro, Gene3D, Pfam, PRINTS, SMART and PROSITE

Prints scores and positions of the sequences patterns from Gene3D, Pfam, PRINTS, SMART and PROSITE

Fig. 1. UniProtJAPI example source code showing the investigation of an uncharacterized UniMes sequence using Blast.

is under the Creative Commons Attribution-NoDerivs license. To use the UniProtJAPI, a compressed data file (zip file) that contains all the Java classes has to be downloaded. The library requires a Java 5 runtime environment or above and an HTTP connection.

4 EXAMPLES

The UniProtJAPI may be used in a broad range of applications. It is used in the generation of IntAct (Kerrien *et al*., 2007) to retrieve a list of all proteins that have been updated between two dates. The IntEnz (Fleischmann *et al*., 2004) project uses it to retrieve all database cross-references for a specific EC number. Additional usages could include a user wanting to investigate characterized proteins that are similar to an unknown protein. In order to assist the user with this task, the similarity search tool Blast is integrated in the API.

The use of Blast is demonstrated in Figure 1, where an uncharacterized sequence from the UniMES database is submitted to retrieve a set of similar *ProteinData* objects (line 13). Common Blast options are available; these include similarity matrix options, searching against a specific organism or setting the *E*-value threshold. For brevity, default options are used in the example. The Blast service returns a collection of *ProteinData* objects and alignment information (line 22). Each of the *ProteinData* objects contains a *UniProtEntry* object which can be used to extract functional annotation, database cross-references or other entry information such as its sequence or the organism name (lines 26–36). The *ProteinData* object that is returned as the best hit by the Blast search corresponds to the UniProtKB/Swiss-Prot entry P84239 (line 26). For further analysis of this object, the database cross-references are accessed, showing hits to the EMBL, SMR, InterPro, Gene3D, Pfam, PRINTS, SMART and PROSITE databases (lines 39–45). In addition, the supplementary *InterProMatch* objects are accessed to provide scores and positions of the sequence patterns from Gene3D, Pfam, PRINTS, SMART and PROSITE databases (lines 47–53). A closer look at the InterPro cross-references shows that the best hit object belongs to three InterPro families all of which describe histone proteins. It would be interesting to compare the proteins belonging to these families against the full Blast result, not only against the best hit. To do this, further queries against InterPro are made, and result sets are combined using binary operations.

The UniProtJAPI supports binary operations that allow the combination or intersection of sets of Java objects (line 64). Intersecting these sets retrieved during the InterPro and the Blast queries, results in a further set of *ProteinData* objects (lines 59–67). Each object in the final result set belongs to the InterPro groups IPR009072, IPR007125 and IPR000164, which can then be used for further analysis (line 66).

ACKNOWLEDGEMENTS

We thank all Consortium members, in particular, N. Sklyar and E. Salazar.

Funding: UniProt is supported by the NIH grant 2 U01 HG02712-04, the EC FELICS grant (021902RII3), the NIH grant 1R01HGO2273-01, by the Swiss Federal Government through the Federal Office of Education and Science and by the NIH grants and contracts HHSN266200400061C, NCIcaBIG, and 1R01GM080646-01, and the NSF grant IIS-0430743.

Conflict of Interest: none declared.

REFERENCES

Fleischmann,A. *et al*. (2004) IntEnz, the integrated relational enzyme database (Database issue). *Nucleic Acids Res*., **32**, D434–D437.

Kerrien,S. *et al*. (2007) IntAct–open source resource for molecular interaction data (Database issue). *Nucleic Acids Res*., **35**, D561–D565.

Labarga,A. *et al*. (2007) Web services at the European Bioinformatics Institute (Web Server issue). *Nucleic Acids Res*., **35**, W6–W11.

Mulder,N. *et al*. (2007) New developments in the InterPro database. *Nucleic Acids Res*., **35**, D224–D228.

The UniProt Consortium (2008) The Universal Protein Resource (UniProt). *Nucleic Acids Res*., **36**, D190–D195.