Function:Troilkatt/SpellWeb2

From FunctionWiki

The SpellWeb2 project implements a web application that provides the Spell data exploration GUI as JavaScript code running in the users browser, which communicates with a server using GWT-RPC (implemented over Ajax). For quick start details refer to the deployment section.

To go to the main design document click here.

Contents

Version History

- Version 0.1 [1] (http://incendio.princeton.edu/functionwiki/index.php?title=Function:Troilkatt/SpellWeb2&oldid=3536)

Requirements

SpellWeb2 has the following requirements:

1. The GUI should be similar to the Ruby-on-rails SpellWeb implementation, and provide at least the same functionality.
2. The client code should be portable and run in all browsers the Google tools run in.
3. The search result visualization should scale to compendium with thousands of genes, and provide smooth interactive performance.
4. The code should be easy to maintain and extend.

Goals

SpellWeb2 goals:

- Look-and-feel on par with SpellWeb.
- Interactive scalable search result visualization, that handles thousands of datasets.
- Java and/or JavaScript implementation.
- Final system should be deployed, including source code, documentation, and test cases.

Dependencies

Internal dependencies:

- DistributedSpell is used to execute search queries. Communication uses the DistributedSpell protocol.
- DistributedSpell code is used by SpellWeb2 (for handling metadata and PCL file data)
- Troilkatt Pipeline provides and organizes the data and metadata for SpellWeb2.
- GoTermFinder is used to execute go-term queries. Communication uses the GoTermFinder protocol.

External dependencies:

- Google Web Toolkit is used to implement the system.
- Jetty (embedded) is used as a container for running the system standalone.
- A Java servlet container is required for deployment.
- The code is in Java 1.6.

Deliverables

Fully implemented and tested system including:

- Source code.
- Design document.
- Javadoc API documentation.
- Automated test cases and test descriptions.
- Performance evaluation.

Open Issues

See also buglist.

Non-implemented, nice to have functions:

1. Query suggestion.
    - Approach 1: suggest single gene names (currently only works for first)
    - Approach 2: suggest a query (string).
2. Move all expression value handling functions to DistributedSpell such that SpellWeb2 only handles meta data.

- SearchResults.java: class used to return search results, including expression values.

Open, minor, design issues:

1. How to display additional information about a gene or dataset?
    - As a new tab (current approach).
    - As a pop-up dialog (as done in SpellWeb for some information).

Detailed Description

Operational Scenario

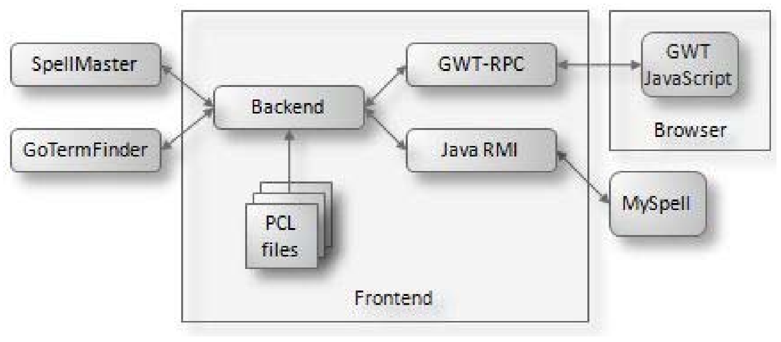
The Troilkatt pipeline generates and organizes multiple compendium with dataset and metadata files (for example one per organism). A single SpellWeb2 server is then started on a public web server. It is synchronized with the SpellMaster search server such that both have identical datasets and metadata information.

A user opens the SpellWeb2 page in her browser (possibly using an URL that specifies the compendium to use). This will load the SpellWeb2 JavaScript code in the browser, which provides the search and result visualization user interfaces. The user interacts with the local GUI elements, which sends search and metadata information requests to the web server. The web server responds with search results and other data, which the JavaScript code displays in the browser.

Architecture

SpellWeb2 has the common three-tier web application architecture. The user interface is run in the users browser. Most of the application logic is handled by a web application server that converts the user provided queries to a format that can be processed by the DistributedSpell search engine, and that presents the received results. The SpellWeb2 server exports two interfaces: A GWT-RPC interface used by the client code running in a users browser, and a Java RMI interface used by clients implemented in Java, such as Function:Troilkatt/Private data. Internally, both the GWT-RPC and Java RMI classes consist of wrapper functions that call functions in a common Backend servlet

The three main design requirements are: portability, high performance, and ease of extension. We believe the Google Web Toolkit (GWT) application design satisfies these three requirements. The code is portable, since GWT is already sed by major Google applications such as Wave that runs on most devices with a JavaScript capable browser. GWT has better interactive performance than the Ruby-on-rails (used by the SpellWeb) since it moves the user interaction code from the server to the browser. In addition, the code is easy to understand and extend for developers since everything is programmed in Java, which is the language used in the other systems (DistributedSpell, Troilkatt, GoTermFinder, and MySpell).



The server is responsible for:

- Mapping of gene names provided by the user to system wide identifiers.
- Provide functions for retrieving expression values from datasets.
- Provide functions for retrieving metadata about datasets.
- Forward Spell search queries to the search server, and return search results and the associated meta data.
- Forward GO term search queries to the go-term server and return results.

The JavaScript code running in the users browser implements:

- All GUI elements (widgets) (and associated event handling code).
- Visualization of expression values received from the server.
- Visualization of tables of dataset information received from the server.

Performance and Resource Usage Considerations

The system should provide interactive performance. The main performance issues are:

1. Network latency will introduce an overhead of up to 100ms for client-server communication. However, most user input events do not require the client to contact the server. Most client-server communication is to execute queries, or to dataset expression values and meta-data.
2. Data transmission time (server side data processing overhead, network bandwidth, and client side data processing overhead) can be hundreds of ms when transmitting data for many datasets in one GWT-RPC call. We therefore apply client side caching, lazy data fetch, and we tune the datasets per call.

Data Structures

The main data structures in the server are (all in SpellWeb2Backend):

- *hom*: a Homology data structure that includes the expression values for all datasets (for additional details see ?)
- *geneName2ID*: hash table that maps gene names (systematic, common, and aliases) to gene identifiers.
- *geneID2Name* and *geneID2Systematic*: hash tables that maps gene identifiers to respectively common gene names and systematic gene names.
- *invalidAliases*: a list of invalid gene aliases. These aliases are mapped to two or more gene IDs and do therefore not uniquely identify a gene.
- *dsetName2ID* and *name2dsetID* hash table that map from dataset names to dataset identifiers and vice versa.
- *datasetDetails* dataset metadata hash table indexed by dataset ID.
- *detailsByPubl*, *detailsByCond*, and *detailsByPubmed*: arrays with indexes to metadata structures sorted by respectively publication details, number of conditions, or pubmed IDs.
- *buf*: a byte buffer shared between all sockets. NOTE! THIS IS A CONCURRENCY BUG!

All data structures are written once at load time. All request handling only requires reads, since the server does not maintain any client state for a session, query, or GWT-RPC call. Note that each organism has a separate instance of the above data structures, such that for example the geneID2Name is implemented as a hash table that maps an organism ID to a hash table with gene ID to name mappings.

Servlet Life Cycle and Concurrency Issues

The server code is run in the context of a Java Servlet. The life cycle for servlets is well defined in the Java Servlet Specification [2] (http://java.sun.com/products/servlet/download.html) :

1. init() is used for costly one-time activities such as loading the PCL files,
2. service() is called by threads handling HTTP requests. There can be multiple concurrent calls to service().
3. destroy() can be called at anytime by the servlet container to remove the service.

The data server data structures described in the previous section are loaded in the init() phase, while GWT-RPC requests are handled in the service() phase by a thread handling one request. Multi-threading is easy for SpellWeb2 since no client state is maintained by the server, no requests requires updating a global data structure, and since most request are light in computation and I/O (the exception being requests that require reading a lot of expression values). Since the data structures are not updated they can be simply deleted during destroy().

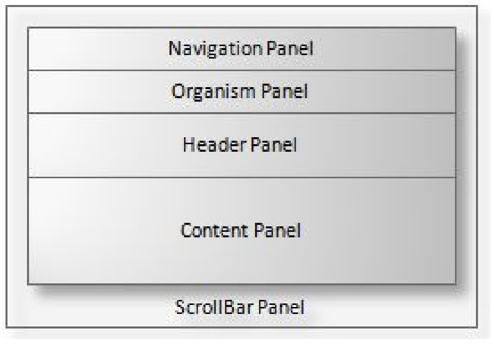
Note that we do not directly touch any of the above functions since GWT-RPC hides the servlet details. The servlet specification also specifies the failure models and exceptions [3] (http://java.sun.com/products/servlet/download.html) . In addition, the Tomcat documentation describes how the CommonsLogging framework can be used for logging [4] (http://tomcat.apache.org/tomcat-6.0-doc /logging.html) .

SpellWeb2 exports a Java RMI [5] (http://download.java.net/jdk7/docs/platform/rmi/spec/rmiTOC.html) interface for clients implemented in Java (and not JavaScript). Java RMI servers are (most likely) multi-threaded, with an object life cycle similar to web application servlets.

GUI Organization and Layout

SpellWeb2 provides several views that the user can switch between:

- Search view: is the main page that allows specifying the genes to search for.
- Result view: shows the results for a search.
- Private search view: is a page for downloading and running MySpell. Once started MySpell provides it's own GUI (which is an extension of the SpellWeb2 GUI).
- Dataset listing view: is a table with details for the datasets in the compendium.
- Gene expression listing view: allows viewing the gene expressions in all datasets for a set of genes.
- Expression values view: shows the results for a gene expression listing.
- About view: is a page with textual information about SpellWeb2 and the compendium.
- Server error view: is displayed when the server has crashed (or is not running).



The SpellWeb2 GUI is organized into four panels that are inside a *scroll panel* for vertical and horizontal scrolling. First, the *navigation panel* has links to switch between the provided views. Second, the *organism panel* has links that allow switching between organism compendium. Finally, the *header panel* has widgets that allow to control the content in the *content panel*.

The header and content panel contains the widgets for all views, but only the widgets for the current view are visible. To switch between views the old views panels are set to non-visible, and the new are set to visible. To switch between views the user clicks the links in the navigation panel. This fires a *history event*, for which the event handling code sets the widgets in the old view to non-visible, and the new view widgets to visible. The user can also use the browsers back and forward buttons to switch views. But we do not save any state about previous searches, so it is not possible to do a search and then go back to the search results of a previous search,

To switch between compendium we use the browsers internationalization mechanism. But instead of changing the language used for messages, we change the organism which is described. The *locale parameter* is therefore set to a KEGG organims identifier. For example, using an URL with locale=hsa gives human specific messages. Using the internationalization mechanism provides ease of implementation and faster startup time due to the way the GWT downloads JavaScript code.

CSS

To change the layout of SpellWeb2 it is typically enough to modify the SpellWeb2.css file which specifies the "styles" used in the GUI (fonts, margins, colors, etc).

There are five main types of styles:

- Overrides of styles such as HTML headers (h3 and h1).
- Derived (subclass) styles for GWT widgets that inherit the default style, but add more padding (such as .gwt-Label-padded-right adds a bigger margin right of a label). To use these styles the addStyleDependentName("suffix") method is called for the widget.
- Panel styles that are used to add margins and padding around GUI elements (most panels use the listSubPanel or contentSubPanel style). These are added to a widget using the addStyleName("name") method.
- Widget specific styles (such as searchBox that specifies the style of the text box on the frontpage).
- Table styles for setting the background color and fonts for the heatmap and go-term tables.

The CSS styles does not specify the following which instead are specified in the code:

- GUI organization and widget/panel placement (this must be done in the code).
- Widget alignment in panels.

We recommend using a browser plugin such as Web Developer and DOM Inspector for Firefox that provides a point-and-click interface to get style information about a given widget, and runtime editing of the .css file.

JavaScript Client

The SpellWeb2 client is a JavaScript program executed by a browsers JavaScript engine, such as V8 in Chromium, SpiderMonkey in Firefox, or JScript in IE. The JavaScript engine provides the types, objects, operators and functions defined in the ECMA standard ensuring portability of JavaScript code among browsers. However, performance may differ since the engine is also responsible for memory allocation, garbage collection, and other optimizations (such as compilation to binary code as done by V8). For more details refer to for example the design document for V8 [6] (http://code.google.com/apis/v8/intro.html) .

The SpellWeb2 client code can only communicate with the web application server since the browsers Single Origin Policy (SOP) restricts the JavaScript running in the browser to interact with a single server. The SpellWeb2 server therefore forwards requests to the search and go-term servers.

Communication Protocols

SpellWeb2 uses GWT-RPC for communication. It is implemented on top of Ajax (asynchronous JavaScript and XML), which is implemented on top of TCP. GWT-RPC is an asynchronous communication protocol to avoid blocking the GUI thread of the single threaded JavaScript engines used in most browsers. The server returns serialized Java objects, which we have designed such that they can be quickly serialized.

The most important function with regards to performance is the search function. After the user clicks the search button the following steps are executed:

1. The client JavaScript code executes an asynchronous GWT-RPC call to the search() function.
2. To handle the call the web application sends a search request to the distributed search engine and receives a list of genes and associated scores, and a list of datasets and associated weights. These are saved in an SearchResult object using Java native types.
3. The server adds the expression values for the top ranked genes and datasets in the SearchResult object.
4. The SearchResult is serialized and sent to the client.
5. Upon receive of he result a function is called by the JavaScript engine that displays the results by converting the expression values to pixels and displays these by updating a GWT widget.

Using the JavaScript based Ajax protocol is impractical for non-JavaScript applications. SpellWeb2 therefore also exports a Java RMI interface with the same functions. The RMI protocol [7] (http://download.java.net/jdk7/docs/platform/rmi/spec/rmi-protocol.html) is over (a non-HTTP) socket.

Logging

Server side logging is implemented using Java Common Logging. The logging level can be adjusted through a command line argument, and be set as follows:

- SEVERE: for exceptions that cannot be handled and that require the server to shut down.
- WARNING: application level exceptions that can be handled on the server side, but introduces a client side error message.

- INFO: informational messages.
- CONFIG: static configuration messages such as ports used, etc.
- FINE: tracing information.
- FINER and FINEST are not used.

The default log handler is used, which typically writes the log files to standard output. For Tomcat logging details refer to [8] (http://tomcat.apache.org/tomcat-6.0-doc/logging.html) .

Exception Handling

The server throws three kinds of exceptions:

- InvalidParameterException: for invalid method parameters (such as a wrong organism ID), but only non-user provided input arguments. This exception therefore represents an error in the code. For user provided input arguments, the correctness is either checked by the client-side code (such as page numbers) or returned as an error message (such as invalid gene names).
- ServerIOException: or IOExceptions either when reading local files or communicating with other servers.
- RuntimeException: thrown by the server if it receives and unrecoverable error such as a wrong configuration file. These exceptions should normally not occur during runtime and hence there is no strategy for handling them.

InvalidParameterException and ServerIOException are handled exceptions that the servlet container simply forwards to the client. For these exceptions the client displays an error message and continues running. For a RuntimeException the servlet is shut down (but may be restarted by the container). For these exceptions client displays an error page informing the user that the server is down.

Portability

The client side code of SpellWeb2 uses only GWT widgets. Hence, it should run on all browsers supported by GWT. As of May 2010 this includes:

- Firefox 1.0, 1.5, 2.0, 3.0, and 3.5
- Internet Explorer 6, 7, and 8
- Safari 2, 3, and 4
- Chromium and Google Chrome
- Opera 9.0

Programming Interface

See also Javadoc for API specification: [link]

SpellWeb2 is split into two projects in the CVS: SpellWeb2 and SpellWeb2JettyServer. SpellWeb2 is a GWT project where the server part can be compiled and packaged as a web application and run in a container such as Apache, Tomcat, or as an Eclipse plugin. SpellWeb2JettyServer is a standalone Java program that starts the SpellWeb2 server.

SpellWeb2 uses the GWT code structure. The code in SpellWeb2/src has the following packets:

- edu.princeton.function.spellweb2: contains the SpellWeb2.gwt.xml and SpellWebSuper.gwt.xml configuration files. The later is for projects that "subclass" SpellWeb2 such as SpellWeb2JettyServer.
- edu.princeton.function.spellweb2.client: contains the client side code.
- edu.princeton.function.spellweb2.server: contains the server side code.
- edu.princeton.function.spellweb2.shared: contains classes shared by the server and client.

In addition SpellWeb2/test contains JUnit test files for the SpellWeb2 code.

SpellWeb2JettyServer/src has a single source file in edu.princeton.function.spellweb2.standalone.

Client

The client code is organized into the following files:

- SpellWeb2.java: has the main entry point and code to switch between views.
- SpellWebService.java and SpellWebServiceAsync.java: the interface exported by the server.
- View.java: superclass for all views. It also contains utility functions to check integer ranges, and creating HTML links.
- [About, DatasetDetails, DatsetList, ExpressionValues, GeneList, GeneListResult, MySpell, Result, Search, ServerError,

SingleExpression]View.java: one class per view.
- [Help, Warning]Dialog.java: code for the help and warning dialog boxes.
- HelpLabel.java: code for the help label which is a clickable icon.
- HelpMessages.java: the text of the help messages.
- SpellWeb2[Constants, Messages].java: definitions for the constants and messages for different organisms
- SpellWeb2Constants[_debug, _hsa, _sce]: the organism specific constants.
- SpellWeb2Messages[_debug, _hsa, _sce]: the organism specific messages.
- [InvalidParameter, ServerIO]Exception.java: excpetions thrown by the server.

The onModuleLoad() code in SpellWeb2 setup the GUI top-level organization, create the different view objects, and handle view change events (implemented using the browsers history mechanism).

Each view class must implement two functions: setWidgets() and setVisible(). These are called respectively when initializing the vidget, and when all widgets in a should be set to non-visible or visible. Each view class also implements the event handling code for it's widgets.

Server

The server code is organized into the following files:

- SpellWebBackend.java: the backend code that implements all functionality.
- SpellWebServiceImpl.java: GWT-RPC interface implementation. All functions call the corresponding function in SpellWebBackend.
- SpellWebServiceRMI.java and SpellWebServiceRMIImpl.java: Java RMI interface specification and implementation. All functions call the corresponding function in SpellWebBackend.
- DatasetDetailsComparator.java: comparator class that allows sorting datasets by different fields.

For a detailed description of the API refer to the SpellWebService Javadoc: [link to SpellWebService interface in javadoc].

Shared

The shared package contains classes that are shared between the client and server:

- DatasetDetails.java: class used to return dataset details to be viewed in the *dataset list view*.
- ExperimentConditions.java: class used to return experiment conditions to be viewed in the *single gene expression levels view*.
- GoTerms.java: class used to return GoTerm search results.
- OrganismIDs.java: a map of organism IDs to common names.
- SearchResults.java: class used to return search results, including expression values.

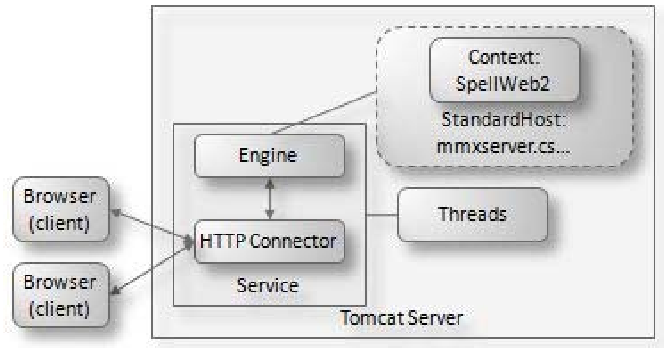
Other

Other important files in the source directory are:

- SpellWeb2/src/edu.princeton.function.spellweb2.SpellWeb2.gwt.xml: GWT configuration files, that contains the entry-point class and one extended-property entry for each organism.
- SpellWeb2/src/edu.princeton.function.spellweb2.SpellWeb2Super.gwt.xml: is a GWT configuration file used by projects that "subclass" SpellWeb2. It is identical to the above file, except that it does not provide an entry class (which is defined in the sub-projects configuration file).
- SpellWeb2/war/SpellWeb2.html: the HTML file for SpellWeb2.
- SpellWeb2/war/SpellWeb2.css: the CSS file that defines the layout.
- SpellWeb2/war/WEB-INF/web.xml: web application deployment specific properties, including the servlet name and initialization parameters.

Tomcat Servlet

The SpellWeb2 server is a Java servelet that runs in the context of a web application container. This section describes the most important aspects of the Tomcat [9] (http://tomcat.apache.org/tomcat-6.0-doc/index.html) container architecture [10] (http://tomcat.apache.org /tomcat-6.0-doc/architecture/index.html) with regards to performance and resource usage. Tomcat was chosen since it widely in use and is well documented. According to the Tomcat documentation it has performance on par with other open source and commercial containers.



We use a fairly simple configuration of Tomcat and do not rely on it's more advanced features. The Tomcat server consist of the SpellWeb2 *service* that ties a *HTTP connector* to an engine that does the processing of SpellWeb2 service requests. The SpellWeb2 *context* represents the SpellWeb2 web application and is run inside a virtual *host* mapped to the servers hostname. The context comprise the class and configuration files found in the SpellWeb2/war directory.

SpellWeb2 Deployment

This section describes how to compile, configure and run the SpellWeb2 code.

SpellWeb2 consists of a server run in a web container, and a client run in the users web browser. It is also possible to run the server without using a web container by using the SpellWeb2JettyServer class as described in the next section.

Quick Setup

In summary the steps to download, compile, and run SpellWeb2 are:

1. Download and install Eclipse Java IDE and the Google Web Toolkit plugin.
2. Download the SpellWeb2 module from the CVS server at cvs.cs.princeton.edu.
3. Create a new Google Web Application project in Eclipse with the downloaded code.
4. Include DistributedSpell in the build path.
5. Configure the server by editing SpellWeb2/war/WEB-INF/web.xml
6. Specify the configuration files for the compendia to use, or reuse existing files (SpellWeb2/data has samples for yeast).
7. Do a GWT compile.
8. Make sure that the DistributedSpell class files are in SpellWeb2/war/WEB-INF/classes/edu/princeton/function, if not do ln -s ../../.. /../../../../DistributedSPELL/bin/edu/princeton/function/distspell/ in that directory.
9. Start the server:
    1. In debugging mode: enavle USE_FAKE_SEARCH and USE_FAKE_GOTERM in SpellWebBacend.java and then start SpellWeb2.java as a regular Google Web Application in Eclipse.
    2. In production mode: setup a web container and copy the war/ files to the web servers webapps/ directory.
10. Access the server by going to the servers URL: http://127.0.0.1:8888/SpellWeb2.html?gwt.codesvr=127.0.0.1:9997

Source Code and IDE Setup

To develop the code we recommend using the Eclipse Integrated Development Environment (IDE), since there is a GWT plugin for Eclipse and the GWT documentation explicitely describes how to use Eclipse for development, debugging, and testing. An alternative IDE is NetBeans. SpellWeb2 has been developed in Linux and Windows 7, and the code runs in both operating systems. Altough, all unit and integration tests should work correctly under Windows we only test under Linux.

Before downloading the code please install the GWT plugin for Eclipse, and the GWT Firefox plugin. Refer to the GWT documentation for installation details.

The SpellWeb2 code is in the spell repository on the server cvs.cs.princeton.edu. Currently Lars Ailo (larsab@cs.uit.no), Wenli (wenliz@princeton.edu) and Qian (qzhu@Princeton.EDU) all have administrator rights for the repository and can therefore grant access to other users. To download the code the IDE or a CVS client can be used. In Eclipse the code is downloaded by:

1. Create a new account at "cvs.cs.princeton.edu" and join the spell repository.
2. Start the New Project Wizard.
3. Select "Projects from CVS".
4. Create a new repository location, using host "cvs.cs.princeton.edu", repository path "/cvs", and connection type "extssh".

5. Select to use an existing module, such that you can browse the repository.
6. Select the SpellWeb2 module.
7. Select "Check out project as a project configured New Project Wizard".
8. Select HEAD.
9. Select Google, Web Application Project.

Build Path Configuratoin

SpellWeb2 uses some of the classes from the DistributedSpell project, so it is necessary to have the DistributedSpell/bin directory on the classpath when running and compiling the code. In Eclipse this can be achieved by adding DistributedSpell to the "Required projects in the build path" in the Java Build Path configuration.

SpellWeb2 forwards Spell queries to the DistributedSpell search engine, and the GoTermFinder server, so these must be running before starting SpellWeb2 (refer to the Distributed Spell deployment documentation, and GoTermFinder deployment for details). Note SpellWeb2 can also be run in debugging mode where it does not use DistributedSpell and GoTermFinder. To start in debugging mode enable USE_FAKE_SEARCH and USE_FAKE_GOTERM in SpellWebBacend.java.

Required Libraries

The server side code should run on all JVM versions 1.6 or later. Except for the GWT libraries, and DistributedSpell libraries no additional libraries are needed. For the development we have used Google App Engine Java SDK 1.3.4, and Google Web Toolkit SDK 2.0.3. We have mainly tested SpellWeb2 on Linux and there may be file specific code that does not port to other platforms.

If SpellWeb2 is configured as a Google Web Application in Eclipse all required libraries should automatically be configured. These are: App Engine SDK, GWT SDK, JRE System Library, and JUnit4 for unit testing.

Compiling

If using Eclipse the SpellWeb2 code should be automatically compiled and organized into the following output directory structure.

The server side code is a Java web application using the WAR directory structure (for details refer to [11] (http://tomcat.apache.org /tomcat-6.0-doc/appdev/deployment.html)).

- The compiled classes are in SpellWeb2/war/WEB-INF/classes
- Library files are in SpellWeb2/war/WEB-INF/lib

The GWT compiler compiles the client side code to JavaScript and writes the compiled code to SpellWeb2/war/spellweb2. It compiles a separate version of the code for different browsers. The most important files in the directory are:

- (SpellWeb2.html, which is in the SpellWeb2/war directory, is initially loaded by the browser).
- spellweb2.nocache.js: is referenced in SpellWeb2.html. It is a bootstrap file that selects the correct code to run for a given browser.
- <md5>.cache.html: contains the actual JavaScript code packed into an HTML file. The filename is the MD5 hash of the content so the files can be cached indefinitely at the user.
- <md5>.gwt.rpc: contains information about the serialized objects sent using GWT-RPC.

For more details about the client side compilation refer to the GWT documentation [12] (http://code.google.com/webtoolkit/doc/latest /DevGuideCompilingAndDebugging.html#DevGuideJavaToJavaScriptCompiler) .

Configuration Files

The SpellWeb2 server is configured by editing the SpellWeb2/war/WEB-INF/web.xml file. The web.xml file has fields for the following required parameters:

- configFile: path to a configuration files that specifies the compendia to load (described in detail in the next section)
- spellMasterHostname: host running the DistributedSpell server.
- spellMasterPort: port on which the DistributedSpell server listens for clients.
- goTermFinderHostname: host running the GoTermFinder server.
- goTermFinderPort: port on which the GoTermFinder server listens for clients.
- mySpellPrefix: location of MySpell files on the web server (typically the default value is fine)
- logLevel: logging level to use.

The client is automatically configured by querying the server. The one exception is the organism to use which is specified by the "locale=KEGG" field in the URL.

Compendium Configuration

A SpellWeb2 server can serve the compendium for many organisms. The compendium to serve are specified in a tab-delimited configuration file, where each line contains:

1. KEGG organism ID for the organism.
2. Path to a file with gene ID to systematic gene name mappings.
3. Path to a file with dataset ID to dataset filename mappings.
4. Directory where dataset files are stored
5. Path to a file with metadata retrieved from PubMed.
6. Path to a file that maps gene aliases to systematic names
7. Path to a file that maps systematic gene names to common names.

All files are tab-delimited text files. The dataset file is create by the Troilkatt pipeline. There is one line per dataset with the following format: globally unique dataset ID (integer)<tab>filename relative to the dataset directory (string).

The gene files (id->systematic, alias->systematic names, systematic -> common name) are manually created. The line format for these files is:

- id->systematic: globally unique ID (integer)<tab>systematic gene name (string)<newline>
- alias->systematic: alias (string)<tab>systematic name 1|systematic name 2|...|systematic name N<newline>
- systematic->common: systematic name<tab>common name<newline>

The metadata file is create by a script (getCitationInformation.rb in Troilkatt/src/scripts/array_utils) that takes as input a file with (dataset, PubMed ID) tuples and queries PubMed to get meta-data information including the authors, journal, and the abstract. The file has the following tab-delimited columns:

1. PubMed ID
2. Base filename.
3. Geo GSD or GSE ID.
4. Geo GPL ID.
5. Number of Microarray channels (one or two).
6. Dataset name:
7. Description:
8. Number of experiment conditions in dataset.
9. Number of genes in the dataset.
10. First author.
11. All authors.
12. Publication title.
13. Journal where the dataset results where published.
14. Year of publication.
15. Description of experiment conditions.
16. Tags: ?

In addition there must be a SpellWeb2Constants_<KEGG ID>.properties and SpellWeb2Messages_<KEGG ID>.properties files in the src/edu/princeton/function/spellweb2/client directory with compendia specific constants and messages (for an example refer to the SpellWeb2Constants_hsa.properties and SpellWeb2Messages_hsa.properties Human compendium files).

The set of property filles to use are specified in SpellWeb.gwt.xml and SpellWebSuper.gwt.xml in the following form:

```
<extend-property name="locale" values="<KEGG ID1>"/>
<extend-property name="locale" values="<KEGG ID2>"/>
```

and the default locale must be set in SpellWeb2.html:

```
<html>
  <head>
    ...
  <meta name="gwt:property" content="locale=<KEGG ID>">
  </head>
```

```
   ...
```

Finally, the mapping of KEGG identifiers to common names must be specified in OrganismIDs.java if it is not already in the list.

MySpell Directory and Other Static Files

SpellWeb2 takes as input parameter a directory where the downloadable MySpell files are stored. These files must be on a publicly accessible web server. Refer to the MySpell deployment for additional details of the directory content.

In addition the SpellWeb2/war/images directory contains the logos and other images used by SpellWeb2.

Security

Tomcat, Embedded Jetty, and other web application containers provide authentication and other security features. We use the default security settings, but configure the container such that it can access files on the local filesystem. SpellWeb2 does not use authentication.

Running SpellWeb2 in Debugging Mode

SpellWeb2 is a GWT project and must therefore have the usual GWT libraries in the class path (creating an Eclipse project includes these automatically). It also includes files from the DistribuedSpell project. Note that due to a bug in the Eclipse GWT plugin the DistribuedSpell class files are not automatically copied to the SpellWeb2 output class folder. Therefore, a link must be added manually to these by:

1. cd ./SpellWeb2/war/WEB-INF/classes/edu/princeton/function
2. ln -s ../../../../../../../DistributedSPELL/bin/edu/princeton/function/distspell/

To run in debugging mode in Eclipse the following must be done:

1. In SpellWebBackend.java change the calls to spellSearch() and spellRefinedSearch() to fakeSearch() by enabling USE_FAKE_SEARCH and USE_FAKE_GOTERM. This is necessary since AppEngine applications are not allowed to connect to other servers.
2. Make sure that SpellWeb2/war has a directory called troilkatt with a content as described in SpellWeb2/war/README
3. If necessary, in./SpellWeb2/war/WEB-INF/web.xml change the <init-param> arguments. This only needs to be done if step 1. is not done (otherwise the default files in the troilkatt directory should work).
4. In "Run Configurations" set the working directory to: "/<your workspace>/SpellWeb2/war"
5. Run SpellWeb2.java as a Web application.
6. Open http://127.0.0.1:8888/SpellWeb2.html?gwt.codesvr=127.0.0.1:9997

Running SpellWeb2 in Production Mode

To run SpellWeb2 in production mode create a war file and deploy it in a container such as Apache or Tomcat, or use SpellWeb2JettyServer. To use Tomcat:

1. In SpellWeb2/war run "zip -r spellweb2.war *". This will create a zip file with the content of the SpellWeb2/war directory.
2. Copy the spellweb2.war file to the *webapps/* directory used by tomcat.
3. Tomcat will then automatically unpack the war file to the webapps/spellweb2 directory.
4. Verify that SpellWeb2 started correctly by accessing the <hostname>:<your port>/SpellWeb2.html file.

SpellWeb2JettyServer Deployment

his section describes how to compile, configure and run the SpellWeb2 without using a web container. To deploy SpellWeb2 using a web container refer to the previous section.

Source Code and IDE Setup

The SpellWeb2JettyServer is a seperate module, that is downloaded using the same steps as above expect that in the last step a regular Java project should be selected (and in step 6. the SpellWeb2JettyServer module should be selected).

Quick Setup

In summary the steps to download, compile, and run SpellWeb2JettyServer are:

1. Download and GWT compile SpellWeb2.
2. Download and install the Jetty web server.
3. Download the SpellWeb2JettyServer module from the CVS server at cvs.cs.princeton.edu.
4. Create a new Java project in Eclipse with the downloaded code.
5. Include SpellWeb2 in the build path.
6. Add gnu-getopt.jar file, gwt-servlet.jar, gwt-user, jetty-client.jar, jetty-continuation.jar, jetty-http.jar, jetty-io.jar, jetty-security.jar, jetty-server.jar, jetty-servlet.jar, jetty-servlets.jar, and jetty-util.jar to the build path.
7. Specify the configuration files for the compendia to use, or reuse existing files (SpellWeb2/data has samples for yeast).
8. Configure the server by editing SpellWeb2/war/runStandalone.sh
9. Start the server by executing SpellWeb2/war/runStandalone.sh
10. In the web browser goto http://localhost:9007/SpellWeb2.html

Source Code and IDE Setup

Refer to the instructions in the previous section for how to download the SpellWeb2JettyServer source code from the CVS server, but note that instead of a Google Web Application Project a regular Java project should be created.

To use SpellWeb2Jetty the Jetty Web Server must also be downloaded and installed (refer to [13] (http://www.eclipse.org/jetty/) for details).

Build Path Configuration

SpellWeb2JettyServer uses the classes from SpellWeb2 (and hence also DistributedSpell), so the SpellWeb2 project should be included in the build path.

The GWT and Jetty libraries must be set manually. These are: gnu-getopt.jar, gwt-servlet.jar, gwt-user.jar, jetty-client.jar, jetty-continuation.jar, jetty-http.jar, jetty-io.jar, jetty-security.jar, jetty-server.jar, jetty-servlet.jar, jetty-servlets.jar, and jetty-util.jar. The gwt-* files are from GWT, and the jetty-* files are from the Jetty distribution. We have used Jetty distribution 7.0.1 for development and testing.

Compiling

If correctly set up Eclipse should automatically compile the code and output the binaries to SpellWeb2JettyServer/bin.

Configuration

The standalone server does not use the web.xml configuration file. Instead all arguments are set as command line arguments. For convenience there is a script to start the server where these can be set: SpellWeb2/war/runStandalone.sh.

Refer to the previous section for how to configure a compendium.

Running SpellWeb2JettyServer in Debugging Mode

To run in debugging mode Eclipse:

1. Make sure SpellWeb2 is up to date by doing a "GWT compile" on the SpellWeb2 project
2. Specify the required arguments (configuration file, mySpell directory, SpellMaster host and port, GoTermFinder host and port) in "Run Configurations". Note that if "fakeSearch" is used the SpellMaster and GoTermFinder ports can be set arbitrarily.
3. Specify the working directory as: "/<your workspace>/SpellWeb2/war".
4. Run as Java application.
5. In the browser goto http://localhost:9007

Note that SpellWeb2JettyServer also opens a Java RMI port on 9006. This port is used by MySpell.

Running SpellWeb2JettyServer in Production Mode

To run in production run the java program with the following arguments (or modify and use SpellWeb2/war/runStandalone.sh):

- -classpath <all the required jar files (as described above)>

- -Duser.dir=<deployment dir>/SpellWeb2/war (if the command is not run the SpellWeb2/war directory)
- edu.princeton.function.spellweb2.standalone.EmbeddedJettyServer
- -c <compendium configuration file>
- -s <path to directory with mySpell files> Note! The path is relative to the SpellWeb2/war directory.
- -m <SpellMaster hostname>:<Spell Master port>
- -g <GoTermFinder hostname>:<GoTermFinder port>
- -p <this servers web server port>
- -r <this servers Java RMI port>
- -d <debugging level to use (severe, warning, info, or trace)>

Testing

See also List of bugs

Status

1. Unit tests implemented
2. All unit tests pass
3. Integration tests "implemented" (these are manual and are described below)
4. All integration tests pass

Unit Tests

We use EclEMMA [14] (http://www.eclemma.org/) (EMMA [15] (http://emma.sourceforge.net/userguide_single/userguide.html) for Eclipse) for code coverage testing, and the JUnit [16] (http://www.junit.org/) [17] (http://junit.sourceforge.net/doc/cookbook /cookbook.htm) framework for unit testing (JUnit is built in Eclipse). For additional details about GWT application testing, refer to the GWT Testing guide [18] (http://code.google.com/webtoolkit/doc/latest/DevGuideTesting.html) .

The unit test code is in SpellWeb2/test. The unit tests are for the server side code only and they assume that fakeSearch() and fakeGoTermSearch() are used. The server side unit tests include exception testing, but some IOExceptions are not triggered since there is no communication with a server. The client side is not unit tested since most of the code is for user input handling and GWT-RPC communication (generating user input events in GWT is not straight forward). However, this code is tested as part of the integration tests described below.

Integration Tests

Due to the interactive nature of SpellWeb2 a lot of the testing must be done manually. This section describes the tests for each page.

Test Setup

To simplify the testing of SpellWeb2, it can be run in a mode where it does not send search and go term queries to the DistributedSpell and GoTermFinder servers. To enable this mode enable the global variables USE_FAKE_SEARCH and USE_FAKE_GOTERM in SpellWebBacend.java. By enabling these search and go term queries will be handled by the fakeSearch() and fakeGoTermSearch() functions. These functions returns a predefined result where the genes and datasets are sorted by their ID and weight/scores are assigned descending by 0.01 starting from 1.00. The returned ranking and scores for genes and datasets is therefore independent of the query and obviously not correct.

The tests use three compendia selected to test different aspects of the system:

- A micro compendium is used to test the correctness of the visualizations.
  - It has yeast gene names, with the ten first gene IDs: ctr9, med2, ent1, crz1, spt6, dom34, sey1, mlp2, bre1 and reg1. Using the fakeSearch() function the resulting genes will therefore be sorted in this order, which is the same as returned by the old spell for a real search for the genes ctr9 and med2.
  - There are two highly yeast datasets for each type of data: Brem05 (dual channel), Bulik03 (dual channel), Caba05 (single channel), and Bernstein00_HDACsin3sap30ume6hda1hos2hos3 (single channel). These are among the datasets that are mostly correlated with the "ctr9 med2" query.
- A yeast compendium with 115 datasets used to compare the gene and dataset information correctness against the old Spell (http://imperio.princeton.edu:3000/yeast).
- A human compendium with 724 datasets used to test the scalability of SpellWeb2.

To configure SpellWeb2 to use the three compendium the SpellWeb2.gwt.xml and SpellWebSuper.gwt.xml files must contain the following three property fields:

```
<extend-property name="locale" values="sce"/>
<extend-property name="locale" values="hsa"/>
<extend-property name="locale" values="debug"/>
```

In addition the spellweb2-integration-web.config (in SpellWeb2/data/config) file must be modified to specify the location of the organism specific configuration and data files. The organism specific configuration files are in the yeast-micro/, yeast-all/, and human-all/ subdirectories in SpellWeb2/data/config.

The micro compendia data files are in SpellWeb2/data/input/yeast-micro. The files for the yeast and human compendia are not in the CVS but can be found in the troilkatt database TODO: specify location and how to get them.

To run the tests modify and execute the runIntegrationTests.sh script in SpellWeb2/war/. This will start a standalone server on localhost which can be accessed using the URL: http://localhost:9007/SpellWeb2.html?locale=debug (or by substituring localhost with the hostname if the browser and server are on different machines). Note that about 7.4GM memory is required for the datasets, so the machine running the backend server should have at least 8GB DRAM.

Navigation

These test are to test the browser navigation of SpellWeb2:

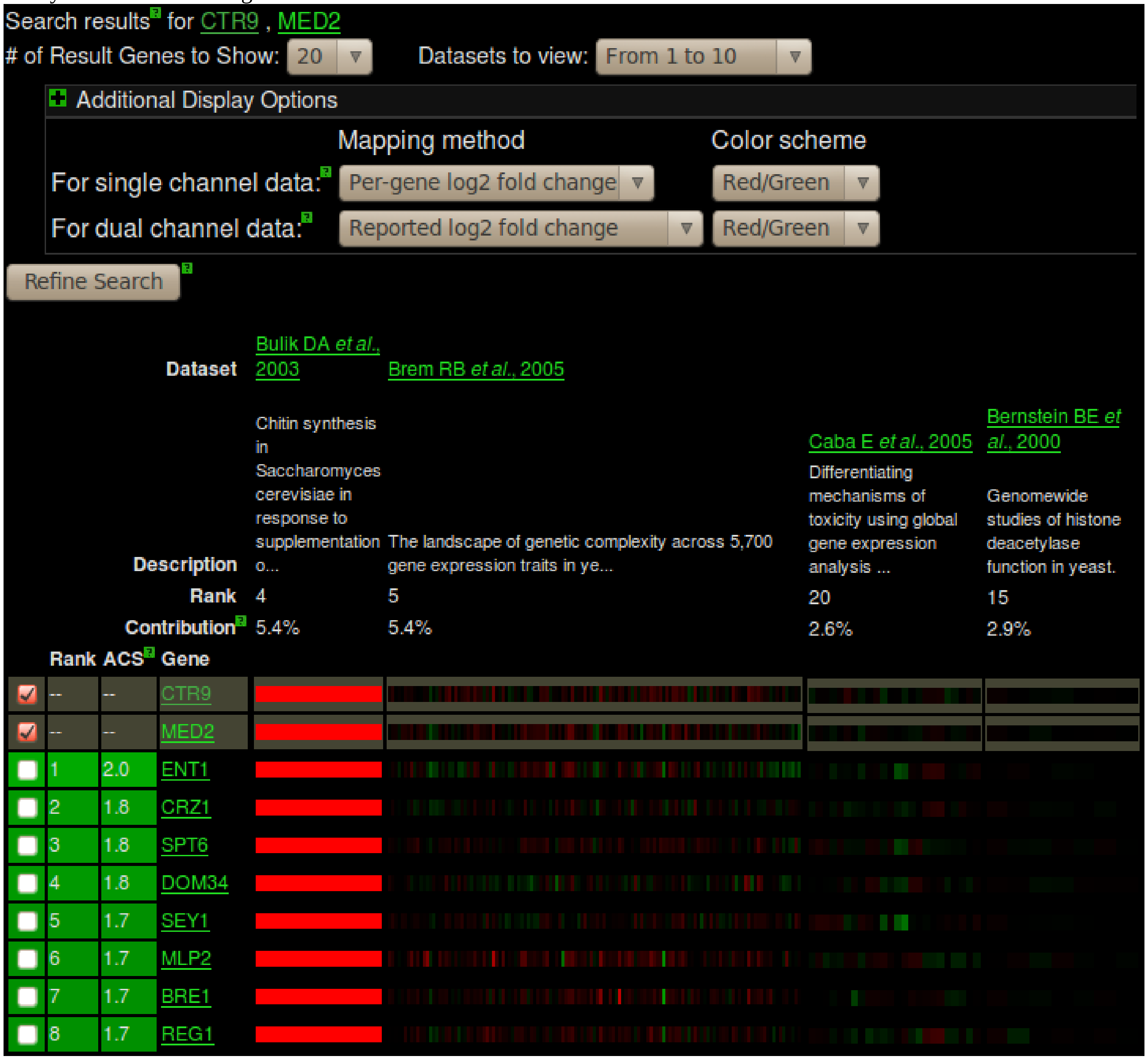
1. Open SpellWeb2, and verify that the main search page is displayed for "debug" compendia.
    1. Verify that the "Search other organisms" panel contains "yeast" and "human"
2. Click on the "human" link on the organism panel
    1. Verify that the "Search other organisms" panel contains "debug" and "yeast"
    2. Enter the query **brca1 brca2**, click search, and verify that the search results page is shown.
    3. Click on the help icon, and verify the results, for the: "Search result" label, "Single-channel" list, "Dual-channel" list, "Refine Search" button, "Contribution" row header, "ACS" column header, and the "Go Term Enrichment" table label.
    4. Click the browser back button to get back to the main search page where the query field should contain "brca1 brca2".
    5. Click the browsers forward button to get back to the search results page.
    6. Click on one of the "Datasets" links to open the dataset details view.
    7. Click on the "Search Results" link to get back to the search results.
    8. Click on one of the expression level heatmaps to open the single expression details view.
    9. Click the browsers back button to get to the search result page.
    10. Click the "Private Search" link in the navigation panel to get to private search page.
    11. Click the "Public Dataset Listing" link in the navigation panel to get to public datasets page.
    12. Click the "Show Expression Levels" link in the navigation panel to get to gene expression values page.
    13. Enter the query **brca1 brca2**, click search, and verify that the expression levels page is shown.
    14. Click on the "About" page to get to the about page.
    15. Click on the "Search Results" link to get back to the search results.
    16. Click twice on the browsers back button to get back to the expression levels page.
    17. Click the "New Search" link for the main search page.
    18. REMOVED FROM GUI: (Click the "Include private data in search" link for the private data search page.)
3. Click on the "yeast" link in the organism panel and verify that the yeast search page is displayed.
    1. Verify that the "Search other organisms" panel contains "debug" and "human"
    2. Enter the query **CTR9 MED2**, click search, and verify that the search results page is shown.
    3. Click on one of the "Datasets" links to open the dataset details view.
    4. Click the "Public Dataset Listing" link in the navigation panel to get to public datasets page.
    5. Click the "Private Search" link in the navigation panel to get to public datasets page.
    6. Click the "Show Expression Levels" link in the navigation panel to get to gene expression values page.
    7. Enter the query **CTR9 MED2**, click list, and verify that the expression levels page is shown.
    8. Click on the "About" page to get to the about page.
    9. Click on the "Search Results" link to get back to the search results.
    10. Click twice on the browsers back button to get back to the expression levels page.

New Search and Search Results Pages

Micro compendium:

1. Search result for query **CTR9 MED2**, with "#gene results" set to "10" and "#dataset results" set to "4"

1. Verify that the following four datasets are shown in the order: Bulik03, Brem05, Caba05, and Bernstein00.
2. Expand the "Additional Display Options" tab and verify that the mapping method is "Per-gene log2 fold change" for single channel data, and "Centred per-gene fold change" for dual channel data.
3. Change mapping method for "dual channel data" from to "Centered per-gene fold change" to "Reported log2 fold change".
4. Verify that the visualized genes look like



Search results for CTR9 , MED2

# of Result Genes to Show: 20 Datasets to view: From 1 to 10

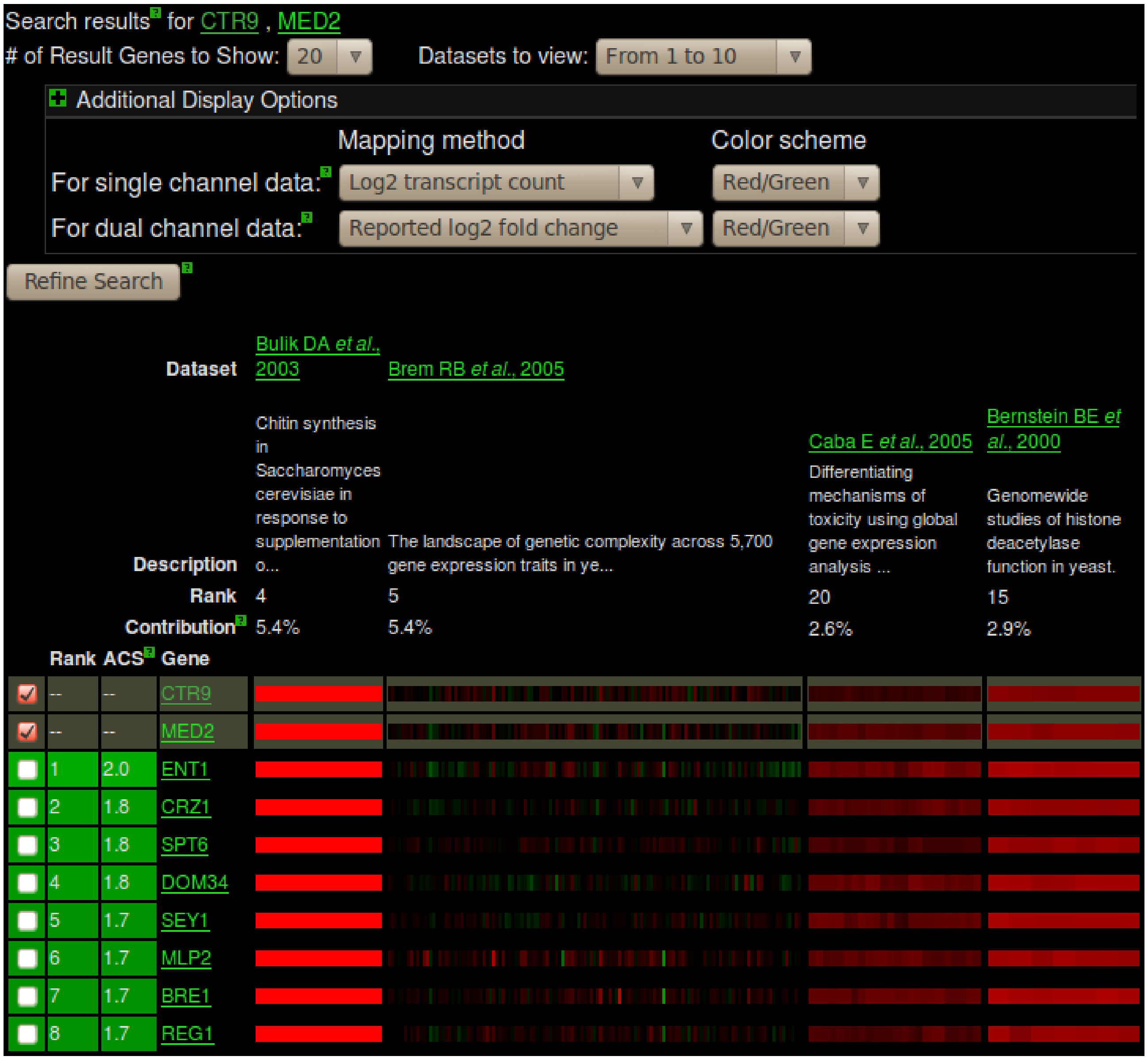
Additional Display Options

	Mapping method	Color scheme
For single channel data:	Per-gene log2 fold change	Red/Green
For dual channel data:	Reported log2 fold change	Red/Green

Refine Search

			Dataset	Bulik DA *et al.*, 2003	Brem RB *et al.*, 2005	Caba E *et al.*, 2005	Bernstein BE *et al.*, 2000
			Description	Chitin synthesis in Saccharomyces cerevisiae in response to supplementation o...	The landscape of genetic complexity across 5,700 gene expression traits in ye...	Differentiating mechanisms of toxicity using global gene expression analysis ...	Genomewide studies of histone deacetylase function in yeast.
			Rank	4	5	20	15
			Contribution	5.4%	5.4%	2.6%	2.9%
	Rank	ACS	Gene				
✓	--	--	CTR9				
✓	--	--	MED2				
☐	1	2.0	ENT1				
☐	2	1.8	CRZ1				
☐	3	1.8	SPT6				
☐	4	1.8	DOM34				
☐	5	1.7	SEY1				
☐	6	1.7	MLP2				
☐	7	1.7	BRE1				
☐	8	1.7	REG1				

5. Click on the expression values for CTR9 in Bulik03 and verify that the reported values are: "5.34, 5.44, 5.49, 5.60, 5.58, 5.64, 5.65, 5.80, 5.89, 5.78, 5.44" (these are the values in the dataset file).
6. Click on the expression values for CTR9 in Caba05 and verify that the reported values are *similar* to: " -0.17, 0.03, 0.04, -0.06, 0.06, 0.32, 0.09, -0.15, 0.02, 0.14, -0.28, 0.01, 0.02, -0.19, -0.00, -0.09, 0.13, 0.14, 0.16, -0.29, -0.16, 0.00, 0.11, -0.06".
7. Change mapping method for "single channel data" from "Per-gene log2 fold change" to "Log2 transcript count" and verify that the visualized genes look similar to



Search results for CTR9 , MED2

# of Result Genes to Show: 20 Datasets to view: From 1 to 10

Additional Display Options

	Mapping method	Color scheme
For single channel data:	Log2 transcript count	Red/Green
For dual channel data:	Reported log2 fold change	Red/Green

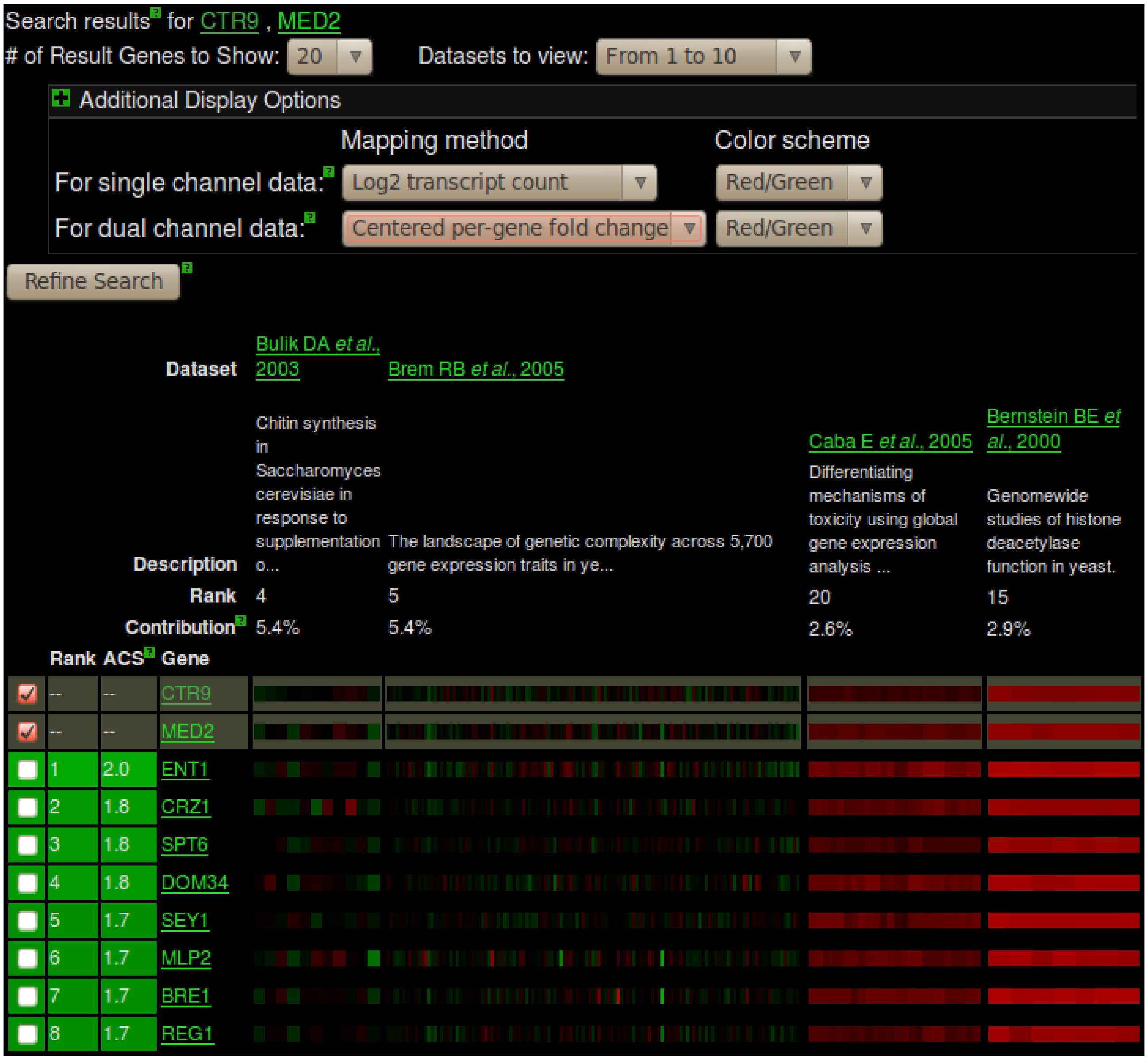
Refine Search

	Bulik DA *et al.*, 2003	Brem RB *et al.*, 2005	Caba E *et al.*, 2005	Bernstein BE *et al.*, 2000
Description	Chitin synthesis in Saccharomyces cerevisiae in response to supplementation o...	The landscape of genetic complexity across 5,700 gene expression traits in ye...	Differentiating mechanisms of toxicity using global gene expression analysis ...	Genomewide studies of histone deacetylase function in yeast.
Rank	4	5	20	15
Contribution	5.4%	5.4%	2.6%	2.9%

	Rank	**ACS**	**Gene**
✓	--	--	CTR9
✓	--	--	MED2
	1	2.0	ENT1
	2	1.8	CRZ1
	3	1.8	SPT6
	4	1.8	DOM34
	5	1.7	SEY1
	6	1.7	MLP2
	7	1.7	BRE1
	8	1.7	REG1

(datasets Caba05 and Bernstein00 should have been changed).

8. Click on the expression values for CTR9 in the Caba05, and verify that the single expression values are "2.94, 3.37, 3.38, 3.17, 3.43, 4.11, 3.5, 2.97, 3.35, 3.63, 2.71, 3.33, 3.35, 2.9, 3.29, 3.10, 3.62, 3.63, 3.69 2.70, 2.95, 3.30, 3.55, and 3.17" (these are the values in the dataset file).
9. Change mapping method for "dual channel data" from "Reported log2 fold change" to "Centered per-gene fold change" and verify that the visualized genes look similar to


Search results for CTR9 , MED2

# of Result Genes to Show: 20 Datasets to view: From 1 to 10

Additional Display Options

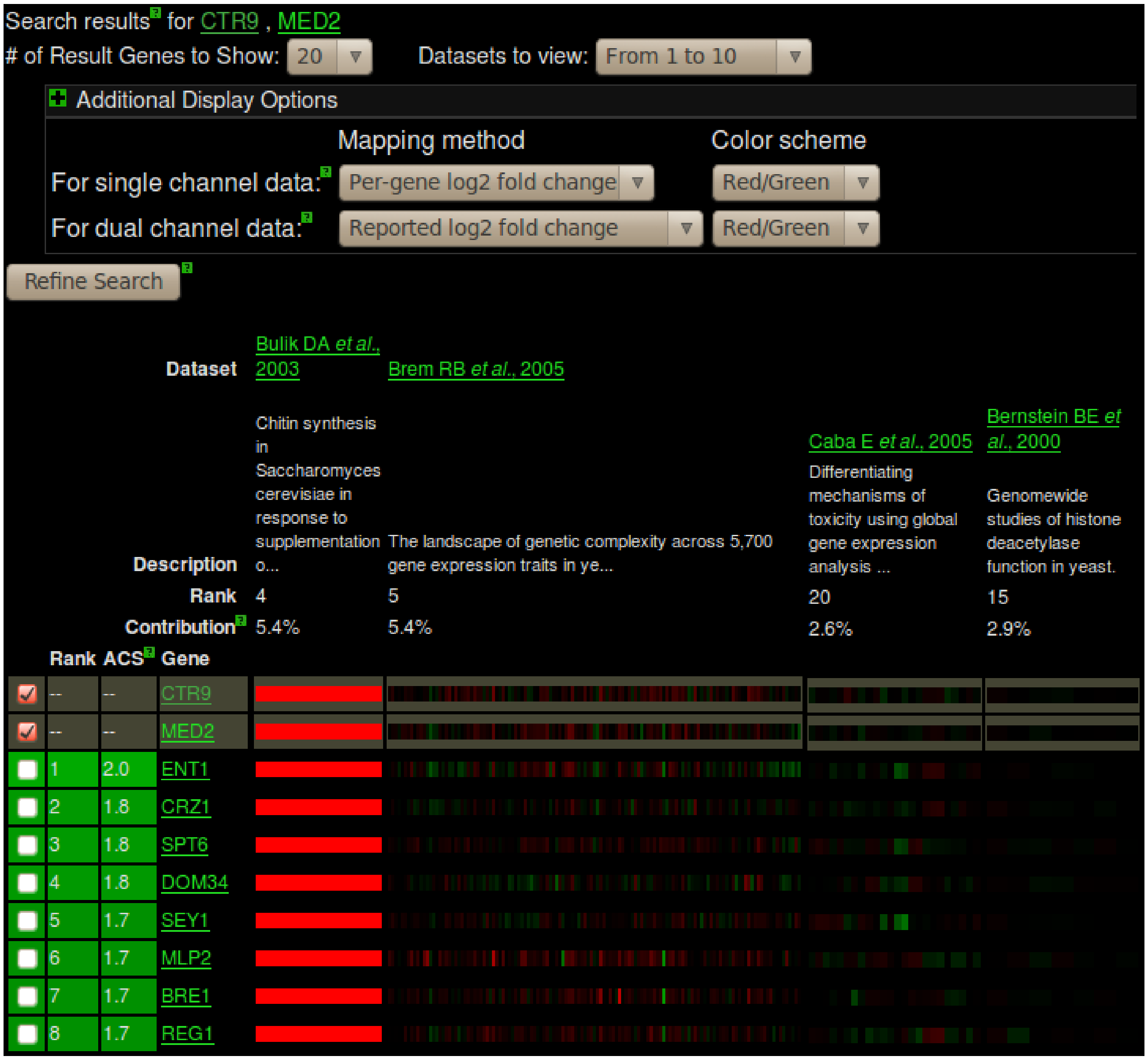
	Mapping method	Color scheme
For single channel data:	Log2 transcript count	Red/Green
For dual channel data:	Centered per-gene fold change	Red/Green

Refine Search

		Dataset	Bulik DA *et al.*, 2003	Brem RB *et al.*, 2005	Caba E *et al.*, 2005	Bernstein BE *et al.*, 2000
		Description	Chitin synthesis in Saccharomyces cerevisiae in response to supplementation o...	The landscape of genetic complexity across 5,700 gene expression traits in ye...	Differentiating mechanisms of toxicity using global gene expression analysis ...	Genomewide studies of histone deacetylase function in yeast.
		Rank	4	5	20	15
		Contribution	5.4%	5.4%	2.6%	2.9%
Rank	ACS	Gene				
--	--	CTR9				
--	--	MED2				
1	2.0	ENT1				
2	1.8	CRZ1				
3	1.8	SPT6				
4	1.8	DOM34				
5	1.7	SEY1				
6	1.7	MLP2				
7	1.7	BRE1				
8	1.7	REG1				

(datasets Bulik03 and Brem05 should have been changed)

10. Click on the expression values for CTR9 in Bulik03 and verify that the values are *similar* to: "-0.26, -0.16, -0.11, -0.00, -0.02, 0.04, 0.05, 0.20, 0.29, 0.18, -0.16".
11. Change the mappings bakc to "Per-gene log2 fold change" and "Reported log2 fold change", and verify that the visualized genes look like


Search results for CTR9 , MED2

# of Result Genes to Show: 20   Datasets to view: From 1 to 10

Additional Display Options

	Mapping method	Color scheme
For single channel data:	Per-gene log2 fold change	Red/Green
For dual channel data:	Reported log2 fold change	Red/Green

Refine Search

	Dataset	Bulik DA *et al.*, 2003	Brem RB *et al.*, 2005	Caba E *et al.*, 2005	Bernstein BE *et al.*, 2000
	Description	Chitin synthesis in Saccharomyces cerevisiae in response to supplementation o...	The landscape of genetic complexity across 5,700 gene expression traits in ye...	Differentiating mechanisms of toxicity using global gene expression analysis ...	Genomewide studies of histone deacetylase function in yeast.
	Rank	4	5	20	15
	Contribution	5.4%	5.4%	2.6%	2.9%

	Rank	ACS	Gene
✓	--	--	CTR9
✓	--	--	MED2
	1	2.0	ENT1
	2	1.8	CRZ1
	3	1.8	SPT6
	4	1.8	DOM34
	5	1.7	SEY1
	6	1.7	MLP2
	7	1.7	BRE1
	8	1.7	REG1

(datasets Caba05 and Bernstein00 should have been changed).

12. Change color scheme for "single channel data" from "Red/ Green" to "Yellow/ Blue" and verify that the color of datasets Caba05 and Bernstein00 changes.
13. Change color scheme for "single channel data" back to "Red/ Green" and verify that the color of datasets Caba05 and Bernstein00 changes.
14. Change color scheme for "dual channel data" from "Red/ Green" to "Yellow/ Blue" and verify that the color of datasets Bulik03 and Brem05 changes.
15. Change color scheme for "dual channel data" back to "Red/ Green" and verify that the color of datasets Bulik03 and Brem05 changes.
16. Verify that the Datasets links are "Bulik DA et al., 2003", "Brem RM et al., 2005", "Caba et al., 2005", and "Bernstein BE et al., 2000"
17. Verify that the Descriptions are "Chitin synthesis in...", "The landscape of genetic complexity...", "Differentiating mechanisms of toxicity...", and "Genomewide studies of histone..."
18. Verify that the dataset ranks are 1, 2, 3, and 4.
19. Verify that the dataset contributions are 1.0, 0.99, 0.98, and 0.97
20. Click on the CTR9 gene link and verify that gene information for the CRT9/YOL145C gene is shown in a separate tab.
21. Close the gene page tab to return to the search result page.
22. Click on the first GOTerm in the GO Term Encrichment table, and verify that the accession for "GO:0005853" (cellular component) is shown in a separate tab.
23. Close the tab.
24. Click on the gene link for the first GoTerm and verify that information for gene "EFB1/YAL003W" is displayed in a separate tab.

2. Go to the "Show Expression Levels" page and enter the gene name **CTR9** (this test is to ensure that the display options are set

correctly cross page).

1. Verify that the mapping method is "Per-gene log2 fold change" for single channel data, and "Centred per-gene fold change" for dual channel data.
2. Click on the expression values for CTR9 in Bulik03 and verify that the reported values are *similar* to: "-0.26, -0.16, -0.11 -0.00, -0.02, 0.04, 0.05, 0.20, 0.29, 0.18, -0.16".
3. Click on the expression values for CTR9 in Caba05 and verify that the reported values are *similar* to: " -0.17, 0.03, 0.04, -0.06, 0.06, 0.32, 0.09, -0.15, 0.02, 0.14, -0.28, 0.01, 0.02, -0.19, -0.00, -0.09, 0.13, 0.14, 0.16, -0.29, -0.16, 0.00, 0.11, -0.06".
4. Change mapping method for "single channel data" from "Per-gene log2 fold change" to "Log2 transcript count" and verify that theexpression values for CTR9 in the Caba05 are "2.94, 3.37, 3.38, 3.17, 3.43, 4.11, 3.5, 2.97, 3.35, 3.63, 2.71, 3.33, 3.35, 2.9, 3.29, 3.10, 3.62, 3.63, 3.69, 2.70, 2.95, 3.30, 3.55, and 3.17, 3.547, and 3.17" (these are the values in the dataset file).
5. Change mapping method for "dual channel data" from "Centered per-gene fold change" to "Reported log2 fold change" and verify that the expression values for CTR9 in Bulik03 are: "5.34, 5.44, 5.49, 5.60, 5.58, 5.64, 5.65, 5.80, 5.89, 5.78, 5.44" (these are the values in the dataset file).

Yeast compendium:

1. Gene name mappings (check these by doing a search and then checking the genes in the "Search result for: " panel).
   1. *Common name*: query **CTR9 MED2 ENT1 CRZ1 SPT6 DOM34 sey1 mlp2 bre1 reg1** maps to "CTR9 MED2 ENT1 CRZ1 SPT6 DOM34 SEY1 MLP2 BRE1 REG1".
   2. *Systematic name*: query **YOL145C YDL005C YDL161W ynl027w YGR116w YNL001w YOR165W YIL149C YDL074C YDR028C** maps to "CTR9 MED2 ENT1 CRZ1 SPT6 DOM34 SEY1 MLP2 BRE1 REG1".
   3. *Valid alias query*: query **CDP1 YDL005C hal8 CRE2** maps to "CTR9 MED2 CRZ1 SPT6".
   4. *Only invalid names*: query **CRF1 foobar foobar** gives the following errors "CRF1 (maps to multiple symbols) foorbar(not found) foobar(not found)", and that no results are shown.
   5. *Many invalid names*: query **HAL8 REG1 TCN1 CRF1 foobar YOL145C SEY1 crz1**maps to the valid genes: "CRZ1 (HAL8 is an alias), REG1 (common name), CTR9 (YOL145C is the systematic name), SEY1 (common name)", and the following invalid names: "TCN1 (another alias to CRZ1), CRF1 (maps to multiple symbols), foobar (invalid name), CRZ1 (duplicate name)".
   6. *Valid separators*: query **CTR9 MED2,ENT1, CRZ1|SPT6 | DOM34/ SEY1,|/MLP2** maps to "CTR9 MED2 ENT1 CRZ1 SPT6 DOM34 SEY1 MLP2".
2. Search result for **CTR9 MED2** with number of genes and datasets shown both set to 30.
   1. Click on the "Datasets" link for dataset Brem05, and verify that citation is "Brem RB, Kruglyak L. The landscape of genetic complexity across 5,700 gene expression traits in yeast. Proceedings of the National Academy of Sciences of the United States of, 2005.", PubMedID is "15659551", short description is "The landscape of genetic complexity across 5,700 gene expression traits in yeast", #conditions is 131, and full description is "Many studies have identified quantitative trait loci...may shed light on the evolution of quantitative traits".
   2. Click on the "PubMed ID" link and verify that the correct PubMed page is shown in a separate tab.
   3. Close the PubMedID tab, and click on the browsers back button to return from the datasets detail window to the search result.
   4. Click on the "MED2" gene link in the "Search results" panel and verify that the gene information for the MED2 gene is shown in a separate tab.
   5. Close the tab and click on the "ATP8" gene link and verify that the gene information for the ATP8/Q0080 gene is shown in a separate tab.
3. Select the 2nd (AI3), 3rd (AI4), 5th (AI5_BETA) and 7th (ATP6) ranked gene, click refined search, and verify that the search results are for genes: "COX1 AI1 AI3 AI4 AI5_BETA ATP6", and that the selected genes are "COX1 AI1 AI2 AI3 AI4 and AI5/ALPHA" (due to the fakeSearch function these will differ from the query).
   1. Deselect the AI1, AI2, AI3 and AI4 ranked gene and click refined search, then verify that the search results are for genes: "COX1 AI5_ALPHA"
   2. Set "#Genes to show" to "1", then to "20", select "COX1", "AI5_ALPHA" and "COB", click refined search and verify that the search result is "COX1 AI5_ALPHA COB".
   3. Deselect all three genes, and click refine search and verify that nothing happens.

Human compendium:

Note that these tests assume that the constants DEFAULT_DATASETS_PER_PAGE = DEFAULT_GENES_TO_SHOW = 30, and DEFAULT_GENES_TO_RECEIVE = DEFAULT_DSETS_TO_RECEIVE = 100.

1. Gene name mappings
   1. *Common name*: query **BRCA1 Brca2** maps to "BRCA1 BRCA2".
   2. **Note!**: humans do not have common and systematic names, since both are identical.

   3. *Valid alias query*: query **RNF53 fancd1** maps to "BRCA1 BRCA2".
   4. *Invalid names*: query **facd QWERTY** gives a "No valid genes" error message and the error label shows that: "facd(multiple symbols), qwerty(not found)".
   5. *Many invalid names*: query **facd fancd1 QWERTY BRCa1 RNF53** maps to "BRCA1 BRCA2" and the invalid genes "facd(multiple symbols), qwerty(not found). rfn53(duplicate of BRCA1).
2. Search results for query **BRCA1 BRCA2**, and gene results to show set to "30" and dataset results to "10".
   1. Verify that the "Search results for" is "BRCA1, BRCA2".
   2. Verify that the the rank of the first "2" genes is "--", "1" for the 3rd gene, "2" for the 4th gene, and so on.
   3. Verify that the ACS of the first gene is 1.00, the second ACS is 0.99, and so on.
   4. Verify that the "# Genes to show" text box is set to "30", and that the number of genes label is "of 24410 genes.".
   5. Verify that the "# Dataset results to show" text box is set to "30", and that number of datasets label is "of 724 datasets."
   6. Click on the "Datasets" link for dataset "Boni JP et al., 2005", and verify that citation is "Boni JP, Leister C, Bender G, Fitzpatrick V, Twine N, Stover J, Dorner A, Immermann F, Burczynski ME. Population pharmacokinetics of CCI-779: correlations to safety and pharmacogenomic responses in patients with advanced renal cancer.. Clinical pharmacology and therapeutics, 2005", PubMedID is "15637533", short description is "Population pharmacokinetics of CCI-779: correlations to safety and pharmacogenomic responses in patients with advanced renal cancer.", #conditions is 103, and full description is "OBJECTIVE: Our objective was to estimate the pharmacokinetic parameters of CCI-779..These transcripts represent potential biomarkers of CCI-779 exposure in peripheral blood".
   7. Click on the "PubMed ID" link and verify that the correct PubMed page is displayed in a separate tab.
   8. Close the PubMed tab and click the browser back button to return to the search results page.
   9. Click on the result gene "BRCA2" gene link and verify that the gene information for the gene is shown in a separate tab.
   10. Close the tab to return to the search result page.
   11. Change number of genes to show to "20", then verify that 20 genes are shown, and that the ACS numbers decrease by 0.01 for each gene.
   12. Reduce number of genes to show to "1", then verify that one gene is shown.
   13. Increase number of genes to show to "23", then verify that 23 genes are shown and that the that the ACS numbers decrease by 0.01 for each gene.
   14. Increase number of genes to show to "50", then verify that 50 genes are shown and that the that the ACS numbers decrease by 0.01 for each gene.
   15. Increase number of genes to show to "125", then verify that 125 genes are shown and that the that the ACS numbers decrease by 0.01 for each gene.
   16. Increase number of genes to show to "150", then verify that 150 genes are shown and that the that the ACS numbers decrease by 0.01 for each gene.
   17. Decrease number of genes to show to "50", then verify that 50 genes are shown and that the that the ACS numbers decrease by 0.01 for each gene.
   18. Verify that setting number of genes to show to "100000" returns an error, that the number of genes shown does not change is still "50" and that "50" genes are still shown.
   19. Verify that setting number of genes to show to "-1" returns an error, that the number of genes shown does not change is still "50" and that "50" genes are still shown.
   20. Verify that setting number of genes to show to "1F" returns an error, that the number of genes shown does not change is still "50" and that "50" genes are still shown.
   21. Verify that the Contribution of the first datasets is 1.00, the second contribution is 0.99, and so on.
   22. Change number of dataset results to show to "5", then verify that 50 genes and 5 datasets are shown, and the the dataset contributions decrease by 0.01.
   23. Increase the number of datasets to show to "8", then verify that 8 datasets are shown, and the the contribution decrease by 0.01.
   24. Increase the number of datasets to show to "50", then verify that 50 genes and 50 datasets are shown, and the the contributions decrease by 0.01.
   25. Increase the number of datasets to show to "125", then verify that 50 genes and 125 datasets are shown, and the the contributions decrease by 0.01.
   26. Increase the number of datasets to show to "150", then verify that 50 genes and 125 datasets are shown, and the the contributions decrease by 0.01.
   27. Decrease the number of datasets to show to "25", then verify that 50 genes and 25 datasets are shown, and the the contributions decrease by 0.01.
   28. Set datasets to show to "0" and verify that no datasets are shown.
   29. Set datasets to show back to "25".
   30. Set datasets to show to "-1" and verify that an error is displayed and that there are still 25 datasets shown.
   31. Set datasets to show to "10 10" and verify that an error is displayed and that there are still 25 datasets shown.
   32. Change number of genes to show to "20", then verify that 20 genes and 25 datasets are shown, and that the ACS numbers decrease by 0.01 for each gene.
   33. Reduce number of genes to show to "1", then verify that one gene is shown.
   34. Increase number of genes to show to "125", then verify that 125 genes and 25 datasets are shown and that the that the ACS

numbers decrease by 0.01 for each gene.

   35. Increase number of genes to show to "200", then verify that 200 genes are shown and that the that the ACS numbers decrease by 0.01 for each gene.
   36. Decrease number of genes to show to "50", then verify that 50 genes and 25 datasets are shown and that the that the ACS numbers decrease by 0.01 for each gene.

3. Select the 2nd (a2mp), 3rd (a3galt2), 5th (a4gnt) and 7th (aacp) ranked gene and click refined search.
    1. Verify that the search results are for genes: "A1BG A2M A2MP A3GALT2 A4GNT AACP"
    2. Verify that 25 datasets are shown, and the the contribution decrease by 0.01 starting from 1.00.
    3. Verify that 50 genes are shown and that the score decrease by 0.01 starting from 1.00.
4. Repeated search test
    1. Click on the new search link, and enter the query **brca1**, and gene results to show set to "30" and dataset results to "10". Then verify that 30 genes and 10 datasets are shown.
    2. Click on the new search link, and enter the query **brca1**, and gene results to show set to "20" and dataset results to "20". Then verify that 20 genes and 20 datasets are shown.
    3. Click on the new search link, and enter the query **brca1**, and gene results to show set to "125" and dataset results to "150". Then verify that 125 genes and 150 datasets are shown.
5. Scalability tests
    1. Set dataset results to show to "724" and verify that "20" genes and "724" datasets are shown.
    2. Set genes to view to "24410" and verify that "24410" genes and "724" datasets are shown.

Private Search Page

1. Verify that the "Required software", "Alternative to Java Web Start", "Sample datasets", and "Problems" expansion tabs work correctly.

The remaining links are tested as part of MySpell integration testing.

Public Dataset Listing Page

Yeast Compendia:

1. Verify that there are "30" datasets per page, that To page is "1/4", and that the last dataset shown is "Duvel K et al., 2003".
2. Increase number of datasets to 60, verify that To page is "1/2", and that the last dataset shown is "Hughes TR et al., 2000" with 8 conditions.
3. Decrease datasets per page to "10", verify that ten datasets are shown and that To page is "1/12", and that the *two* last dataset shown are"Brauer MJ et al., 2005".
4. Change "To page" to "7" and verify that the first dataset is "Ideker T et al., 2001".
5. Change "To page" to "0", verify that an error is displayed and that the datasets shown do not change, and that the to page remains "7".
6. Change "To page" to "-1", verify that an error is displayed and that the datasets shown do not change, and that the to page remains "7".
7. Change "To page" to "ads3", verify that an error is displayed and that the datasets shown do not change, and that the to page remains "7".
8. Change "To page" to "5 6", verify that an error is displayed and that the datasets shown do not change, and that the to page remains "7".
9. Click the forward (>>) button two times, and verify that the "To page" is "9" and that the first dataset is "Prinz S et al., 2004".
10. Click the back (<<) button two times, and verify that the "To page" is "7" and that the first dataset is "Ideker T et al., 2001".
11. Change datasets to show to "15", verify that 15 datasets are shown and that To page is "5/8", and that the last dataset shown is "Ogawa N et al., 2000".
12. Change "To page" to "1", then click on the back (<<) button and verify that nothing happens.
13. Change "To page" to "8" and verify that N datasets are shown and that the first dataset is "Williams RM et al., 2002".
14. Click on the forward (>>) button and verify that nothing happens.
15. Change "To page" to "1" and verify that the first dataset is "Angus-Hill ML et al., 2001".
16. Click on the "Publications" header and verify that the order changes from A-Z order to Z-A order. and that the first dataset shown is "Zhu G et al., 2000".
17. Change "To page" to "3" and verify that the first dataset is "Protchenko O et al., 2001".
18. Click on the "Publications" header, and verify that the "To page" is set to "1" and that the first dataset is "Angus-Hill ML et al., 200".
19. Click on the "Cond. Num." header, and verify that the datasets are ordered with with respect to condition numbers in ascending order, and that the first dataset has "3" condition variables.
20. Click on the "PMID" header, verify that the datasets are ordered with with respect to PubMed IDs in ascending order, and that the

first dataset is "DeRisi JL et al., 1997".

21. Change "To page" to "4" and verify that the first dataset is "Kuhn KM et al., 2001".
22. Click on the "PMID" header, and verify that the first page is shown and that datasets are ordered in descending order with with respect to PubMed ID and that "Eriksson PR et al., 2005" is the first dataset.
23. Click on the "details" link in the third dataset ("Lee A et al., 2005"), and verify that the dataset details page has PubMed ID "15989963".
24. Click on browsers back button to get back to the first dataset list page, and then click on the PubMed ID of the third dataset ("Lee A et al., 2005") and verify that the correct PubMed page is opened in a new tab.
25. Close the tab and click on the "Cond. Num" header, and verify that the datasets are ordered in descending order with with respect to condition numbers, and that "To page" is "1" and that the first dataset has 300 conditions.

Human Compendia:

1. Set datasets per page to "724" for scalability test.

Show Expression Levels and Expression Levels Pages

Micro compendium:

1. List the genes **CDP1 YDL005C HAL8 CRE2** with "#Datasets results" set to 10.
    1. Verify that 4 genes (CTR9 MED2 CRZ1 SPT6) and 4 datasets (Bernstein00, Brem05, Bulik03, and Caba05) are shown.
    2. In "additional display options" change mapping method for "single channel data" from "Per-gene log2 fold change" to "Log2 transcript count" and verify that datasets Caba05 and Bernstein00 change.
    3. Change mapping method for "dual channel data" from "Centered per-gene fold change" to "Reported log2 fold change" and verify that datasets Bulik03 and Brem05 change (the changes to Brem05 may be hard to see).
    4. Change color scheme for "dual channel data" from "Red/ Green" to "Yellow/ Blue" and verify that the color of datasets Bulik03 and Brem05 changes.
    5. Change color scheme for "dual channel data" back to "Red/ Green" and verify that the color of datasets Bulik03 and Brem05 changes.
    6. Click on the expression values for gene CTR9 in the Bulik03 dataset, and verify that the expression values are similar to "-0.26, -0.16, -0.11 -0.00, -0.02, 0.04, 0.05, 0.20, 0.29, 0.18, -0.16".

Yeast compendium:

1. List the genes **ctr9 med2** with "#Datasets results" set to 30.
    1. Verify that there are "30" datasets shown, that the number of datasets label is "of 112 datasets", and that the last dataset shown is "Duvel03".
    2. Increase number of datasets to 60, and verify that last dataset shown is "Hughes TR et al., 2000".
    3. Decrease datasets per page to "10", verify that ten datasets are shown and that the *two* last dataset shown are "Brauer MJ et al., 2005".
    4. Change "Datsets to show" to "0", verify that an error is displayed and that the datasets shown do not change, and that the to page remains "10".
    5. Change "Datsets to show" to "-1", verify that an error is displayed and that the datasets shown do not change, and that the to page remains "10".
    6. Change "Datsets to show" to "ads3", verify that an error is displayed and that the datasets shown do not change, and that the to page remains "10".
    7. Change "Datsets to show" to "5 6", verify that an error is displayed and that the datasets shown do not change, and that the to page remains "10".
    8. Click on the Bedalov01 dataset link, and verify that the dataset details for "Bedalov A, Gatbonton T, Irvine WP, Gottschling DE, Simon JA. Identification of a small molecule inhibitor of Sir2p.. Proceedings of the National Academy of Sciences of the United States of, 2001" are shown.
    9. Click on the "Expression Levels" for MED2 in the Bedalov dataset, then click on the "med2" gene link in the Single Expression detail view, and verify that the gene information for MED2/YDL005C is shown in a separate tab.
2. Repeat search test:
    1. Search for "CTR9", with "Datasets to show set to "50", and verify that 50 datasets are shown.
    2. Search for "CTR9", with "Datasets to show set to "112", and verify that 112 datasets are shown.
    3. Search for "CTR9", with "Datasets to show set to "10", and verify that 10 datasets are shown.
    4. Search for "CTR9 MED2", with "Datasets to show set to "112", and verify that 112 datasets are shown.

Human compendium:

1. Search for genes **BRCA1 BRCA2** with "#Datasets results" set to 724.

    1. Click on the "BRCA1" gene link and verify that the gene information for the gene is shown in a separate tab.

About Page

Yeast compendia:

1. Verify that the information for yeast is shown.

Human compendia:

1. Verify that the information for human is shown.

Layout

Verify the overall layout of each page, all tables, and all widgets by going through the following steps:

1. Open SpellWeb2 to get to the main search page for "Debug", then click on the "Human" link in the organism page.
2. Enter the query "brca1 brca2 aars" and click search to open the Search Results page.
3. Click on the help symbol after the "Search result" label to open the Help dialog (then close it after verifying the layout).
4. Enter "q" in the "Genes to show" text box to open the Error dialog.
5. Click on the "show additional display options" to open the hidden panel.
6. Close additional display options.
7. Click on a dataset link to open the Dataset Details page.
8. Click the browsers back button and then click on an expression level heatmap to open the Single Expression Details page.
9. Click on the "Private Search" link to open the Private Search page.
10. Open all hidden panels ("Required software", "Alternative to Java Web Start", "Sample datasets", and "Problems").
11. Close all hidden panels.
12. Click on the "Public Dataset Listing" link to open the Dataset List page.
13. Click on the "Expression Levels" link to open the Gene Expression Values link.
14. Enter the query "brca1 brca2 aars" to open the Expression Levels page.
15. Click the "About" link to open the About page.
16. Goto the address <server><port>/SpellWeb2.html#debug, and click on the buttons on that page.

System Testing

The systems tests that SpellWeb2 can display results received from DistributedSpell correctly.

Test Setup

The tests must be done manually.

Start DistributedSpell spellweb2-integration.config file, and SpellWeb2 using the spellweb2-integration-web.config configuration file. Then do the following tests:

1. Click on the yeast link in the organism panel.
2. Click on the dataset list link in the navigation panel and verify that there are 112 yeast datasets.
3. Click on the new search link, enter the query: **GOOD QUERY**, set gene to show to *50* and datasets to show to *40*, and click search.
    1. Verify that the search result is for genes:
    2. Verify that the number of genes to show is 50, number of datasets to show is 40, and that there are X weighted datasets.
    3. Verify that 50 genes and 40 datasets are shown.
    4. Verify that the 10 first genes are:
    5. Verify that the 10 first genes have ACS:
    6. Verify that the 10 first datasets are:
    7. Verify that the 10 first contributions are:
4. Select genes: , then click refined search.
    1. Verify that the number of genes to show is 50, number of datasets to show is 40, and that there are X weighted datasets.
    2. Verify that 50 genes and 40 datasets are shown.
    3. Verify that the 10 first genes are:
    4. Verify that the 10 first genes have ACS:
    5. Verify that the 10 first datasets are:
    6. Verify that the 10 first contributions are:

5. Click on the new search link, enter the query: **BAD QUERY**, and set genes to show to *25* and datasets to show to *25*.
    1. Verify that the number of genes to show is 25, number of datasets to show is 25, and that there are X weighted datasets.
    2. Verify that there is a warning about the datasets being weighted equally.
    3. Verify that 25 genes and 25 datasets are shown.
    4. Verify that the 10 first genes are:
    5. Verify that the 10 first genes have ACS:
    6. Verify that the 10 first datasets are:
    7. Verify that the 10 first contributions are:
6. Click on the new search link, enter the query: **GOOD QUERY**, and set genes to show to 50 and datasets to show to 50
    1. Verify that the search result is for genes:
    2. Verify that the number of genes to show is 50, number of datasets to show is 50, and that there are X weighted datasets.
    3. Verify that 50 genes and 50 datasets are shown.
    4. Verify that the 10 first genes are:
    5. Verify that the 10 first genes have ACS:
    6. Verify that the 10 first datasets are:
    7. Verify that the 10 first contributions are:
    8. Change the number of genes to show to 100, and the number of datasets to show to 100, and verify that 100 genes and 100 datasets are shown.
    9. Change the number of genes to show to 150, and verify that 150 genes and 100 datasets are shown.
    10. Verify that the genes with rank 100 through 105 are: , and that these have ACS's:
    11. Change the number of datasets to show to 125, and verify that 150 genes and 125 datasets are shown.
    12. Verify that the datasets with ranks 100 through 105 are:, and that these have conttributions:
7. Click on the new search link, enter the query: **GOOD QUERY**, and set genes to show to 150 and datasets to show to 150
    1. Verify that 150 genes and 150 datasets are shown.
8. FILTER TESTS
9. GO TERM TESTS
10. HUMAN TESTS
11. Repeat the tests described in the Integration tests section for the human compendium.

Regression Testing

Eclipse can be configured to run the Unit tests at compile time (invoke JUnit in the build process).

Unfortunately most of the integration and system tests must be done manually by following the step-by-step guides in the above sections.

Performance Testing

Stress Tests

Known Bugs and Issues

1. In the "Public Dataset List" page the table headers that are clickable do not have any visual hints that they can be clicked.
2. The Warning and Help dialog boxes could look nicer.
3. The help image could look nicer.
4. Update button is necessary since GWT TextBox change events are browser dependent. If changeHandler is used to update a table, entering "123" may update the table three times, once for each of "1", "12", and "123". Due to the same issue pressing enter in a text box may not disable the Update button in all views.
5. Should have a "wait while processing" message for high latency operations.
6. Color scheme may have to be changed.