Databases and ontologies
caBIONet—a .NET wrapper to access and process genomic data stored at the National Cancer Institute’s Center for Bioinformatics databases
Piotr Kraj and Richard A. McIndoe

1 Institute of Molecular Medicine and Genetics and 2 Center for Biotechnology and Genomic Medicine, Medical College of Georgia, Augusta, GA 30912, USA

Received on January 28, 2005; revised on March 15, 2005; accepted on June 14, 2005
Advance Access publication June 16, 2005

ABSTRACT
Motivation: The National Cancer Institute’s Center for Bioinformatics (NCICB) has developed a Java based data management and information system called caCORE. One component of this software suite is the object oriented API (caBIO) used to access the rich biological datasets collected at the NCI. This API can access the data using native Java classes, SOAP requests or HTTP calls. Non-Java based clients wanting to use this API have to use the SOAP or HTTP interfaces with the data being returned from the NCI servers as an XML data stream. Although the XML can be read and manipulated using DOM or SAX parsers, one loses the convenience and usability of object oriented programming paradigm. caBIONet is a set of .NET wrapper classes (managers, genes, chromosomes, sequences, etc.) capable of serializing the XML data stream into local .NET objects. The software is able to search NCICB databases and provide local objects representing the data that can be manipulated and used by other .NET programs. The software was written in C# and compiled as a .NET DLL.
Availability: The program is freely available to academics and non-profit organizations from www.amdcc.org. The source code is available from the authors upon request.
Contact: rmcindoe@mail.mcg.edu

1 INTRODUCTION
With the rapid accumulation of genomic information in biomedical research, advanced analytic tools and data warehouses have been developed to advance discoveries in basic research to clinical trials and drug discoveries. This objective is made possible by the development of software tools and environments that support complex data integration from multiple databases. The National Cancer Institute’s Center for Bioinformatics (NCICB) has developed a collection of integrated data repositories that provide web-based analysis tools and development environments. These repositories are collectively known as caCORE and consist of the Gene Expression Data Portal (GEDP), MAGE database, Cancers Model Database (caMOD), Cancer Image Database (caIMAGE), Cancer Molecular Analysis Project (CMAP), Cancer Genome Anatomy Project (CGAP) and the Cancer Bioinformatics Data Objects (caBIO) (Covitz et al., 2003). caCORE serves an important role in defining standards for biomedical nomenclature as well as the modeling, representation and exchange of data. caBIO is the software API of caCORE, providing programmatic access to the genetic information. This API can be used by Java clients and/or Java server pages to access the data (e.g. BioGopher). The sources of genomic information pertaining to human and mouse stored at the NCI servers include NCBI UniGene and Entrez-Gene databases, Gene Ontology Consortium, HomoloGene database, BioCarta pathways, CGAP, CMAP and others. This information has been curated and is optimized for access by multiple APIs.

dcaBIO is developed as an n-tiered architecture consisting of a presentation layer, object layer and data layer. Clients (web browsers) interact with the presentation layer via Java Server Pages and/or Java Servlets to return dynamic content in response to client requests. Logical objects in the presentation layer are implemented using Java Beans. External clients can access caBIO data and objects using either SOAP or HTTP interfaces. Java servlets receive the HTTP query request formatted as a URL and return XML-encoded responses to HTTP clients. Non-Java applications can use the SOAP API to retrieve biological data from caCORE. The returned SOAP result can then be parsed by the local SOAP client.

The caBIO object layer consists of Domain Objects, Object Managers and Data Access Objects implemented as Java classes. Domain Objects consist of biological components, such as genes, chromosomes, sequences, libraries and pathways. Object Managers are server-side objects that act as an intermediary between the Domain Objects and Data Access Objects. Finally, Data Access Objects provide direct access to the data stored in relational databases, flat files or distributed systems at NCI.

The caBIO software suite and databases can be downloaded and installed on a local server. However, this will require local curation, maintenance and update of these data warehouses. This can be quite time consuming and complicated. Alternatively, investigators wanting to take advantage of the NCI database infrastructure can access the genomic information in caBIO using the caBIO SOAP/HTTP API.

Because caBIO is implemented using the Java programming language, non-Java clients and programs are required to use the SOAP/HTTP interfaces to search and retrieve the biological data stored at NCI. This means all the data are transferred as XML and
must be consumed using either DOM or SAX parsers. Computer programmers can then extract specific data based on the caBIO DTD provided by NCI. A principal drawback in this approach is that one loses the advantage of an object-oriented programming paradigm to consume and manipulate the biological data.

Our laboratory develops a number of programs and web-based applications using the .NET framework. One of these websites, the Animal Models of Diabetic Complications (www.amdcc.org), requires the use of genomic information for the description of the genotypes of the animal models developed by the consortium. Rather than maintain external databases locally, we decided to use caBIO to retrieve genomic and biological data for the genes under investigation.

2 DESCRIPTION OF caBIONet

To access caBIO data and manipulate it locally we have developed a .NET wrapper that consumes the returned XML data stream and creates local .NET caBIO domain objects. In order to accomplish this, we had to not only write the .NET domain object equivalents, but also the search criteria objects, parser and the client communication interface necessary to interact with the NCI servers. caBIONet, the resulting software, is based on caBIO v2.0 and written in C# and compiled as a .NET dynamic link library (DLL). .NET is a development platform designed to facilitate object-oriented web development with compilers available for both Windows and Linux. C# is the major programming language for the .NET platform, specially suited for developing distributed web applications based on shared components. C# implements features found in both C/C++ (high performance, object-oriented structure) and Java (security, memory management, garbage collection).

Implementation

caBIONet accesses caBIO data using the HTTP API interface and reads data formatted as an XML data stream. The caBIONet domains currently supported are: caBIOGene, caBIO GeneHomolog, caBIOGeneAlias, caBIOChromosome, caBIOGoOntology, caBIOMapLocation, caBIOPathway, caBIOProtein, caBIOProteinHomolog, caBIOSNP, caBIOSequence and caBIOTaxon (Fig. 1). We have implemented most of the attributes and functionalities of the respective caBIO classes. The current version of caBIONet has 29 classes with 313 properties and methods.

Similar to caBIO, programmatic retrieval of data from NCI is accomplished using domain specific methods of a ManagerClass. Unlike caBIO, we have one manager class with methods to retrieve the various biological domain data rather than a separate manager for each domain. This simplified our implementation and streamlined the code execution. Each method in the ManagerClass can accept either a domain specific SearchCriteria object or a URL string. The latter are returned from NCI as a part of the XML data stream. Each of these domain specific methods will create a valid URL based on the search criteria object properties, make the connection to the NCI servers, parse the returned XML data and create the local .NET objects based on the returned XML. One can also provide proxy information (host, port, user and password) for institutions that require a proxy to access the internet.

As stated above, the methods in the ManagerClass accept search criteria objects corresponding to the various object-specific domains (e.g. GeneSearchCriteria, PathwaySearchCriteria). Each of the individual domain SearchCriteria classes has object-specific class attributes (member variables) based on the caBIO API documentation and are accessible by the corresponding class property (get and set). The use of properties is a characteristic of the .NET framework. They act like Java class set/get methods to the class constructor as well as provide access to the class state as if they were class attributes. The domain-specific properties for the search criteria classes are essentially the filtering options for domain-specific searches. For example, setting the Symbol property of the GeneSearchCriteria class will restrict the search to genes matching that value (Fig. 2).

Every property of the object-specific SearchCriteria class has an associated custom attribute. The value of this custom attribute is a string corresponding to the search criteria keyword in the URL to be posted to the NCI server (e.g. &crit_cloneName= &crit_scienificName=). These keywords are described in the caBIO documentation. The URL keyword is set to the value of the associated object property of the object-specific SearchCriteria class. This caBIO URL keyword and keyword value is then concatenated to a base URL string containing the location of the caBIO java servlet (http://cabio.nci.nih.gov/servlet/GetXML?). Using object reflection, we are able to build this URL string very quickly because the value of the property as well as the value of the custom attribute associated with that property are returned simultaneously and concatenated into the URL string.

Once the query results are received from NCI, the returned XML data is input into an XML DOM document object and passed to a Parser class containing methods for extracting object-specific information from XML (e.g. ParseGeneXML, ParseTaxonXML) as well as URL links for subsequent searches (e.g. GetGeneXLink, GetTaxonXLink). Information extracted by the Parser class is then used to populate domain specific caBIONet objects. These methods can return either the object itself or an ArrayList of domain objects. For example, the goOntologies property of the caBIO Gene object is an ArrayList of caBIOGoOntology objects (Fig. 2).

3 SUMMARY

caBIONet provides a very simple object-oriented programming interface for accessing NCI data for local .NET applications. As illustrated in Figure 2, a basic search for a gene can be performed in as little as 5–7 lines of code (depending on whether or not you need to define proxy information). We will provide the source code and compiled libraries free to the community under an open source
Fig. 2. Example C# code using caBIONet. Institutional proxy information can be defined in the manager class. This example, ASP.NET web page, performs a gene lookup for the mouse gene symbol H2-Aa and writes selected returned information to the web page.

```csharp
using System;
using System.Net;
using caBIONet;

cabIONet.ManagerClass manager = new ManagerClass();
manager.ProxyHost = "proxy.mcg.edu"; //add any necessary proxy information.
manager.ProxyPort = 0000;
cabIONet.GeneSearchCriteria gsc = new GeneSearchCriteria();
gsc.Symbol = "H2-Aa";
gsc.TaxonId = 6; //mouse
ArrayList gscResults = manager.GetGenes(gsc); //returns an ArrayList of caBIOGen objects
foreach (caBIOGene gene in gscResults)
{
    Response.Write(gene.title + "(" + gene.symbol + ")" + <br>);
    Response.Write("Chromosome: " + gene.chromosome.name + ")" + <br>);
    Response.Write("Number of Ontology Terms: " + gene.goOntologies.Count.ToString() + ")" + <br>);
    foreach (caBIONet.caBIOGoOntology ontology in gene.goOntologies)
    
    Response.Write(ontology.name + " " (GOId=" + ontology.idGO.ToString() + ")" + ")" + <br>");

license agreement (version 1.1). This will allow others to augment or modify the code as we only implemented the objects pertinent to our needs. For example, we did not implement the clinical trial related objects.

GLOSSARY
API—Application Programming Interface
CGAP—Cancer Genome Anatomy Project
CMAP—Cancer Molecular Analysis Project
DOM—Document Object Model
SAX—Simple API for XML
SOAP—Simple Object Access Protocol

URL—Uniform Resource Locator
XML—Extensible Markup Language

ACKNOWLEDGEMENT
This work is supported by grant U01DK60966-01 from the National Institute of Diabetes Digestive and Kidney disorders to RAM.

Conflict of Interest: none declared.

REFERENCE