An Application Programming Interface (API) for Programmable Access to Animal QTLdb

Zhi-Liang Hu*, Carissa A. Park, James E. Koltes, Eric Fritz-Waters and James M. Reecy

Department of Animal Science and Center for Integrated Animal Genomics, Iowa State University, 2255 Kildee Hall, Ames, IA 50011

Abstract

The development of the Animal QTLdb has been successful in terms of its growing content and numbers of utilities, users, and citations by journal papers. The success of the Animal QTLdb is partly attributable to the web, which made it possible for many remote users to access the database simultaneously through interfaces designed to serve various pre-defined purposes. We have continually develop the web interfaces by implementing new functions, query options, and online analysis tools, to assist users with hypothetical and requested types of data query. To date, we have developed more than 70 user options implemented in 20+ CGI programs. Since the development of these tools was driven partly by our motivations and partly by user demands, and they are implemented on the server side, the users' ability to alter the ways data are queried and summarized is limited. Recently we have started to develop a programmable access (application programming interface, API) platform for users to write their own scripts to query data. This interface allows users to script their own logics and query combinations, and possibly set up a cron job to make the queries automatic. To showcase, we have implemented a half dozen API portals to accept scriptable access to the database. The advantage of this platform is that users can modify and run their scripts at any time, obtain data or data summary, and flag certain data with their own filters, directly into a tabulated or other preferred format.

Introduction

The utility of the Animal QTLdb is evidenced by the growth in web portal access, frequency of downloads, and increasing number of citations by scientific publications over the past 10 years (Hu et al., 2013; and *http://www.animalgenome.org/QTLdb/pubs*), yet the demand is building for better data access and for diverse types of data analysis. Although we have been trying to meet the challenges by adding more web portal tools in response to user requests, there is a relatively large lag time spent developing these tools. To improve our ability to better help the community, we have developed an application programming interface (API) for the Animal QTLdb to allow users more expedited data access, and for implementation of their own data analysis strategies in real time.

We have created the Animal QTLdb API using REST architectural framework and XML data mark up language following models used in NCBI's E-utilities (NCBI, 2010). Access to the API tool is through the HTTP protocol on the internet. Therefore, the API is available for any users with internet access and the programming ability to make HTTP calls in any programming language (Figure 1).

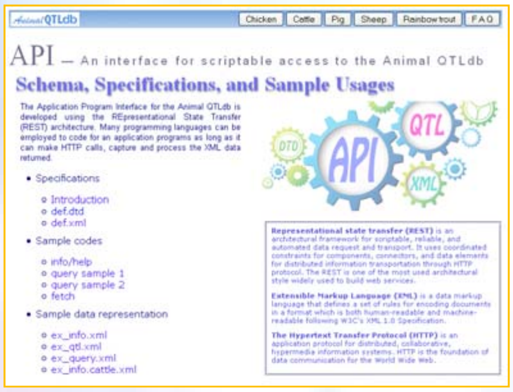
Results

The results from our recent developmental efforts on a new Animal QTLdb API presented here are preliminary. We show the common architectural syntax, parameters, and options (Table 1) in actual client-side program examples in Perl (Box 1). We try to show the logic and parameters in a straight forward manner, so that users can follow the examples to program their own client-side applications in any other languages as long as the programs support HTTP communication calls over internet (Figure 2). These results are also available online (Figure 1). Since this work is under active development, future changes are expected.

This platform is also designed to provide user space on the server to allow additional functions for registered users. For example an application programmer may chain query actions and the API server may save a query history for the user for easier query interactions. The privileged features will be available in the coming weeks.

Figure 1.

The web resources for the Animal QTLdb API. Please check the URL for updates (http://www.animalgenome.org/QTLdb/API/), as the tools are under active development.



AnimalQTLdb | Chicken | Cattle | Pig | Sheep | Rainbow trout | FAQ

API — An interface for scriptable access to the Animal QTLdb

Schema, Specifications, and Sample Usages

The Application Program Interface for the Animal QTLdb is developed using the REpresentational State Transfer (REST) architecture. Many programming languages can be employed to code for an application programs as long as it can make HTTP calls, capture and process the XML data returned.

- Specifications
  - Introduction
  - def.dtd
  - def.xml
- Sample codes
  - info/help
  - query sample 1
  - query sample 2
  - fetch
- Sample data representation
  - ex_info.xml
  - ex_qtl.xml
  - ex_query.xml
  - ex_info.cattle.xml

Representational state transfer (REST) is an architectural framework for scriptable, reliable, and automated data request and transport. It uses coordinated constraints for components, connectors, and data elements for distributed information transportation through HTTP protocol. The REST is one of the most used architectural style widely used to build web services.

Extensible Markup Language (XML) is a data markup language that defines a set of rules for encoding documents in a format which is both human-readable and machine-readable following W3C's XML 1.0 Specification.

The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web.

Framework and Implementation

The Animal QTLdb API is implemented following REST (REpresentational State Transfer) architecture. The REST is an inter-computer framework designed for scriptable, reliable, and automated data request and transport, and is one of the most widely used architectural styles to build web services. Because it uses coordinated constraints for components, connectors, and data elements for distributed information transportation through well-known HTTP protocol (Figure 2), it is relatively straightforward to program client-side applications.

We have implemented a server-side API platform using Perl/CGI. It serves the client-side API program calls through Apache web server with data queried from the backend MySQL database.

Scheme and Usage

We designed the Animal QTLdb API to follow human intuition in the way one would go about looking into a database, in three logical steps: (1) "info"/"help" query for an overall idea of what the database has and how the data are organized, or how to use the API platform; (2) keywords "query" for targeted information; and (3) data "fetch" to obtain the data located in the above steps. These logical functions are implemented in three server-side programs called "iinfo", "iquery", and "ifetch". In Box 1 is shown what's required and how client-side API programs can be structured.

Box 1

The three server-side programs that accept API/HTTP calls. The comment lines give limited annotation of the functions, purposes, and options on the main parameters as defined in Table 1.

```
use strict;
use warnings;
use LWP::Simple;
my $api= "http://www.animalgenome.org/cgi-bin/QTLdb/API";
```

iinfo

```
my $query = 'info';   #- Command: 'info', 'help'
my $scope = 'cattle'; #- Species: 'cattle', 'pig', etc.
my $email = 'myname@mydomain';

#- option 1:
my $einfo = "$api/iinfo?q=$query&history=y";
#- option 2: with "scope"
my $einfo = "$api/iinfo?q=$query&s=$scope&history=y";
#- action:
my $getinfo = get($einfo);
print "$getinfo\n";
```

iquery

```
my $api= "http://www.animalgenome.org/cgi-bin/QTLdb/API";
my $handle = 'genes'; #- Data scope: 'traits',
                      #- 'publications', 'breeds', 'genes',
                      #- 'chrloc'.
my $scope = 'cattle'; #- Species: 'cattle', 'pig', etc.
my $query = 'holstein';   #- Breed name
my $query = 'bone';       #- Key words
   $query =~ s/\s/\%20/g; #- Use URL-safe encoding
my $myquery = "$api/iquery?q=$query&s=$scope&h=$handle&history=y";
my $query_results = get($myquery);
print "$query_results\n";
```

ifetch

```
my $query = '12890, 12896, 12925, 13826, 15090, 15110, 17227, 17228,
17241, 17242, 17730, 17734, 17741 '; #- QTL IDs, may be from ‘iquery’
my $scope = 'pig'; #- Species: 'cattle', 'pig', 'chicken', etc
my $fetch = "$api/ifetch?q=$query&s=$scope&history=y";
my $fetched = get($fetch);
print "$fetched\n";
```

Box 2

Sample XML-representations of Animal QTL API outputs

(a) on "info" request

```
<EINFOresults xmlns="http://www.animal.../def.xml">
   <QTLdbSummary>
      <Species name="Cattle">
         <Count name="QTL">11,543</Count>
         <Count name="Publication">505</Count>
         <Count name="Trait">481</Count>
      </Species>
   </QTLdbSummary>
   <QTLdbSummary>
      <Species name="Chicken">
         <Count name="QTL">4,337</Count>
         <Count name="Publication">209</Count>
         <Count name="Trait">305</Count>
      </Species>
   </QTLdbSummary>
   <QTLdbSummary>
      <Species name="Horse">
         <Count name="QTL">345</Count>
         <Count name="Publication">16</Count>
         <Count name="Trait">9</Count>
      </Species>
   </QTLdbSummary>
   <QTLdbSummary>
      <Species name="Pig">
         <Count name="QTL">11,610</Count>
         <Count name="Publication">433</Count>
         <Count name="Trait">649</Count>
      </Species>
   </QTLdbSummary>
   <QTLdbSummary>
      <Species name="Rainbow Trout">
         <Count name="QTL">127</Count>
         <Count name="Publication">10</Count>
         <Count name="Trait">14</Count>
      </Species>
   </QTLdbSummary>
   <QTLdbSummary>
      <Species name="Sheep">
         <Count name="QTL">789</Count>
         <Count name="Publication">90</Count>
         <Count name="Trait">217</Count>
      </Species>
   </QTLdbSummary>
</EINFOresults>
```

(b) on "query" for "1:1232-1923873"

```
<EqueryResults xmlns="http://www.animal.../def.xml">
  <QTL>
    <dataType>QTL</dataType>
    <symbol>FATTH</symbol>
    <id>1317</id>
    <location property="chromosome">1</location>
    <location property="linkage_map">0.9</location>
    <location property="genome_map">50468-180633</location>
  </QTL>
</EqueryResults>
```

Table 1.

Programs and parameters used in the QTLdb API. (a) Basic programs and parameters; (b) Additional parameters.

(a)

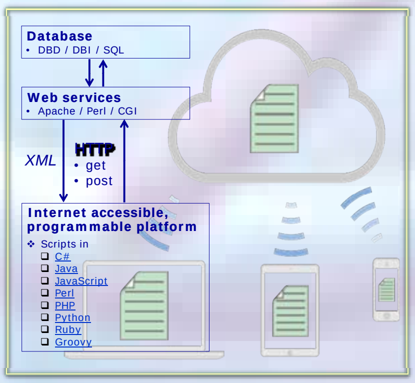
Program	Keys	Property	Values
iinfo	query (q)	required	"info", "help"
	scope (s)	required	Species, e.g. "cattle", "horse", etc.
iquery	query (q)	required	Any key word
	scope (s)	required	Species, e.g. "cattle", "horse", etc.
	handle (h)	optional	"publications", “traits", "breed", "genes", "chrloc"
ifetch	fetch (f)	required	A string of QTL IDs delimited with comma.
	format (m)	optional	"XML"(default), "GFF", "CSV", etc.

(b)

Programs	Keys	Property	Values
iinfo, iquery, ifetch	email (e)	required	a valid email address for contact
	usehistory (u)	optional	"y" vs "n"

Figure 2.

A conceptual REST architecture to show the information flow where data are handled through HTTP protocol.



Discussion

The Animal QTLdb API is designed for structured data to be queried and transmitted in an unambiguous manner. We adhere to XML for reliable data transfers and parses on the client side. Although it's possible to support other data formats, such as JSON, YAML, and CSV, for the time being, we leave it to client-side programs to convert to or from a needed data format.

References

1. Hu, Zhi-Liang, Carissa A. Park, Xiao-Lin Wu and James M. Reecy (2013). Animal QTLdb: an improved database tool for livestock animal QTL/association data dissemination in the post-genome era. Nucleic Acids Research, 41 (D1): D871-D879.
2. National Center for Biotechnology Information (2010); NCBI Help Manual: Entrez Programming Utilities. NCBI Bookshelf online, URL: http://www.ncbi.nlm.nih.gov/books/NBK25501/.
3. Fielding, Roy Thomas (2000). Architectural Styles and the Design of Network-based Software Architectures. Doctoral Dissertation. University Of California-Irvine URL: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

Acknowledgements

This work is partially supported by the Animal Genome Bioinformatics Coordinators funds from USDA-NIFA.