# Chapter I

# Introduction to Bioinformatics and Java

## The Origins of Bioinformatics

On April 15, 2003, the International Human Genome Sequencing Consortium (IHGSC) – the association of laboratories from around the world which had jointly undertaken the *Human Genome Project* (HGP) formally announced the completion of the project and the colossal task that lay at its core: the sequencing and assembly of the more than 3 billion bases that comprise the *Homo sapiens* (human) genome. This is a truly landmark achievement for science and medicine. According to Nobel Laureate James D. Watson, President of the Cold Spring Harbor Laboratory, "*The completion of the Human Genome Project is a truly momentous occasion for every human being around the globe.*" In the words of Elbert Branscom, Founding Director of the Joint Genome Institute (JGI), "*We will see everything before this like the dark ages of biology*".

The HGP has had wide ranging implications on every aspect of science and medicine. As a result of the HGP, scientists have mapped the DNA hieroglyphic of the human genome to an accuracy of 99.99 percent and have estimated that human life and all its molecular, cellular and organismal machinery is programmed by 30,000 odd individual genes. It has given birth to *Bioinformatics* - a new scientific discipline at the crossroads of biology, medicine and information technology and provided an impetus for the rapid development of the fields of *Genomics* (the study of the genome) and *Proteomics* (the study of the entire complement of

proteins expressed by the genome). Along with the sequencing of the human genome, the sequencing of model plant and animal genomes such as *Arabidopsis thaliana* (thale cress), *Caenorhabditis elegans* (worm), *Danio rerio* (zebrafish) and *Drosophila melanogaster* (fruit fly) has led to the development of fundamentally new discovery approaches and technologies that promise to revolutionize medicine.

In the space of just a few years, we have taken a giant step closer to a paradigm shift from "just-in-time" medicine (where treatment is provided after the appearance of symptoms) to "predictive medicine" (where the entire spectrum of disease susceptibility of an individual can be mapped at birth and treated in advance of the appearance of disease). We are also moving closer to an entirely new concept in therapy - "personalized medicine" (as opposed to "generalized medicine"), where individuals receive treatment with "designer" drugs that are tailored to suit their specific genetic backgrounds, thereby maximizing therapeutic potential and minimizing the occurrence of adverse events.

Why does one person respond to a certain medication while another does not? Why do some women get breast cancer while others do not? Why are some individuals more susceptible to an infectious disease than others? These are the kind of questions that biologists are trying to address. The next few decades will be completely consumed in research that leads to answers to these issues. The need to analyze vast amounts of genetic data has lead to the growth of powerful technologies that enable researchers to study the regulation of tens of thousands of genes at the same time. To be able to perform these information intensive tasks, scientists and clinicians must be comfortable with both the biological and the computational aspects of Bioinformatics as well as with the basic tasks of retrieving, extracting, organizing, analyzing and representing the data. While *Perl* and other scripting languages are preferred for day-to-day analysis of biological data, they are not suited for creating enterprise-level software. A robust *Object-Oriented Analysis, Design and Programming* language such as *Java* is better suited for this purpose. The *Java 2 Enterprise Edition* (*J2EE*) framework provides the ability to develop distributed, multi-tier applications that can be deployed and connected over the web. J2EE is platform-agnostic, meaning that it can run on virtually any platform. This is because the Java code is compiled into an intermediate code called *byte code*, which is interpreted and executed by the *Java Runtime Environment* (*JRE*) at run-time. Since *JRE* is available

on any platform, code once created in Java can be run on any operating system.

In this Chapter, we will explore some bioinformatics applications that have been written in Java in order to demonstrate the power of J2EE technologies for creating biomedical software. In particular, we will focus on applications that have been developed for cancer research that have achieved the "industry standard" reputation in modern research and are actively being integrated for use in such cutting-edge research initiatives as the National Cancer Institute's *cancer Biomedical Informatics Grid* program (caBIG™, http://cabig.nci.nih.gov/). In doing so, we will provide an introduction to caBIG™ in this chapter and discuss how the different tools and applications that are being built or are being brought under the caBIG™ umbrella are helping solve the many bottlenecks in biomedical research.

## Current State of Biomedical Research

Traditionally, biomedical research has been (and is still being) conducted in laboratories around the world in relative isolation from other laboratories, even if the subject of research may have been (or is) the same. While this method of operation has over the decades led to a rich collection of research data and many significant biomedical discoveries, it has also led to the isolation of data and capabilities into independent silos of information and expertise that lie locked in databases or within people and inaccessible to the larger research community. In addition, since the majority of individual laboratories have evolved their own operating procedures, methodologies and vocabularies to suit their own specific research problems, there has been a relative dearth of standardized ways of conducting and reporting experimental data. The lack of standardization and data sharing has proven to be a significant impediment to biomedical research and directly affects our ability to design better and more effective treatments.

Experts all over the world now generally agree that a better use of research data, especially with the aim of enhancing the pace of biomedical research for the benefit of the patient, is through open collaboration and sharing. This approach eliminates duplication of effort and result in a more efficient use of limited resources. This realization is especially significant in the post-genomic era. Modern day *high-throughput* assay technologies

have given researchers the power to probe living systems with unprecedented precision and depth. This has in turn led to the adoption of a "systems" approach to research with an increasing trend towards studying entire pathways, hundreds and thousands of genes and whole organisms in one single experiment. However, this approach has also led to an explosion of raw data. There is today an ever-increasing need to connect this raw data into meaningful actionable knowledge that can yield real insights into disease processes.

Another significant change is the realization that a more powerful way of conducting research is to integrate data from multiple different fields of study spanning basic (laboratory-based) and clinical (patient-focused) research. This new approach called "*Translational research*" requires a team approach between physicians, scientists, bioinformaticians, statisticians and a host of other professionals working closely together towards specific outcomes. This method of operation brings together the cellular, molecular, biochemical, genetic and other biological aspects of research together with a clinical understanding of disease that results in practical outcomes of valuable clinical relevance. For example, translational research on lung cancer may involve a team consisting of *molecular biologists*, *computational biologists* and *biochemists* on one hand and, *thoracic surgeons*, *medical oncologists*, *radiation oncologists* and *nurse practitioners* on the other to understand basic disease mechanisms and to improve patient outcomes.

The basic idea behind this approach is to assimilate as much corroborating evidence as possible to test and validate a hypothesis rather than dealing with separate isolated bits and pieces of raw data, which do not point to a robust testable hypothesis. With the appropriate standards, processes, policies and technologies in place, a researcher following a promising lead, for example, a gene or a protein that is significantly overexpressed in a specific cell population or in a laboratory model and is suspected to play an important role in disease causation, can extend the research in meaningful ways by:

1. performing experiments that prove that inhibiting protein overexpression or inhibiting a specific step in a biochemical pathway reverses the ill-effects of the abnormal protein expression or the aberrant pathway

2. confirming that the results can be duplicated in biospecimens - that is, samples derived from tissues obtained from specific human organs (for example, lungs) possessing the same disease pathology and characteristics, thereby extending the evidence in actual patient samples

3. confirming that the protein is not present in normal non-target tissues (for example, liver, kidney, etc.) to avoid occurrence of toxicity due to a chemical agent being tested for interventional therapy

4. identifying patient cohorts who fit the study criteria and conducting therapeutic clinical trials to test efficacy of known or experimental agents for interventional therapy

The over or under expression of a *biomolecule* (typically a gene or a protein) - that is, its presence in higher or lower amounts, respectively, under a diseased condition (as compared to the levels that are observed under normal conditions) is generally referred to as *differential expression.* The differentially expressed protein in question can serve as a signature or a fingerprint of the underlying disease mechanism and is the living system's response to an alteration in normal physiology caused by disease or other external stimuli. Since it is a signal or a "marker" with significant biological implications, it is called a *biomarker.* Biomarkers can be any biomolecule - proteins, peptides, nucleic acids, carbohydrates, lipids, metabolites, etc. - the concentrations of which may increase or decrease, under specific abnormal conditions. An example of a biomarker is cholesterol, which is commonly used to identify risk of heart disease. Biomarkers can be assayed by standard biochemical methods and can be used as indicators of disease states in diagnostics as well as provide targets for therapeutic intervention. The application of biomarkers to diagnostics includes the ability to diagnose and monitor disease, risk stratification, disease prognosis, drug eligibility, prediction of safety and efficacy, and therapeutic monitoring. The therapeutic aspect is equally important because they provide a reliable readout of drug function and treatment efficacy and therefore guide decisions on the clinical development of promising drug candidates.

The research can be further extended by identifying patient cohorts who fit the study criteria in clinical trials to test efficacy of known or experimental agents that inhibit overexpression or otherwise reverses the ill-effects of the causative protein. Of course, this is a rather simplistic

representation of an actual research scenario. The researcher may spend months or even years studying disease causation in the laboratory eliminating other suspected causative agents, sifting through literature and accumulating data from studies performed by other scientists, mining the available data using statistical and analytic algorithms, and iterating through each of these steps till a model that fits the observed data can be created with a high-level of confidence. In reaching this goal, the researcher has to have access to the appropriate tools to identify relevant research, assure that the data can be compared across experiments done under different conditions or if not, apply the necessary manipulations using appropriate tools, have access to those tools, and have the necessary resources to identify tissues, experimental models or human subjects locally or at other institutions. Such "bench to bedside" research can be conducted only in a situation where data, resources, applications and people are connected with one another and accessible via standardized ways on a network or grid infrastructure. This is the rational and promise of NCI's caBIG™ program.

## The cancer Biomedical Informatics Grid program

caBIG™ was started by the NCI in July 2003 as a pilot project to create a standards-based interoperable network of cancer centers across the nation to increase data sharing and cooperation between biomedical scientists and to enhance the pace of cancer research. The aim of caBIG™ is to integrate bioinformatics, *cancer informatics*, *tissue informatics*, and *pathology informatics* to create a network of data, applications and individuals who can share data and tools seamlessly across geographical boundaries. To cover the various aspects of the complex cancer research domain, caBIG™ is divided into four Workspaces - Clinical Trial Management Systems (CTMS), Integrative Cancer Research (ICR), In Vivo Imaging and the Tissue Banks and Pathology Tools (TBPT) Workspace. Simply stated, caBIG™ is putting the "e" in cancer research, leading to an "e-research" platform that integrates data and knowledge from basic (laboratory-based) research to clinical (patient-based) research. To draw an analogy with the term e-business that refers to the application of Internet technologies to streamline enterprise business processes, caBIG™ is aimed at building the infrastructure, processes and policies to make research data from multiple research centers available via the web, handle secure transactions across networks, support queries and secure information interchange between distributed institutions, and enhance the efficiency of the cancer research

engine as a whole. Making cancer data available electronically over the Internet enhances the speed of access to information, offers the opportunity to globalize data access and interchange, enables access to the most up-to-date data, enables researchers to adapt and quickly incorporate the latest understanding of disease biology into their experimental designs, and ultimately, to respond faster to critical patient needs and provide high quality service.

While there are some parallels between biomedical research data and business data, the two differ fundamentally in many respects, especially with respect to data on patient related medical information. caBIG™ therefore has to create this e-research infrastructure in strict compliance with applicable federal regulations for the protection of what is known as *individually identifiable* health information that can be linked to personal medical data and, if exposed, provoke the risk of misuse. In particular, the privacy provisions of the *Health Insurance Portability and Accountability Act of 1996* (HIPAA), apply to and seek to protect patient health information that is created or maintained by health care providers who engage in certain electronic transactions, health plans, and health care clearinghouses. A detailed treatment of the HIPAA rule is beyond the scope of this book. Suffice to say that this federal law gives patients rights over any personal medical data that health professionals and care providers collect in medical records and sets rules and limitations around who can receive and view their personal health information.

## caBIG™ Organization and Architecture

As of this writing, caBIG™ had grown to a large enterprise consisting of more than 70 individual projects, more than 800 individual participants spanning greater than 70 public and private organizations. The caBIG™ enterprise has to support a complex interplay of customers (patients, research investigators, clinicians, bioinformaticians, etc.), and federated data (both text and image), services and analytic tools (data extraction, organization, querying, mining, clustering and visualization tools) over the web, while ensuring that it meets the necessary performance and capacity requirements for such operations. By its very design, caBIG™ systems need to be compatible with other systems on the network and make data and services available irrespective of the type of web-based system or device accessing caBIG™ resources. The caBIG™ infrastructure has to provide fail-safe mechanisms to serve its resources in a continuous manner

without downtime for optimal benefit for the research community. The need to access and distribute sensitive clinical, pharmacogenetic and financial billing data under caBIG™ means that appropriate technologies and policies must be implemented to assure privacy, confidentiality and integrity of data, while blocking unauthorized access. These are just a few issues that make the caBIG™ initiative such a complex undertaking. The NCI Center for Bioinformatics (NCICB) has a key role in the making of caBIG™ and is actively developing the critical infrastructure components needed to address these requirements. Information on a sampling of such tools, for example, the *Common Ontologic Representation Environment* (caCORE) Software Development Kit (caCORE SDK), the *Common Security Module* (CSM), *caAdapter* and others, can viewed at the NCICB website at the following URL (http://ncicb.nci.nih.gov/NCICB/infrastructure).

How does one design a secure and scalable solution for an enterprise this large that covers all the pieces – the biomedical and clinical organization, the computing infrastructure, including applications, systems, servers, storage and the network - of a complex and distributed modern research and healthcare environment? How can the various building blocks or business components be assembled to deliver the services and capabilities required to address the lifecycle needs of the federated biomedical enterprise? The presence of data, services and tools in a distributed manner and the requirement of data sharing between organizations via the web means that we can no longer develop monolithic applications with user interfaces that simply talk to a backend database. Instead, the architecture has to accommodate a new design consisting of several "layers" or "tiers" that may be present on separate physical machines, operate independently of one another and subserve specific functions. In effect, any number of such layers may be present and because of the functional separation that the layer architecture provides, each layer preserves its distinct identity and can be maintained without regard for the implementation details of other layers. In effect, this design affords the developer with immense convenience for use and maintainability because entire tier implementations can be modified without affecting the rest of the application. The users can in turn access the required resources in a seamless and transparent manner. Such an architecture is called an *n-tier architecture*. The n-tier architecture consists of several tiers that perform the following functions - the display or presentation of data, the conduct of business logic and the storage of data. These are commonly referred to as the *Presentation tier*, the *Business tier*, and the *Data* or *Persistence tier*,

respectively. **Fig. 1.1** below provides a graphical representation of this model.

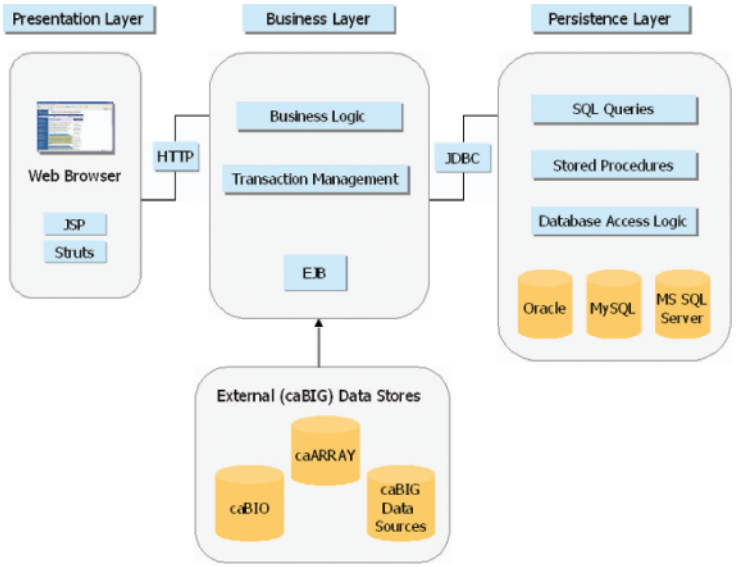


**Fig. 1.1.** Components of an n-tier architecture

## The Model-View-Controller Framework

A concept that is closely associated with the n-tier architecture is a design principle called the *Model-View-Controller* (MVC) framework. The MVC framework defines separation between the data (Model), the visual component (View) and the communication that occurs between them (Controller). There are a number of advantages of using such a design.

The separation of components allows developers to prototype an application and validate its requirements quickly. The *view*, for example, can be designed and developed independently without affecting the design of the rest of the application. It's likely that the *View* will be modified more often than the *Model* (the data) to adapt to the requirements of users navigating through the user interface (UI). In addition, the way the *Model* is implemented is fully encapsulated and transparent to the other parts of the application.

The *Controller* handles the input that the *View* receives; it can then take action to update the *Model*. The *Controller* can also inform the *View* to update itself or the *View* can register itself as a *listener* of a *Model*, in which case the *View* will update anytime the *Model* notifies its *listeners*. This is the definition of the *observer pattern* where a *View* is the *Observer* and the *Model* is the *observable*. The most important thing in *MVC* is to keep the separation between the *Model* and the *View*. We will use this as a guiding principle as we build our applications in subsequent chapters.

## Web Services and Service-Oriented Architecture

The biomedical enterprise needs to transform itself from an unorganized collection of data, tools and services into an interoperable, integrated and standards-based model that allows the system and its users to interact with a variety of business elements and invoke a variety of services along logical workflows. Under this scheme, any machine located on the web can be thought of as a provider of a consistent, reliable and defined "service" that can be invoked in a repeatable and standard manner. The *Basic Local Alignment Search Tool* (BLAST) server provided by the National Center for Biotechnology Information (NCBI), for example, provides a distinct service to a user – the ability to perform homology searches with a given nucleotide of amino acid sequence. The *Genscan* web server at MIT provides a different kind of service called "*gene prediction*" or the identification of complete gene structures in genomic DNA sequences. One can imagine the *World Wide Web* as made up of a large number of such services that can be accessed via standard Internet protocols such as *HTTP*, *FTP* etc. Each of these separate bits of functionality is a service and in each case, a service consumer (user or client) communicates and requests services from a service provider; the service provider in return communicates back the service requested. Both transactions (request and response) are carried out using messages that both parties can understand. Messaging between the services can be performed using the *eXtensible Markup Language (XML)*. This is the concept behind the emerging web architecture called *service-oriented architecture (SOA)*. The individuals services are connected together using *Web Services*, which define a set of technologies that enable connections between services.

The individual (web) services are self-contained, self-describing, modular applications that can be published, located and invoked across the

Web as well as discovered by other applications on the web. Each of these characteristics of a web service defines an essential component of the web services platform:

1. The means to communicate (pass messages and data) between services. This is usually achieved using *Simple Object Access Protocol* (*SOAP*), which defines a uniform way of passing XML-encoded data and a way to perform *remote procedure calls* (*RPCs*) using the Hypertext Transfer Protocol (HTTP) .

2. The ability to dynamically locate other services present on the web using a directory service. This is called *Universal Description, Discovery and Integration Service* (*UDDI*).

3. The ability to describe *what* a web service can do, *where* it resides, and *how* to invoke it. This is achieved through the *Web Services Definition Language* (*WSDL*).

As is apparent from the above, web services must use interfaces based on common Internet protocols such as HTTP and must use the XML standard for messaging. Although a detailed description of the web services platform and *SOA* is beyond the scope of this text, we will illustrate how the caBIG™ grid architecture called caGrid addresses the complex interoperability and integration issue we described earlier. We will delve into caBIG™ and the technologies being developed under the project in more detail in Chapter 6.

## CaGrid

As mentioned briefly before, to make data interchange and collaboration possible, NCI and the caBIG™ participating institutions are using a number of technologies that the NCICB has been developing for the last several years. These include, for example, caCORE, Cancer Bioinformatics Infrastructure Objects (caBIO) and the Cancer Data Standards Repository (caDSR). These technologies allow integration of biomedical applications with a vast array of NCI data sources including genomic, animal model and clinical data. The NCI has also formulated compatibility guidelines to ensure that applications developed under the caBIG™ umbrella can interoperate with one another. The caBIG compatibility guidelines necessitate the use of controlled vocabularies and

terminologies, Common Data Elements (CDEs), well documented API and *Unified Modeling Language* (*UML*) based object models to ensure interoperability with other caBIG applications. caCORE, which is caBIG's principle software development platform allows users to create caBIG™-compatible systems using an in-built modeling tool and a code generator.

The caBIG™ grid framework or *caGrid* is based on the service-oriented architecture model and open standards such as *Open Grid Services Architecture* (OGSA) created by the *Global GridForum* (GGF) for grid computing. The current version of caGrid as of this writing (caGrid 0.5) is built using the *Globus Toolkit 3.2* and the *OGSA Data Access Integration* (OGSA-DAI) framework version 5.0. The Globus Toolkit provides services and applications for the secure sharing and management of computing power, databases, and analytic tools over the web across organizational and geographic boundaries. OGSA-DAI component provides the middleware needed for accessing and integrating data via web services from the multitude of geographically distributed biomedical data sources on the grid including relational databases and XML based databases. Through the combination of these various components, caGrid empowers the caBIG™ engine and its users to develop and deploy of community provided services and API for building client applications.

Now that we have the basic background on caBIG™ and bioinformatics, lets examine a few software applications that are currently being used or are being developed under the caBIG™ program for oncology research to illustrate what scientists, clinicians, bioinformaticians and software engineers have together accomplished to address the needs in this area. We will use the research scenario we had discussed earlier - the differential expression of a gene and its product in a specific cell population or, in a disease model that leads to the plausible hypothesis that it has a role in disease causation - to provide examples of biomedical software applications. **Table 1.1** provides a breakdown of the translational research scenario into discrete sub-components and lists out the corresponding categories that apply to the scenario.

**Table 1.1.** Research use cases and corresponding categories

| Research scenario | Category |
|---|---|

| | |
|---|---|
| Analyze genes that are differentially expressed in a specific cell population or a disease model | Gene expression analysis |
| Analyze proteins that are differentially expressed in a specific cell population or in a disease model | Proteomics |
| Analyze pathways that the differentially expressed molecules participate in | Pathway analysis |
| Query for and identify tissue samples located in distributed biospecimen resources that match the clinical, pathologic, and experimental parameters of the disease under investigation | Biospecimen inventory and annotation systems |

**Table 1.2** provides brief descriptions of the tools that we will introduce in this chapter to illustrate a representative set of Java-based bioinformatics applications. Also listed are the caBIG™ Workspaces under which each of the tools are being developed.

**Table 1.2.** Java-based bioinformatics tools

| Name of application | CaBIG™ Workspace | Description |
|---|---|---|
| CaArray | ICR | Repository for managing, analyzing and visualizing of gene expression data from microarray experiments |
| CaWorkBench | ICR | Gene expression, pathway and sequence analysis, transcription factor binding site analysis, and pattern discovery |
| RProteomics | ICR | Statistical analysis, visualization m modeling of proteomics spectra |
| cPath | ICR | Integration and analysis system for integrating protein-protein interaction and molecular pathway information from multiple sources |
| caTissue Core | TBPT | Core biospecimen management tool for inventory, tracking and basic annotation of biospecimens. |
| CaTissue Clinical | TBPT | Tool for addition of pathology annotation to |

| | | |
|---|---|---|
| Annotation Engine (CAE) | | stored biospecimens using data from Anatomy Pathology systems, Clinical Pathology systems and tumor registries. |
| cancer Text Information Extraction System (CaTIES) | TBPT | Tool for extraction of pathology data such as tumor histology, staging, molecular markers, etc., from free text surgical pathology reports. |

Let's look at each of the tools in turn and understand how they subserve or address a small component of the bigger research problem.

## CaArray

caArray is an open-source standards-based repository for managing, analyzing and visualizing of gene expression data from microarray experiments. caArray enables researchers to make their microarray data publicly available to the larger cancer research community across geographically separated research centers via a web portal interface as well as through API. caArray uses a number of NCI technologies such as caCORE, caBIO and caDSR. In addition, caArray is built upon a number of caBIG™ compliant standards for data exchange such as Minimum Information About a Microarray Experiment (MIAME), MicroArray and Gene Expression Markup Language (MAGE-ML), MicroArray and Gene Expression Object Model (MAGE-OM) and uses controlled vocabularies based on the Microarray and Gene Expression Database (MGED) Ontology. caArray source code and API are available from NCICB for local installation under an open source license.

> MIAME is a set of guidelines that define the minimum set of data that is needed to enable the unequivocal interpretation of the results of a microarray experiment and to allow researchers the ability to reproduce the results of previously reported experiments. The guidelines include elements of microarray experiments such as aim and brief description of experiment, conditions under which the experiment was carried out, experimental design, quality control procedures used, the experimental protocol used, protocol and conditions used for hybridization and processing of the array, data normalization, extraction and processing protocols, etc.

The MicroArray and Gene Expression (MAGE) group aims to provide a standard for the representation of microarray expression data that would

facilitate the exchange of microarray information between different data systems. This is being done under the aegis of the Object Management Group™ (OMG™), an international not-for-profit consortium defining standards for distributed object computing and interoperable enterprise applications. This has led to the establishment of a data exchange object model (MAGE-OM) and data exchange format (MAGE-ML) for microarray expression experiments. The purpose of the MGED Ontology is to provide standard terminology for the annotation of microarray experiments and to enable unambiguous descriptions of how the experiment was performed.

caArray is available for download at the NCI website at the following URL: http://caarray.nci.nih.gov/. **Fig. 1.2** shows the outcome of a query run on the caArray web portal for an experiment performed by investigators on the classification of complex diseases such as Diffuse large B-cell lymphoma to identify targets for interventional therapy.


**Fig. 1.2.** Querying the caArray web portal for information on Experiments

**Fig. 1.3** shows the results of a query to identify frozen samples (Biosource type) of type "cell" with name "lung" for organism "Homo sapiens" (that is, human samples) supplied by NCI.

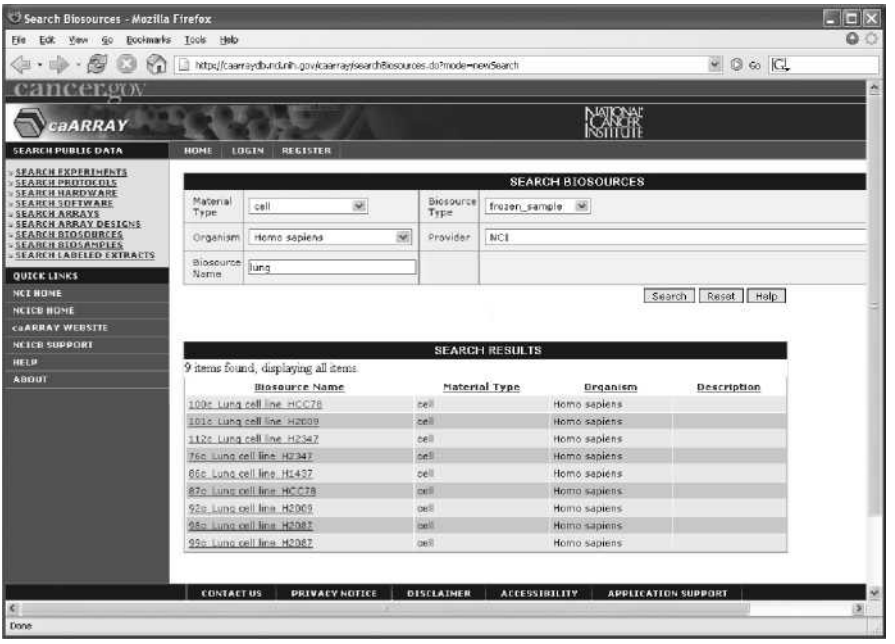


**Fig. 1.3.** Querying the caArray web portal for information on Biospecimens

## CaWorkbench

caWorkbench is a suite of tools for loading, visualizing and analyzing gene expression data and provides the capability to integrate data of different types and from across a number of research institutions. caWorkbench is written with the Java programming language, uses the Java SWING libraries for creating the user interface. It runs on any platform that supports Java 1.5 including Windows XP, Solaris, Linux and OS X 10.5. The software is built on a component based architecture where each feature within the application such as pathways, annotation, expression profiles, etc. is available as a separate component that can be loaded individually when the application is started. caWorkbench is designed to retrieve data from the caArray database via the MAGE-OM API, and utilizes NCICB's caBIO API to access genomic, cancer models, molecular pathway and clinical trials information. caWorkbench is

available for download from the NCI website at http://ncicb.nci.nih.gov/download.

## RProteomics

The goal of the RProteomics project is to build open-source tools and develop standards for proteomics data analysis. As described earlier, Proteomics is the systematic study of the complete complements of proteins expressed by the genome. While gene expression is a study of the process of gene transcription (the synthesis of RNA from DNA), proteomics is the study of the process of gene translation (the synthesis or expression of protein from RNA). The protein machinery constitutes the signal transduction mechanism of a living cell or organism and is responsible for much of the physiological processes that sustain life. Proteomics is therefore a powerful tool in the arsenal of the biologist in the pursuit of molecular mechanisms of disease. Proteomics encompasses the determination of protein expression levels, protein-protein interactions, protein localization, and regulation by post-translational modifications, etc., ultimately to decipher protein function. The basic methodology in proteomics is the separation of proteins in a sample by gel electrophoresis, extracting the proteins of interest and followed by mass spectrometry (MS) to determine their identity and characteristics.

RProteomics derives its name from the open source R software environment that it uses for statistical analyses and visualization of proteomics data. In the future it will also provide a proteomics repository and access to proteomics data via web services. RProteomics includes statistical routines to analyze spectrometric data including algorithms for background curve determination, denoising, peak calibration, normalization of peak intensities, and predictive modeling. RProteomics supports the *mzXML* proteomics data standard and the *MIAPE* (*Minimal Information About a Proteomics Experiment*) standard, the latter of which is being developed by *The Human Proteome Organisation Proteomics Standards Initiative* to standardize data representation in proteomics and facilitate data comparison and exchange.

## cPath

The cBio Pathway Information Resource or cPath is an open source pathway integration and analysis system for integrating protein-protein

interaction and molecular pathway information from multiple sources. It also provides data visualization and analysis functionality via Cytoscape, another open source platform for visualizing interaction networks and integrating them with gene expression profiles. CPath provides access to data via a standard web service query interface that connects with a MySQL database backend as well as a HTTP based web service. Java and is based on a *3-tier architecture* using Java *servlets* and *Java Server Pages* (*JSP*). We will learn more about the Java servlets and JSP technology in chapter 4. Briefly, servlets and JSP provide a server and platform independent mechanism to create web-based applications that can serve dynamic web content.

## CaTissue Core, caTissue Clinical Annotation Engine and caTIES

The simple research scenario we outlined earlier assumes that researchers can locate the biospecimens or tissues samples with the matching disease pathology or disease parameters so to perform the necessary follow-up and validation studies. For example, researchers may want to query a database for biospecimens that have associated gene expression data for a gene or set of genes that may be differentially expressed under a specific disease condition. Under caBIG™, the functionality to manage, annotate and identify matching biospecimens that may be present in a federated manner in geographically dispersed research institutions is being done through the caTissue suite of tools - caTISSUE Core (not to be confused with caCORE), caTissue Clinical Annotation Engine and caTIES.

These are some of the most advanced tools that are currently available in caBIG™ in terms of the software development effort as well as in terms of their adoption by a number of cancer centers and research institutions across the nation. We will illustrate the development efforts behind the caTissue Core application to demonstrate how the various elements of the J2EE platform have been applied to create a robust application to facilitate tissue banking operations.

### CaTissue Core

As described earlier, the function of the caTISSUE Core system is to serve as the base or core solution for biospecimen inventory, tracking and basic annotation for use across cancer centers and other institutions with biospecimen resource facilities. In addition, CaTissue Core establishes the foundation of the TBPT object model that represents the tissue banking and pathology domain. Together with the other TBPT applications – caTissue Clinical Annotation Engine and caTIES, caTissue Core constitutes what is called the caTISSUE system, the comprehensive suite of tools for managing the life cycle events and operations of the tissue banking and pathology information domain.

The caTISSUE Core application is comprised of an n-tiered architecture (**Fig. 1.4**). The presentation tier consists of a web interface as well as HTTP based Java API. The web application used Java Server Pages (JSP) technology to serve dynamic web content. The HTTP API enable users to access all caTissue Core functionality that is available through the web based application. The web-based user interface is designed using the Apache Struts framework following the Model-View-Controller (MVC) Model 2 design approach. The Model 2 approach is a variation of the classic Model-View-Controller (MVC) design paradigm we described earlier. Applied to the Java servlet and JSP technology, under Model 2, the execution of the business logic is managed by the servlet and the presentation logic is managed by the JSPs. CaTissue Core also uses the Tiles framework which specifies the layout of each JSP page using templates and provides a mechanism to manage and reuse the various visual components such as the headers, footers and navigational elements of individual web pages. The caTissue Core business tier contains domain objects and model classes where the tissue banking related business logic resides. The Persistence tier is a local database for storage of tissue banking data, as well as, external data sources such as NCI's Cancer Data Standards Repository (caDSR) and Enterprise Vocabulary Services (EVS).

CaTissue Core provides two mechanisms for interaction between the user interface and the backend data stores - through an Object-Relational Mapping (ORM) tool called *Hibernate* and through the Java Database Connectivity (JDBC) API. Hibernate is used to define the mapping between Java classes to the tables in a relational database in order to persist the objects in a relational database. JDBC API provide database-independent connectivity and access to a wide range of SQL databases as well as other types of data sources, including spreadsheets and flat files. caTissue Core provides support for Oracle as well as MySQL databases.

The caDSR and EVS are a set of resources and tools to describe biomedical data and concepts in standardized ways using Common Data Elements (CDE) and controlled vocabulary, respectively. Access to these services is provided through the caCORE API. We will learn more about these resources in Chapter 6.

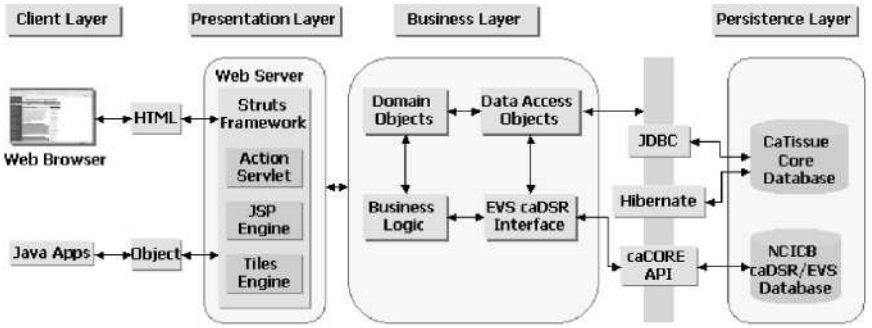


**Fig. 1.4.** caTissue Core n-tier architecture

## Summary

This Chapter provides a brief introduction to The Human genome Project, perhaps the single most important event in the history of medicine after the elucidation of the double-helix structure of the DNA and to the fields of Bioinformatics, Genomics and Proteomics. While computing technology and software have played a fundamental role in the advancements that medical research has made in the last few decades, they have also led to problems in data quality. The silo approach that the biomedical research enterprise has taken has led to isolation of critical scientific expertise and knowledge, depriving patients of the benefits of modern science. To correct these issues, and to bring speedier benefits to individuals with cancer, the NCI in partnership with its Center for Bioinformatics and a number of Cancer Centers across the country launched the caBIG™ program with the aim of providing scientists with the infrastructure and resources to better control, share, assimilate and integrate data from disparate sources. The chapter also provides an overview of the role that the J2EE platform has played in biomedical research especially with the advent of the Internet age and the availability

of the WWW as a catalytic medium for the sharing of resources across space. We also provided examples of a few software applications that demonstrate the power of the J2EE platform

In the next Chapter, we will build on the understanding we have gained so far of the state of and the challenges faced by the biomedical enterprise and begin the exercise of understanding how software is built using the J2EE platform. We will illustrate this by building an application using the Java *Swing* library to run biological sequence searches using the NCBI BLAST engine.

## Questions and Exercises

1. Trace the origins of the Human Genome Project beginning from the elucidation of the structure of DNA in 1953. What were some of the landmark events and technologies associated with the successful sequencing of the human genome?

2. Visit the caBIG™ website to learn more about its organization and activities. Identify the main reasons behind the launch of the caBIG™ project. What are the technological and social hurdles that caBIG™ has to overcome in order to be successful? How will caBIG™ transform medicine if it meets its goals?

3. Compare HGP and caBIG™. What are some of the parallels you can draw between the two projects? Think about how these projects contribute to understanding of disease, especially cancer, and the advancement of modern medicine.

4. What tools and technologies are being created by the NCICB and participating cancer centers to advance the caBIG™ mission? What role does J2EE play in this effort?

## Additional Resources

- Apache Struts - http://struts.apache.org/index.html

- caBIG™ Compatibility Guidelines - http://cabig.nci.nih.gov/guidelines_documentation
- caDSR - http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview/cadsr
- CaWorkBench - http://wiki.c2b2.columbia.edu/workbench/index.php/Main_Page
- EVS - http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview/vocabulary
- Genscan - http://genes.mit.edu/GENSCAN.html
- Global GridForum - http://www.gridforum.org/
- Hibernate - http://www.hibernate.org/
- HGP (US Department of Energy site) - http://doegenomes.org/
- MAGE-ML - http://www.mged.org/Workgroups/MAGE/mage-ml.HTML
- MGED Ontology - http://mged.sourceforge.net/ontologies/MGEDontology.php
- NCBI BLAST - http://ncbi.nih.gov/BLAST/
- OGSA - http://www.globus.org/ogsa/
- OMG - http://www.omg.com/
- The OGSA-DAI project - http://www.ogsadai.org.uk/
- Unified Modeling Language - http://www.uml.org

## Selected Reading

Initial sequencing and analysis of the human genome. Lander et al. Nature. 2001 Feb 15;409(6822):860-921.

The sequence of the human genome. Venter, JC et al. Science. 2001 Feb 16;291(5507):1304-51.

The caCORE Software Development Kit: streamlining construction of interoperable biomedical information services. Phillips J, Chilukuri R, Fragoso G, Warzel D, Covitz PA. BMC Med Inform Decis Mak. 2006 Jan 6;6:2.

Covitz PA, Hartel F, Schaefer C, De Coronado S, Fragoso G, Sahni H, Gustafson S, Buetow KH. caCORE: a common infrastructure for cancer informatics. Bioinformatics. 2003;19:2404–2412.

Common data element (CDE) management and deployment in clinical trials. Warzel DB, Andonaydis C, McCurry B, Chilukuri R, Ishmukhamedov S, Covitz P. AMIA Annu Symp Proc. 2003; 1048.