The Ensembl Web Site: Mechanics of a Genome Browser

James Stalker,[1,3] Brian Gibbins,[2] Patrick Meidl,[1] James Smith,[1] William Spooner,[1] Hans-Rudolf Hotz,[1] and Antony V. Cox[1]

[1]*The Wellcome Trust Sanger Institute and* [2]*EMBL European Bioinformatics Institute, Wellcome Trust Genome Campus, Hinxton, CB10 1SD, UK*

The Ensembl Web site (http://www.ensembl.org/) is the principal user interface to the data of the Ensembl project, and currently serves >500,000 pages (~2.5 million hits) per week, providing access to >80 GB (gigabyte) of data to users in more than 80 countries. Built atop an open-source platform comprising Apache/mod_perl and the MySQL relational database management system, it is modular, extensible, and freely available. It is being actively reused and extended in several different projects, and has been downloaded and installed in companies and academic institutions worldwide. Here, we describe some of the technical features of the site, with particular reference to its dynamic configuration that enables it to handle disparate data from multiple species.

[Supplemental material available online at www.genome.org.]

It has become a truism that the body of data associated with genomes is large and ever increasing. It has also become evident that this plethora of data can make it difficult to access specific pieces of information, get a high-level overview of data, or examine data patterns. Ensembl is one of several systems that have been developed to help manage and display genomic sequence and annotation. It is characterized by the breadth of data it displays, by its extensibility and scalability, by its open Application Programming Interface (API), and by the availability of the entire system for download. (Glossary of technical terms available online as Supplemental material.) Users are encouraged to modify and experiment with the code and data.

Web-browser-based displays, although arguably not the best platform for a complex interactive application, have emerged as the dominant user interface (UI) to genomic data for a broad audience. The client-server nature of the Web not only provides a convenient UI, but also allows users to take advantage of large-scale compute facilities, for example, for searching or data-mining, that they might not otherwise have access to.

Individual sites have developed along different lines and present different data sets, but all use the same concept of graphical summaries, linked to detailed reports, as a way of navigating the sea of genomic data. Examples of large Web-based data browsers are the UCSC Genome Browser (http://genome.ucsc.edu/; Kent et al. 2002), a fast, configurable track-based browser, and the NCBI Map Viewer (http://www.ncbi.nlm.nih.gov/mapview; Wheeler et al. 2003), which is supported by the NCBI's extensive database resources such as LocusLink and dbSNP.

In addition to these multispecies sites, there are several highly detailed and specific sites for individual model organisms. Notable examples include the WormBase (http://www.wormbase.org/; Harris et al. 2003) and FlyBase (http://flybase.org/; FlyBase Consortium 2003) sites, which also provide extensive download facilities. The genome browsing components of these two sites, and others, are provided by GBrowse (http://www.gmod.org/; Stein et al. 2002), a modular software toolkit for building browsers.

The Ensembl Web site provides access to a wide variety of annotated metazoan genome information. At present, nine species are represented in Ensembl (human, mouse, rat, zebrafish, pufferfish, fruitfly, mosquito, and two nematode worms: *Caenorhabditis elegans* and *Caenorhabditis briggsae*). More than 20 different view pages present displays of distinct aspects of the data including assembled sequence, cross-species synteny, genes, transcripts, proteins, supporting evidence, dot-plots, protein domains, and gene/protein families. The site also provides similarity searching via BLAST (Altschul et al. 1990) and SSAHA (Ning et al. 2001), rapid retrieval of highly specific data sets with EnsMart (Kasprzyk et al. 2004), and the ability to display and share custom, user-specific data sets via the Distributed Annotation System (Dowell et al. 2001) or by using simple uploaded data files.

The Ensembl Web site is built on an open-source software platform comprising Apache/mod_perl and MySQL: a combination widely used to deploy open-source Web applications. Apache (http://httpd.apache.org/) is a popular Web server, and mod_perl (http://perl.apache.org/) embeds a Perl interpreter into Apache, providing, among other benefits, rapid execution of cached Perl scripts that would otherwise have to be compiled each time they were executed. MySQL (http://www.mysql.com/) is an open-source relational database that has proved to be fast, reliable, and that has scaled well to serve databases of 80 GB (gigabyte) or more, including individual tables >3 GB.

The Ensembl project offers an integrated, extensible, and reusable framework for generating, storing, retrieving, and displaying genomic annotation data. The Web site is only a part of that system, but relies heavily on the other components. The data displayed originate from the Ensembl pipeline (Potter et al. 2004) and are retrieved via the Ensembl Perl API (Stabenau et al. 2004). Although the Ensembl Web site is perhaps not as fast as one written in a compiled language, such as C, running against a fully denormalized database, it is an integrated part of the Ensembl system and thus fully exploits a set of powerful, reusable tools. Like Apache, Ensembl is not built purely for speed, but also for portability, extensibility, and flexibility. These properties are demonstrated in the way the Web code can be freely downloaded and easily installed and run locally. The code has been reused, both in its entirety on sites such as the Gramene rice genome (http://www.gramene.org/), the Vega curated annotation browser (http://vega.sanger.ac.uk/), and as a display toolkit for visualizing data ranging from SNP density distribution along chromosomes, to the results of gene expression experiments.

This paper does not explore the features or usage of the

[3]Corresponding author.
E-MAIL JWS@sanger.ac.uk; FAX 1223 494919.
Article and publication are at http://www.genome.org/cgi/doi/10.1101/gr.1863004.

Ensembl Web site; details of these can be found in the on-line documentation at http://www.ensembl.org/. Instead, we focus on some of the issues of Web code maintenance, and the approaches taken to address them.

RESULTS

One of the major challenges for the authors of the Web site is to keep it maintainable in the face of increasing quantities of data, and at the same time flexible enough to cope with data of new types. Ensembl has grown from its first release in late 1999, which had a relatively small amount of human sequence, through the addition of mouse in early 2001, to the current nine species, with more in preparation. The amount of data to be handled has increased dramatically—version 1.0 of Ensembl from April 2001 contained <8 GB, whereas the latest release (version 17) has grown >10-fold.

Whereas the volume of data is in itself an issue, particularly with regard to speed of display, additional complications arise from data differences among species: some sequences have assemblies, some are largely unassembled; some are mapped to chromosomes, some not; some have a full range of annotation; some are little more than the raw sequence. It is neither practical nor desirable to run a separate site for each species, or to maintain multiple versions of the same code. To scale to more than one species, the Web code must therefore be able to handle these differences between the data. For software to be truly flexible and reusable, it should not have hard-coded dependencies on particular aspects of the data, but generate dynamically an appropriate display depending on individual data sets. This is what the Ensembl site aims to achieve, and is why it has been possible for users to modify the site to display data from other species of interest without having to perform extensive re-engineering.

Site Internals

The Web site consists of a series of "view" scripts written in Perl (e.g., GeneView, ContigView, MapView) running in an Apache/mod_perl Web server environment. Mod_perl provides a persistent Perl interpreter embedded in each Apache process. This gives a significant increase in speed of execution of Perl scripts compared with CGI invocation because it is not necessary to call an external interpreter. In addition, the Perl code is compiled only once per Apache process and cached for subsequent requests. These benefits largely offset the difference in execution speed compared with a compiled language.

View scripts retrieve data from one or more MySQL databases, via the Ensembl API, and generate Web pages that are returned to a user's Web-browser. In this way, the majority of the site is generated dynamically from databases; only a relatively small set of static content exists, such as species home pages and documentation.

mod_perl Handlers

mod_perl offers other important benefits besides speed. It enables Apache to be configured dynamically, and provides access to the Apache internals, allowing default Web server behavior to be overridden by custom software modules. These features of mod_perl are used by the Ensembl Web site to enable a single Web software installation to support multiple data sets.

Apache processes Web-browser page requests through a cycle of distinct phases. Apache reads the incoming requests, translates the request into a file path, checks that the user is allowed to retrieve that resource, returns the content to the user, logs that it has done so, and then performs any necessary cleanup. At each stage, the Apache API allows an external module to take over handling of that stage from the internal Apache version. mod_perl provides Perl programmers access to this system, and allows "request handlers" to be written in Perl and installed dynamically as the server starts up and reads its configuration files.

The Ensembl Web site uses mod_perl handlers extensively to manage the delivery of dynamic content.

Ensembl Databases

For species data to be displayed by Ensembl, it must be represented, minimally, by a main Ensembl ("core"), and a denormalized ("lite") database, the latter being built from the core via several data transformation scripts, and contains denormalized data from some of the core objects (e.g., genes and transcripts) to speed up access. Depending on availability of data, there may also be additional databases associated with a species (e.g., SNP, EST, disease). The core database encapsulates the output of the Ensembl analysis pipeline and contains the assembly, DNA sequence, gene predictions, and their supporting evidence and similarity features mapped to the sequence.

Additional databases that extend Ensembl are accessed via database "adapters." Adapters are software modules that convert specialized database contents into software objects that can be manipulated by the wider Ensembl system. It is straightforward to extend Ensembl to include new types of data simply by writing a new database adapter.

The Web site's dynamic configuration enables it to handle the presence or absence of these additional data sources and modify displays appropriately. For example, if a "disease" database is added to a species configuration, Ensembl will automatically include disease information in GeneView displays.

In addition to these species-specific databases, there are several cross-species databases used by the Web site. These include the "compara" database of synteny/homology data, "family" database of protein family analyses, the Gene Ontology Consortium's GO database (http://www.geneontology.org/), and the EnsMart database. The latter is a query-optimized, denormalized data warehouse, constructed from Ensembl and third-party data sets.

How Ensembl Manages URL Requests

Ensembl's flexibility comes from its ability to handle multiple species using a single software installation. This flexibility is data-driven because species-specific code quickly becomes unmaintainable. How then, if code works for any species, is the current species to be identified in a multispecies site? One solution is to pass the species from page to page as a URL parameter or HTTP header. This can be awkward and requires significant effort to "maintain state." Another way might be to maintain user "sessions" with cookies; small text tickets containing configuration information, stored by a browser and accessible to the Web server. However, this could lead to problems on Web sites, such as Ensembl, where there are many server machines, or if two browser windows were opened looking at different species. The session-based approach can also make it difficult to bookmark a page. A more elegant solution, adopted by Ensembl, is to include the species in the URL. For example, a human GeneView page might have the address http://www.ensembl.org/Homo_sapiens/geneview?gene=BRCA2, whereas the same gene for mouse could be found at http://www.ensembl.org/Mus_musculus/geneview?gene=BRCA2. This seems natural to most users, who have become used to mentally dissecting URLs to orient themselves within a site. It is immediately clear from the URL which page and species is currently being browsed.

This solution makes it possible to mix static and dynamic content using the same URL naming conventions, and at the same time keep the files they refer to quite separate on the server.

For example /Homo_sapiens/geneview is a Perl script, kept in the /perl/default directory, whereas /Homo_sapiens/index.html is a static Web page located in the /htdocs/Homo_sapiens directory.

URL mapping is carried out by an Ensembl code module loaded into Apache (a mod_perl handler) that replaces its default URL translation mechanism. This occurs early in the request cycle, and maps URL requests to a file location.

Mapping a URL to a Species

The Ensembl code for the URL mapping phase first extracts the part of the URL immediately following the domain name, and assumes this to be the species name, for example, http://www.ensembl.org/**Homo_sapiens**/geneview. The species name (here "Homo_sapiens") is then looked up against a list of species configured for the Web site. If there is no match, it assumes the request is for a non-species-specific page (or an unknown species), and declines the request, passing it back to Apache, which processes it using its default behavior.

If a matching species is found, the handler attempts to find a file that matches the file part of the URL.

Mapping a URL to a File

The URL handler maps a URL to a file by checking for matches in several locations; first looking for scripts, then static content. Although the aim is to have one set of scripts that work for all species, to be able to make ad hoc species-specific changes, the system is designed so that the default script can be overridden by a species-specific version. The perl directory thus looks like:

/perl/default/	contains geneview, contigview, mapview, etc.
/perl/multi/	contains blastview, martview, helpview, etc.
/perl/Danio_rerio/	contains a zebrafish-specific version of geneview

Most scripts reside in /perl/default; to override a default script for a species, a different version may be placed in a directory of the species name. In the example above, there is a specific version of GeneView for *Danio rerio*. This is only intended to be for short-term fixes while differences among the data sets are resolved. In practice, this has proven to work well and be maintainable without sacrificing flexibility.

As scripts can be present in several directories, the order in which they are matched to the URL is important. First, the handler checks to see if there is a species-specific version of the script, and if so, maps the URL to that script and does no further checking. In the absence of a species-specific copy, it checks to see if this script should be handled by a multiple-species script in /perl/multi. This means that URLs like http://www.ensembl.org/Homo_sapiens/blastview and http://www.ensembl.org/Mus_musculus/blastview will both map to /perl/multi/blastview, enabling species-specific URL entry points to reference a cross-species script. After multispecies, the /perl/default directory is checked, and, finally, if no scripts can be found to match the requested URL the request is passed back to Apache's default handling system, which continues the search, but for a static Web file instead of a perl script. If the request is for a species-specific static Web page, for example, http://www.ensembl.org/Homo_sapiens/index.html, then Apache will look for that page under the static document directory tree. If, at the end of all these checks, no matching file at all can be located, then a "Page Not Found" error is returned.

Before the handler responsible for the file-mapping phase completes and returns control to Apache, it sets a process flag containing the name of the current species. This is then available for use by all downstream code to allow appropriate data and configuration settings for that species to be used by subsequently executed code.

Wrapping Static Pages

Even "static" content on the Ensembl site is partially dynamic. An Ensembl code module is installed in Apache to handle all requests for all files with names ending in ".html," and wraps the raw HTML retrieved from disk with Ensembl headers and footers before returning the completed page to the client browser.

Limiting Process Size

One trade-off for speed gains resulting from using an embedded Perl interpreter is that it can cause individual Apache processes to use much more server memory than they otherwise might. This is because of factors such as cached code and data being held in the process and not released until the server process exits. The Apache Web server operates by effectively copying itself and using these "child" copy processes to service incoming requests. Configuration options define how many requests each child will serve before it is shut down (and its memory pool released). Although it is advantageous for performance reasons to make use of caching and allow children to service many requests, the longer a child persists, the more likely it is to use large amounts of memory. Ensembl adopts the pragmatic approach of generally letting children service many requests but selectively removing those that exceed a threshold memory limit, using a code module called Apache::SizeLimit (http://search.cpan.org/author/GOZER/mod_perl-1.28/lib/Apache/SizeLimit.pm). This cleanup phase occurs after the response has been sent to the client; thus such process elimination is invisible to Web site users.

Errors

Careful handling of internal (code) and external (URL) errors is important. Ensembl installs code to handle HTTP 500 (Server Error) and 404 (Page Not Found) errors. Apache redirects these errors to special URL locations (/Crash and /Missing), and additional code handlers for those specific locations return appropriate error pages to the user, and optionally send E-mail to the site maintainers to warn of the problem.

Storing User Preferences

The Ensembl Web site maintains user preferences (such as which tracks are displayed in ContigView) through the use of cookies. These provide an anonymous key into a simple database that stores the settings. A code handler registered to the start of the Apache request cycle is used to initialize this cookie system for each request.

Configuration

Simplicity was the principal consideration in designing the Web site configuration system so that external users could easily install it. The configuration defined in several text files in a single directory. In general, it is necessary to specify only a few settings to get a copy of the site running. Settings are split into configuration for the whole site, stored in a Perl module (named "SiteDefs.pm"), and species-specific settings stored in ⟨species_name⟩.ini files (e.g., Homo_sapiens.ini).

SiteDefs.pm contains the global configuration information required to start the site up: the file system location of the Web site code, the port on which to run the server, process user and group security, the location of temporary files, and so on. It also holds some general settings, such as the species enabled on a given site. As SiteDefs.pm is a Perl module, these settings are stored as simple Perl variables, for example:

```
$ENSEMBL_SERVERROOT = '/my/ensembl/server';
$ENSEMBL_PORT       = 80;
$ENSEMBL_USER       = 'nobody';
$ENSEMBL_GROUP      = 'nobody';
```

SiteDefs.pm is used to configure Apache on startup. This is achieved with "Perl sections" embedded in the Apache configuration file (httpd.conf). Upon startup, the code in the Perl sections is compiled, and used to configure Apache. The Ensembl configuration contains Perl sections to locate and use SiteDefs.pm, and thereby import all the settings necessary to start Apache. These settings are inherited by all Apache child processes and remove the need to rely on further configuration information in each script.

Species-specific initialization (or ".ini") files describe data available and some general settings for each species. This includes which databases are present, and how to connect to them, whether the species is fully assembled or not, which chromosomes the assembly contains, whether BLAST/SSAHA searches are available, and so on. General settings include default text to use in "view" pages, the abbreviated name of the species, external linking URLs, and which distributed annotation sources to use. Initialization files are simple stanza-based key/value pair text files, divided by group headings, for example:

```
[general]
ENSEMBL_HOST        = localhost
ENSEMBL_HOST_PORT = 3306
ENSEMBL_DBUSER      = mysqluser

[databases]
ENSEMBL_DB          = homo_sapiens_core_16_33
ENSEMBL_LITE        = homo_sapiens_lite_16_33
ENSEMBL_DISEASE     = homo_sapiens_disease_16_33
```

Most species-specific configuration details are, in practice, common to more than one species; for example, MySQL database connection details, the color scheme to use, and lists of external URLs. To remove the requirement for redundant copies of these data, there is a default initialization file, called "DEFAULTS.ini," which contains these common settings. A species-specific initialization file can override general values from DEFAULTS.ini, but otherwise the species will all share these defaults. Multispecies databases need only be configured once, thus settings for these are stored in a separate "MULTI.ini" file, which has the same format as the species-specific files.

During server startup, a Perl module called SpeciesDefs parses the species initialization files and builds a complex data structure containing all configuration settings. SpeciesDefs then saves the data structure to disk, and provides methods to access it and retrieve values. The Web code uses the SpeciesDefs module, and the methods it provides, to access rapidly the cached configuration settings.

In addition to storing the explicit settings from the initialization files, SpeciesDefs determines several implicit values from the configured databases, and also stores these in the data structure. These settings include the sizes of various database tables, the type of analysis features present in the database, and the length of the longest assembly unit (usually the longest chromosome). These settings are used by the Web code to enable/disable certain displays, are generally slow to generate dynamically and do not change for a given set of databases. Precalculating and caching the values in the configuration data structure is an efficient approach.

DISCUSSION

All public genome browsers, including the prominent UCSC Genome Browser, NCBI Map Viewer, and the GBrowse system, have their particular strengths. The UCSC browser exemplifies speed: written in a high-level compiled language, and with a tuned, denormalized database backend, it is a responsive and powerful application for rapidly displaying a wide range of track-based data. NCBI's Map Viewer is integrated into a larger site, and is linked to the impressive range of databases that the NCBI curates. This enables users to investigate features in depth without leaving the site. GBrowse is a sophisticated toolkit designed to simplify building data browsers to display custom data. As part of the Generic Model Organism Database Project, it has particular strengths in handling model organism data, as typified in the WormBase and FlyBase sites.

The strength of the Ensembl browser lies in its flexibility, and its integration into the larger Ensembl system. These factors help it provide probably the broadest range of data displays of all the public browsers. They have led to the site being downloaded and used in academic institutions and pharmaceutical and biotechnology companies worldwide, providing rapid access for local users, display of private data, and integration with local systems. The fact that it is written in Perl, rather than a compiled language, has probably contributed much to its reusability and to its extension by external users. Rather than start from scratch, users may copy, modify, and extend existing code as required using the rich Ensembl API. In the future we intend to make this easier by restructuring the Web display code into separate modules for data access, HTML generation, and display. This will produce a higher-level API to the Web code, encouraging reuse and simplifying the creation of custom pages, Web services, and the like.

Being part of the larger Ensembl system greatly benefits the site. For example, automated gene predictions from the Ensembl pipeline are an extremely valuable and useful resource for the site to display. The complexity of the database may make it slower than if it were simpler and denormalized, but it is this complexity that gives the site a rich feature set and comprehensive annotation, not merely the start and end locations of chromosome features. Similarly, access to data via the Ensembl APIs may not be as fast as querying the database directly, but the API provides an extremely powerful toolkit for working with the biological objects represented by the data.

Despite these positive qualities, we always seek to improve. We feel that seeing the strengths of other browsers is an important spur to keep getting better, and friendly competition among the public genome browsers has helped us all toward achieving our aim of turning a deluge of data into useful information.

ACKNOWLEDGMENTS

The Ensembl group is funded primarily by the Wellcome Trust with additional funding from EMBL and NIH-NIAID. Special thanks to the other members of the Ensembl group for their assistance and support; to Roger Pettett for, in particular, the original ensembl-draw code; and to the rest of the Sanger Web team. We also thank the subscribers to the ensembl-dev mailing list (ensembl-dev@ebi.ac.uk) for their many useful suggestions and comments.

The publication costs of this article were defrayed in part by payment of page charges. This article must therefore be hereby marked "advertisement" in accordance with 18 USC section 1734 solely to indicate this fact.

REFERENCES

Altschul, S.F., Gish, W., Miller, W., Myers, E.W., and Lipman, D.J. 1990. Basic local alignment search tool. *J. Mol. Biol.* **215:** 403–410.

Dowell, R.D., Jokerst, R.M., Day, A., Eddy, S.R., and Stein, L. 2001. The distributed annotation system. *BMC Bioinformatics* **2:** 7.

The FlyBase Consortium. 2003. The FlyBase database of the *Drosophila* genome projects and community literature. *Nucleic Acids Res.* **31:** 172–175.

Harris, T.W., Lee, R., Schwarz, E., Bradnam, K., Lawson, D., Chen, W., Blasier, D., Kenny, E., Cunningham, F., Kishore, R., et al. 2003. WormBase: A cross-species database for comparative genomics.

Nucleic Acids Res. **31:** 133–137.
Kasprzyk, A., Keefe, D., Smedley, D., London, D., Spooner, W., Melsopp, C., Hammond, M., Rocca-Serra, P., Cox, T., and Birney, E. 2004. EnsMart—A generic system for fast and flexible access to biological data. *Genome Res*. **14:** 160–169.
Kent, J.W., Sugnet, C.W., Furey, T.S., Roskin, M.K., Pringle, T.H., Zahler, A.M., and Haussler, D. 2002. The Human Genome Browser at UCSC. *Genome Res*. **12:** 996–1006.
Ning, Z., Cox, A.J., and Mullikin, J.C. 2001. SSAHA: A fast search method for large DNA databases. *Genome Res*. **11:** 1725–1729.
Potter, S.C., Clarke, L., Curwen, V., Keenan, S., Mongin, E., Searle, S.M.J., Stabenau, A., Storey, R. and Clamp, M. 2004. The Ensembl analysis pipeline. *Genome Res*. (this issue).
Stabenau, A., McVicker, G., Melsopp, C., Proctor, G., Clamp, M., and Birney, E. 2004. The Ensembl core software libraries. *Genome Res*. (this issue).
Stein, L.D., Mungall, C., Shu, S., Caudy, M., Mangone, M., Day, A., Nickerson, E., Stajich, J.E., Harris, T.W., Arva, A., et al. 2002. The Generic Genome Browser: A building block for a model organism system database. *Genome Res*. **12:** 1599–1610.
Wheeler, D.L., Church, D.M., Federhen, S., Lash, A.E., Madden, T.L., Pontius, J.U., Schuler, G.D., Schriml, L.M., Sequeira, E., Tatusova, T.A., et al. 2003. Database resources of the National Center for Biotechnology. *Nucleic Acids Res*. **31:** 28–33.

WEB SITE REFERENCES

http://blast.wustl.edu/; WU-BLAST home page.
http://flybase.org/; FlyBase home page.
http://genome.ucsc.edu/cgi-bin/hgGateway; The UCSC Genome Browser.
http://httpd.apache.org/; The Apache HTTP Server home page.
http://perl.apache.org/; The Apache mod_perl home page.
http://search.cpan.org/author/GOZER/mod_perl-1.28/lib/Apache/SizeLimit.pm; Apache::SizeLimit module description.
http://vega.sanger.ac.uk/; Vega curated annotation browser.
http://www.ensembl.org/; The Ensembl Genome Browser.
http://www.geneontology.org/; Gene Ontology Consortium home page.
http://www.gmod.org/ggb/index.shtml; GBrowse home page.
http://www.gramene.org/; Gramene genome browser.
http://www.mysql.com/; MySQL home page.
http://www.ncbi.nlm.nih.gov/mapview/; NCBI Map Viewer.
http://www.wormbase.org/; WormBase home page.

Received August 8, 2003; accepted in revised form November 25, 2003.