

JAVA and the Wild Wild Web for Mainframe DBA's

Baltimore/Washington DB2 Users Group
December 10, 2003

Murray Wood
Murray.Wood@ca.com

Agenda

- The Web's Impact on the DB2 DBA
- Java and Web Speak – A quick translation
- JDBC vs SQLJ – Things a DBA should know
- Java Stored Procedures and EJBs
- Performance

- Opening up to the big wide world
 - Traditional users → internet users
 - 24x7 is critical
 - Recoverability (application and disaster recovery)
 - Data Sharing
- In a nutshell, the DBA has a new set of challenges

The Mainframe and the Web

- E-business is about integration
 - 20-30 years of IT investment on the mainframe
 - Leverage your existing IT assets
 - 70 – 80% of world's data reside on IMS and DB2
- Yes, you get to hear the buzz words ... (again)
 - **Scalability, security and availability**
 - Internet traffic is unpredictable
 - S/390 offers unlimited growth
 - Parallel Sysplex cluster can link up to 32 servers to create a SINGLE large computing resource
 - Mainframe (e-server) can handle the peaks and troughs
 - WLM – workload manager

The Mainframe and the Web

- How often do you re-boot your mainframe ?
 - Once a day, twice a day or whenever?
- 30 years of solid infrastructure
 - (schedulers, change control, security etc)
- Man power
 - How many people does it take to look after a mainframe vs 1000 servers?
- **Telia.net**
<http://www-1.ibm.com/industries/telecom/doc/content/news/pressrelease/260811102.html>

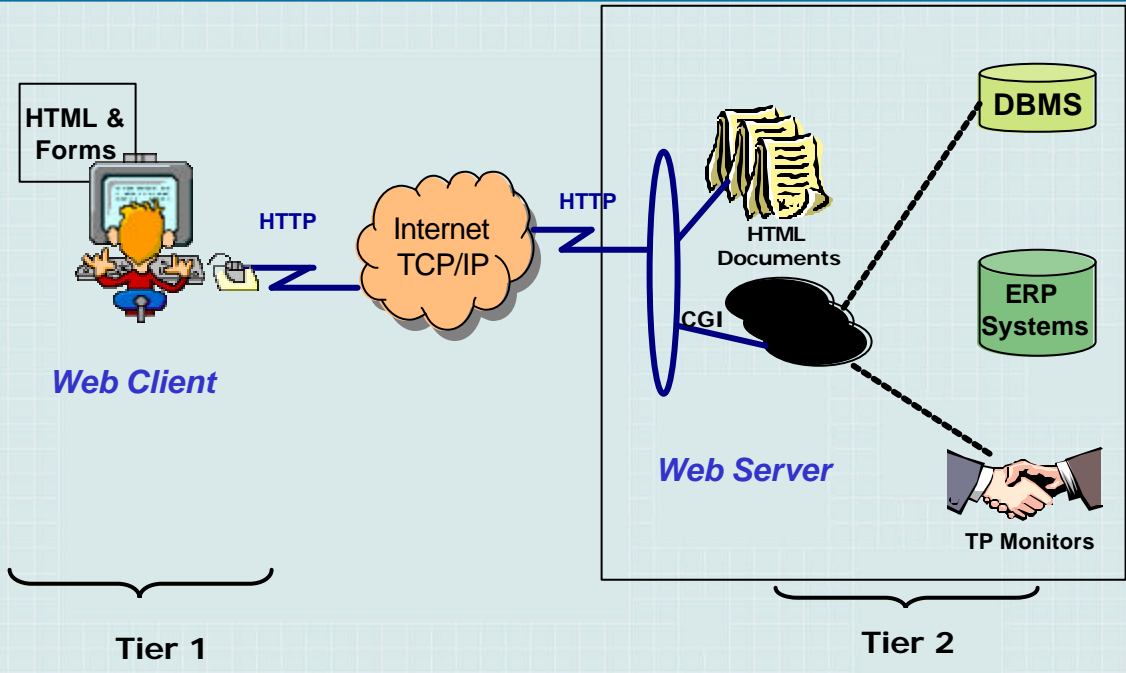
ca.com



Excerpt from IBM's Telia Success Story to be found at <http://www-1.ibm.com/industries/telecom/doc/content/news/pressrelease/260811102.html>

“As a result of its technical and commercial success with Linux on an IBM S/390, Telia Net has decided to upgrade its existing mainframe to an eServer z900, the model that became available in December 2000. The upgrade will increase by 240 percent the capacity of its current installation, which until now has allowed Telia Net to run up to 1.500 virtual Linux internet servers simultaneously.”

Web Components



HTTP and HTTPS

■ HTTP

- Application level protocol for distributed, collaborative, hypermedia information systems
- It is generic and “stateless” object-orientated protocol
 - “stateless” means it does not keep track of connections
- Based on a request/response system algorithm
- Web browsers use HTTP to communicate with Web Servers

■ HTTPS

- Secure Sockets Layer provides security
 - Many web sites use SSL to get confidential info eg. credit card numbers.
 - URLs that use an SSL connection start with *https*: instead of *http*:

cs.com



SSL

Short for **Secure Sockets Layer**, a [protocol](#) developed by [Netscape](#) for transmitting private documents via the [Internet](#). SSL works by using a public [key](#) to [encrypt](#) data that's transferred over the SSL connection. Both [Netscape Navigator](#) and [Internet Explorer](#) support SSL, and many [Web sites](#) use the protocol to obtain confidential user information, such as credit card numbers. By convention, [URLs](#) that require an SSL connection start with *https*: instead of *http*:

Another protocol for transmitting data securely over the [World Wide Web](#) is [Secure HTTP \(S-HTTP\)](#). Whereas SSL creates a secure connection between a client and a [server](#), over which any amount of data can be sent securely, S-HTTP is designed to transmit individual messages securely. SSL and S-HTTP, therefore, can be seen as complementary rather than competing technologies. Both protocols have been approved by the [Internet Engineering Task Force \(IETF\)](#) as a [standard](#).

CGI (Common Gateway Interface)

- Specification for **transferring information between the www and CGI program**
- A **CGI program is ANY program** designed to accept and return data that conforms to the CGI specification.
- The program could be written in any programming language ie. C, Perl, Java, Visual Basic
- CGI programs are the most common way for Web servers to interact dynamically with users.
- Many HTML pages that contain forms, use a CGI program to process the form's data once it's submitted.
- The use of **CGI is a *server-side* solution** because the processing occurs on the Web server.
- One problem with CGI is that **each time a CGI script is executed, a new process is started**. For busy web sites, this can slow down the server noticeably.
- **A more efficient solution is to use Java Servlets**

cs.com



When the internet first emerged it consisted of static contents written in HTML (Hypertext Markup Language). But this did not last very long however. Very soon dynamic web contents were made available via CGI (Common Gateway Interface) technology. CGI enabled the web server to call an external program and pass HTTP request information to that external program to process the request. The response from the external program is then passed back to the web server which forwards it to the client browser.

CGI programs are written in any language that can be called by the web server, and Perl, became a popular choice.

As the internet grew more popular the number of people visiting popular web sites increased exponentially and it was obvious that CGI could not deliver scalable internet applications.

The flaw in CGI is that each client request makes the web server spawn a new process of the requested CGI program. Process creation is an expensive operation that consumes a lot of CPU cycles and memory.

JavaScript vs Java

- **JavaScript (Netscape) and JScript for (Microsoft)**
 - Cross Platform object-oriented language
 - Ability to interact with HTML forms
 - Validate user input
 - Improve client side performance, by reducing the number of requests flowing over the network
 - Not recommended to use JavaScript/JScript on the server side due to variations between Microsoft and Netscape
- **Java Applet**
 - Java program downloaded from the web server and runs on the browser
 - Applets rarely consist of one CLASS file
 - JARs (Java Archive File) – packages the class files into one package (hence reducing the number of requests to the server)

cs.com



Java was developed by James Gosling and his team at Sun Microsystems. Java is based on C and C++ and was originally intended for writing programs that control consumer appliances like toasters, microwave ovens etc. The language was first called OAK, named after the oak tree outside Gosling's office, but that name was already taken so the team renamed it to Java.

Java has been described as a Web programming language because of its use in writing programs called applets that run within a web browser. The applet feature is unique to Java. Applets allow more dynamic and flexible dissemination of info on the internet, and this feature alone makes Java very attractive. A Java application is a complete standalone program that does not require a web browser.

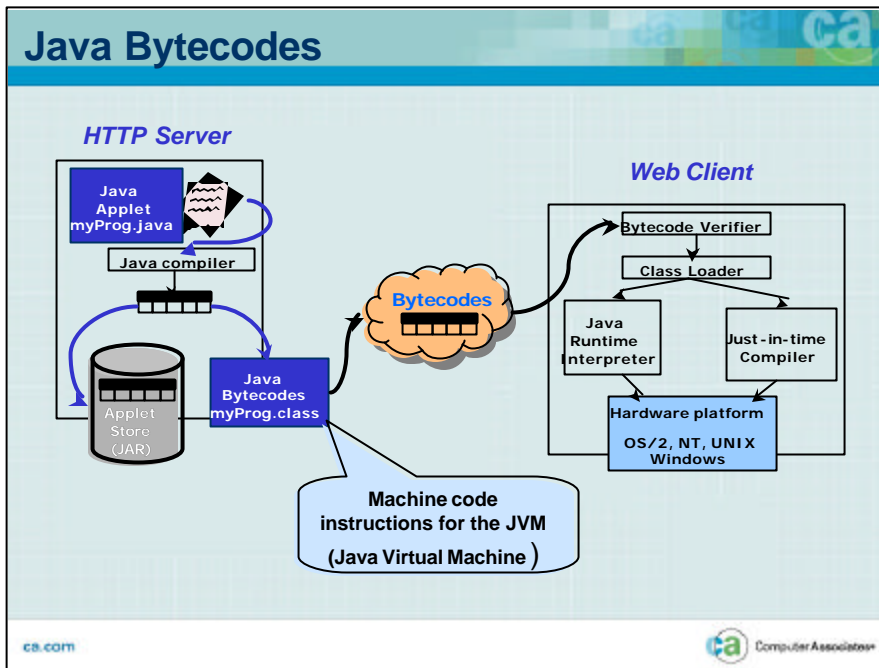
The language designers of Java took a minimalist approach; they included only features that are indispensable and eliminated features that they considered excessive or redundant. The minimalist approach makes Java easier to learn than other Object orientated (OO) languages.

Java Notes:

- Introduced by Sun in 1995
 - Simple and object orientated
 - Similar to C (but does not allow direct memory manipulation)
 - Large Library of predefined objects from other applications
 - Encourages the reuse of objects from other applications
- Platform Independent
 - Java will run on any platform that has a JVM (Java Virtual machine) – All browsers have a JVM
- Network Friendly
 - Works well with browsers and the Internet
- Robust and Secure
 - JVM monitors and disallows malicious code

Java Performance Advantages:

- Supports multi threaded architecture (it does not create a new process each time a servlet is executed) – unlike CGI
- Performs faster than scripting languages e.g Perl



With most programming languages you either compile or interpret a program so that you can run it. Java is unusual in that you both compile and interpret a java program. With the compile you first translate a program into an intermediate language called byte codes (Platform -independent codes which are interpreted by the interpreter on the Java platform.) The interpreter parses and runs each Java bytecode instruction on the computer. Compilation happens just once. Interpretation occurs each time the program is executed.

Think of Java bytecodes as the machine code instructions for the JVM (Java Virtual Machine). Every JVM whether it's a development tool or Web browser that is able to run applets is an implementation of the JVM.

Java bytecodes are "write once, run anywhere"

```
<html>
<head>
<title></title>
</head>
<body>
<h3>Bazla's Personal Info </h3>
<applet code="appBazla.class" archive="appBazla.jar" width="300" height="300">
</applet>
</body>
</html>
```

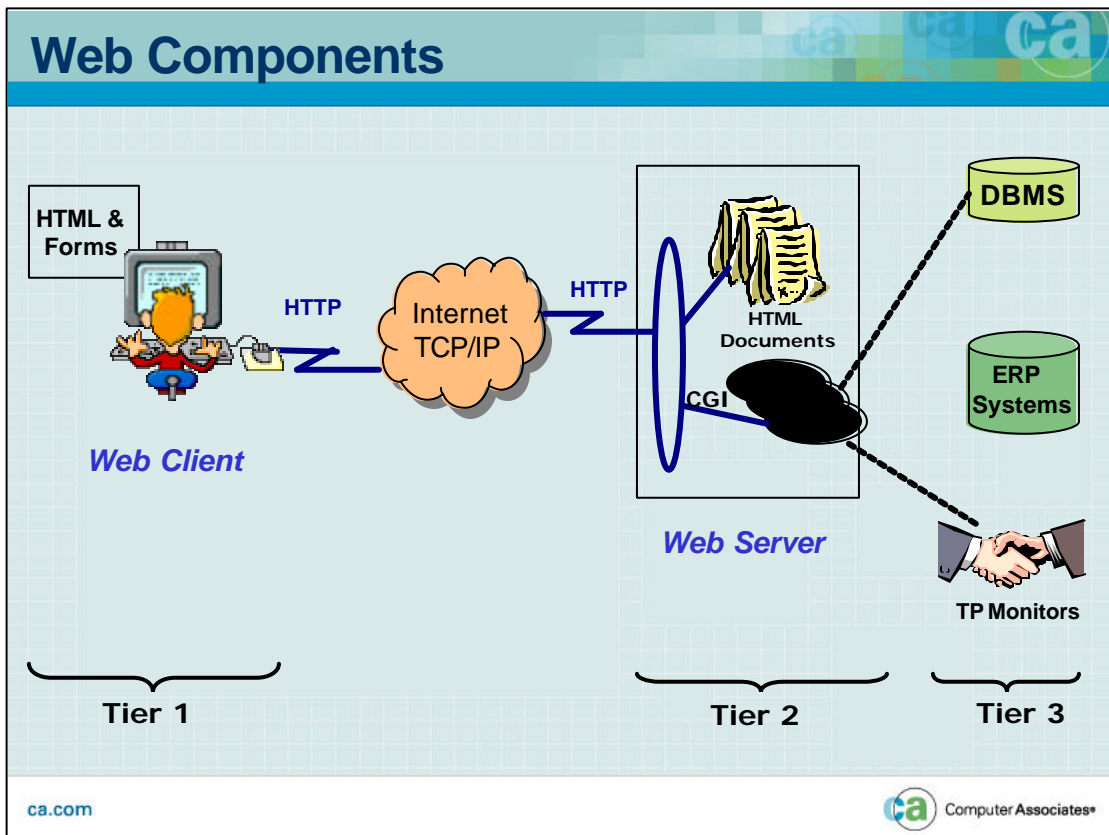
HTML calling a .JAR file

TAG

(noun) A [command](#) inserted in a document that specifies how the document, or a portion of the document, should be [formatted](#). Tags are used by all format specifications that store documents as text files. This includes [SGML](#) and [HTML](#).

(verb) To mark a section of a document with a formatting command

Web Components



The 3 – Tier Architecture is based on a distributed environment where any number of tiers of application logic and business services are separated into components that communicate with each other across the network. This means that there is a logical, but not necessarily a physical, separation of processes.

A 3-tier Architecture looks as follows:

Tier 1 – Presentation (client tier)

The client contains logic pertaining to the presentation of information and performs requests to applications through a browser or Java Applet.

The main function is to present info and results produced by an application to the user. Clients are also referred to as 'thin' clients i.e. they have little software installed on the client machine. Client machines can be PDAs, smartcards, digital wireless phones, network computers, PCs etc. Clients are implemented with industry-standard technologies which: interact with the user, communicate the middle tier and send/receive standard data formats.

The trend right now is to simplify the access to the internet with smaller devices, which can be moved or carried around easily, while still giving the ability to access data or applications e.g. Stock purchases (financial), stock levels at a supermarket, etc

Tier 2 – Web Application (middle tier)

Contains business logic and processes which control the reading and writing of data

Middle tier hosts many servers and services including:

- Web Servers
- Web Application Servers
- Transaction Servers
- Servlets
- JavaBeans
- Connectors

The middle tier servers include a standards-based Web Server to interact with the client tier and define user interaction and a Web Application Server to execute business logic independently of the client type and the user interface style.

The Web Application server is the platform that provides the runtime environment for the application's business logic. It is implemented using Internet and Java technologies, including HTTP server and Enterprise Java services that facilitate rapid development and deployment in a distributed environment.


Java Servlets, JavaServer Pages and Enterprise Java Beans are some of the components deployed in the Web Application Server. These server-side components communicate with their clients and other applications via HTTP (Hypertext Transfer Protocol) or IIOP (Internet Inter-ORB Protocol) and use the directory and security services provided by the network infrastructure.

Tier 3 – Enterprise Information Systems (third tier)

Contains the data and the transactional applications used by the Web Application Server processes.

Servlet, JSP

- **Java Servlet**
 - Replace CGI based techniques in web programming
 - Only run on Web Server
- **JSP (Java Server Page)**
 - Simplify the process of creating web pages
 - JSPs contain HTML
 - Insert dynamic content into web pages. For example
 - Answer to a search (All book titles beginning with “Once upon a time”)
 - List the last ‘n’ products viewed
 - Contain some Java code which encapsulates the logic and generates the page content
 - The java code, may call “beans” to access re-usable components and back-end data

cs.com 

Servlet and JSP offers the following benefits that are not necessarily available in other technologies:

Performance

The performance of servlets is superior to CGI because there is no process creation for each client request. Instead, each request is handled by the servlet container process. After a servlet is finished processing a request, it stays resident in memory, waiting for another request.

Portability

Similar to other Java technologies, servlet applications are portable. You can move them to other operating systems without major problems.

Rapid Deployment Cycle

Servlets have access to the rich Java library which helps speed up the development process.

Robustness

Servlets are managed by the JVM (Java virtual Machine), so you do not have to be concerned with memory leaks and garbage collection

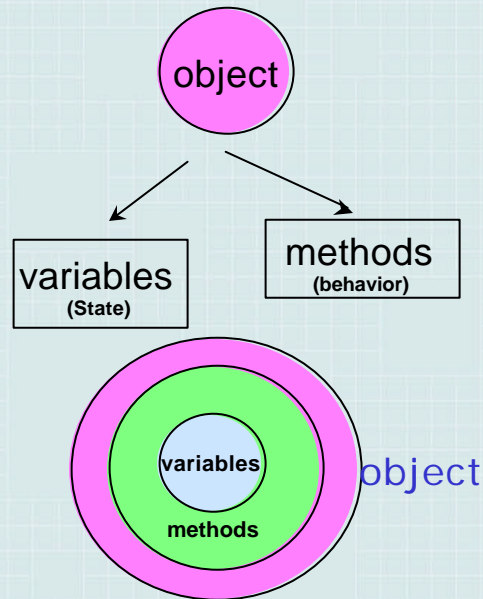
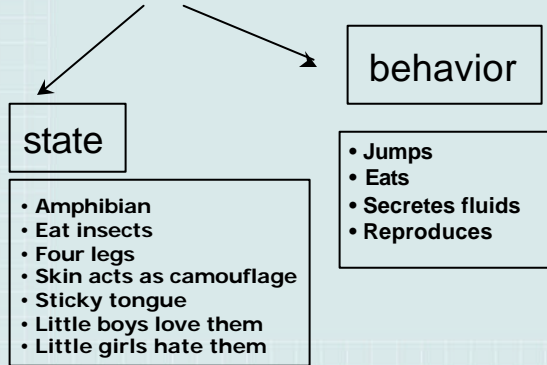
Because Java is widely used technology, nearly all vendors work on Java-based technologies, which means that you can easily find and purchase components to suit your requirements which saves valuable development time.

JSPs (Java Server Pages)

JSPs were released at the same time as servlet technology. JSPs are not meant to replace servlet technology, in fact they are an extension of it. It is common to find both servlet and JSPs in the same web applications.

It is very cumbersome to program with servlets especially when you have to send a long HTML page that includes little code. With JSPs it is much easier to combine fixed or static template data with dynamic content. JSP solves drawbacks in the servlet technology by allowing the programmer to intersperse code with static content.

Object Orientated Programming



ca.com

ca Computer Associates®

To those of use unfamiliar with object orientated languages, lets use the common garden frog to illustrate a point!

The frog has certain characteristics, it also has particular sets of behaviour, just like objects. But with software objects the terminology is different although the concept is the same. With software objects the 'state' is known as 'variables' and the 'behaviour' is known as 'methods'.

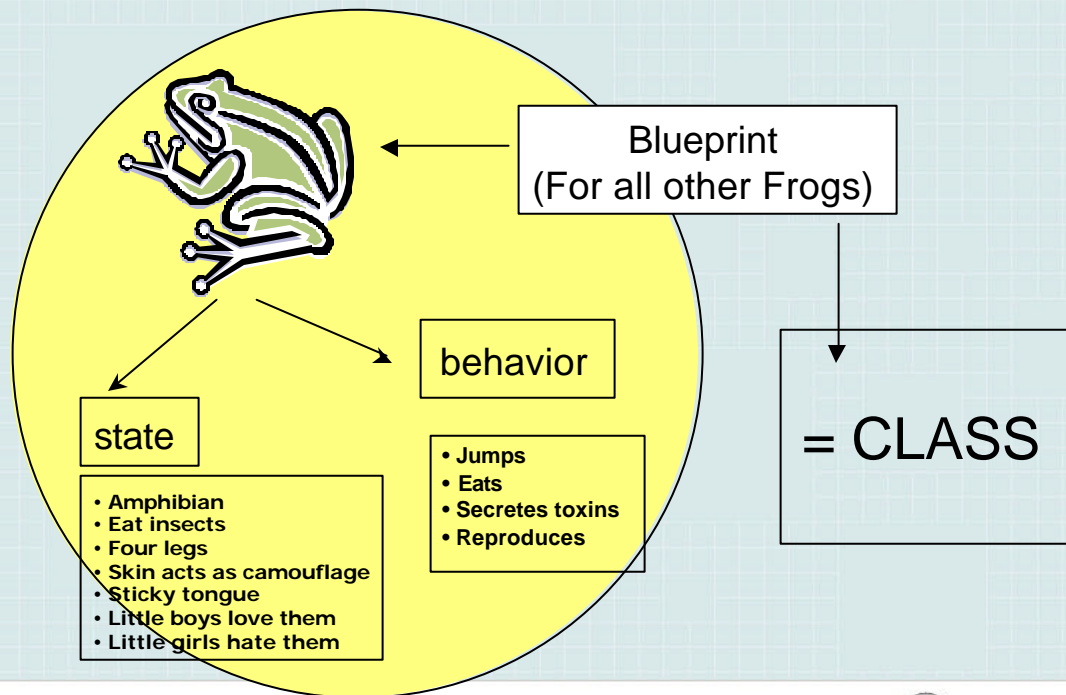
The two most important concepts on object-orientated programming are the 'class' and the 'object'.

An object is made up of variables and methods.

In the the broadest term – the object is a 'thing', both tangible and intangible. A program written in object-orientated style will consist of interacting objects. For a program to maintain bank accounts for a bank we may have many Account, Customer, Transaction and ATM objects. An object is comprised of data and operations that manipulate the data.

Almost all non-trivial programs will have many objects of the same type. Eg. In the bank program we expect to see many Account, Customer and other objects.

Object Orientated Programming



ca.com

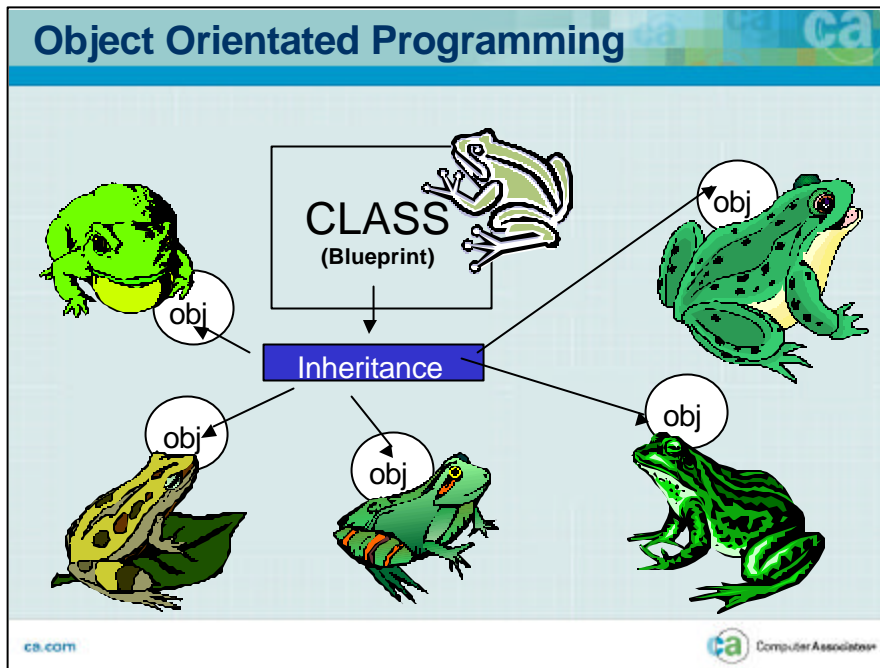
ca Computer Associates®

Inside a program we write instructions to create objects. For the computer to be able to create an object, we must provide a definition, called a 'class'. A *class* is a *template or blueprint* that the computer uses to create objects. An object is called an 'instance' of a class. An object is an instance of exactly one class. An instance of a class 'belongs to' the class.

A class must be defined before you can create an instance (object) of the class.

In writing object-orientated programs we first define classes, and while the program is running, we create objects from these classes to accomplish tasks. A task can range from adding two numbers to computing an interest payment for a loan. To instruct a class or an object to perform a task we send a message to it.

For a class or an object to process the message it must be programmed accordingly. You cannot just send a message to any class or object. You can send a message only to the classes and objects that understand the message you send to the them. For a class or an object to process the message it receives, it must process a matching method, which is a sequence of instructions a class or an object follows to perform a task. A method defined for a class is called a class method, and a method defined for an object is an instance method.



Back to the frog example again ...

Frogs (*subclass*) are members of the zoological class called *Amphibia* (*superclass*). But how does this relate to object orientated programming?

In object-orientated programming we actually use a mechanism called inheritance to design two or more entities that are different but share many common features. First we define a class that contains the common features of the entities. Then we define classes as an extension of the common class inheriting everything from the common class. We call the common class the 'superclass' and all classes that inherit from it, are called 'subclasses'.

A subclass will inherit everything from its superclass. It is not possible to inherit 'some' of the features of the class, ALL the features from the superclass are inherited, but these features can be 'overridden' in the subclass.

Inheritance is not limited to one level. A subclass can be a superclass of the other classes, forming an inheritance hierarchy.

CORBA and LDAP

- **CORBA (Common Object Request Broker Architecture)**
 - Set of conventions, standards and protocols for interprocess communication
 - Developers can write applications for many different OS's at once in any number of languages
 - Any application that matches that has the defined interfaces and protocols is allowed to communicate with another CORBA implementation
- **LDAP (Lightweight Directory Access Protocol)**
 - Repository to retrieve data, resources, addresses etc.
 - These repositories are known as "directories"
 - Finding information for distributed systems without directories would not be easy
 - Each vendor may implement their own version
 - X.500 (adopted by ISO – International Standards Organisation) is a directory standard of choice

cs.com



CORBA (Common Object Request Broker Architecture) is the most important middleware project undertaken in our industry. It is a product of the world's largest software consortium – OMG (Object Management Group) – which includes over 800 companies representing the entire spectrum of the computer industry. A notable exception is Microsoft, which has its own competing product called DCOM (Distributed Component Object Model).

What makes CORBA so important is that it defines middleware that has the potential of encompassing all other forms of existing client/server middleware, especially if it is combined with Java. CORBA uses objects as a unifying metaphor for bringing existing applications together. CORBA is great, because the entire system is self-describing, and the service is always separated from the implementation.

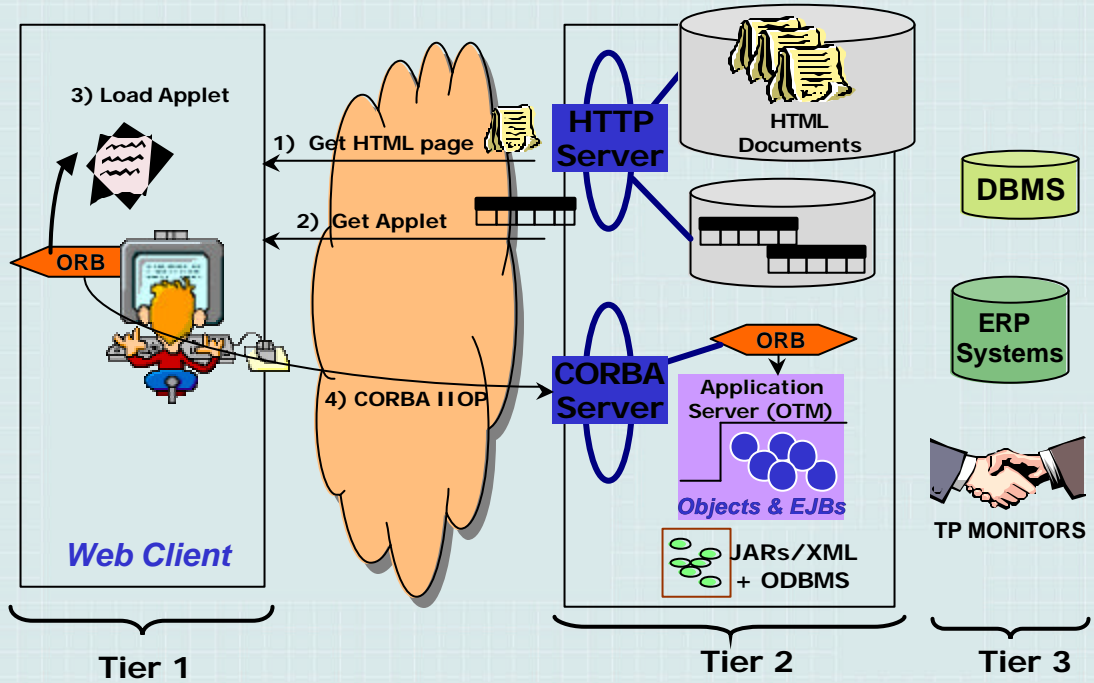
CORBA was designed to allow intelligent components to discover each other and interoperate. It also has an extensive set of services for creating and deleting objects, accessing them by name, storing them in persistent stores, externalising their states and defining ad hoc relationships between them.

LDAP (Lightweight Directory Access Protocol)

Previously a directory service was tied to a single application such as email or to a specific operating system such as NT domains. In contrast a 'universal directory' must be totally independent of both applications and Network Operating Systems. The original industry standard for universal directories was CCITT's X.500. The X.500 standard was written to run on top of the OSI communication protocol. X.500 was implemented fully or partially in a number of products including DCE, Microsoft Exchange, Lotus Notes and Netware Directory service. However, these partial implementations did not inter-operate which was a big problem. Most vendors found X.500 too bulky, so LDAP (Lightweight Directory Access Protocol) was born.

LDAP was designed by Netscape and the University of Michigan to provide a simplified version of X.500 which could run on TCP/IP networks. The thought was to provide a lightweight PC client implementation that could access X.500 directory servers over the Internet.. LDAP became very popular as is now supported by all major directories.

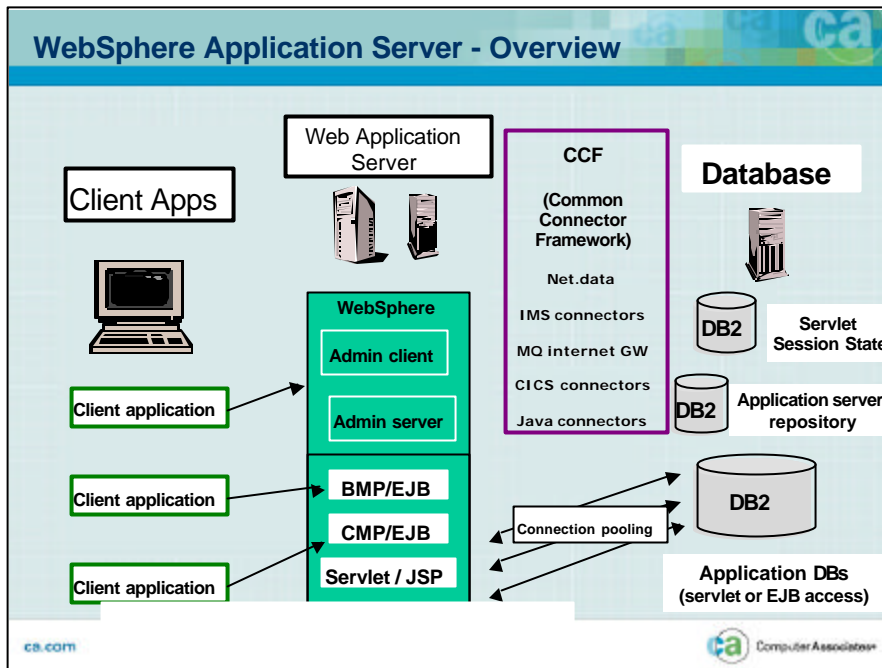
HTTP, CORBA and Java ...



WebSphere

- Application Server

- Sits in the Middle-tier
 - Communicates with the back-end systems e.g. DBMS (DB2, IMS, Oracle etc.) TP Monitors (CICS, IMS, Encina)
 - Many DBMS cannot understand commands written in HTML the WAS (web application server) acts as a translator
 - Communicates with front-end clients (e.g. web browsers)
 - Provides a runtime environment for business logic
- Has naming service and uses JNDI (Java Naming Directory Interface)
- Provides security
 - Controls access to web resources e.g. HTML pages, JSPs, EJBs etc.
- Transactional
- Work Load management
- Implemented on J2EE standards



WebSphere Application Server has a servlet engine that implements the Java Servlet API 2.1. It includes its own packages that extend and add to the Java Servlet API, and leverage JDK 1.2 across all supported operating systems

The Application

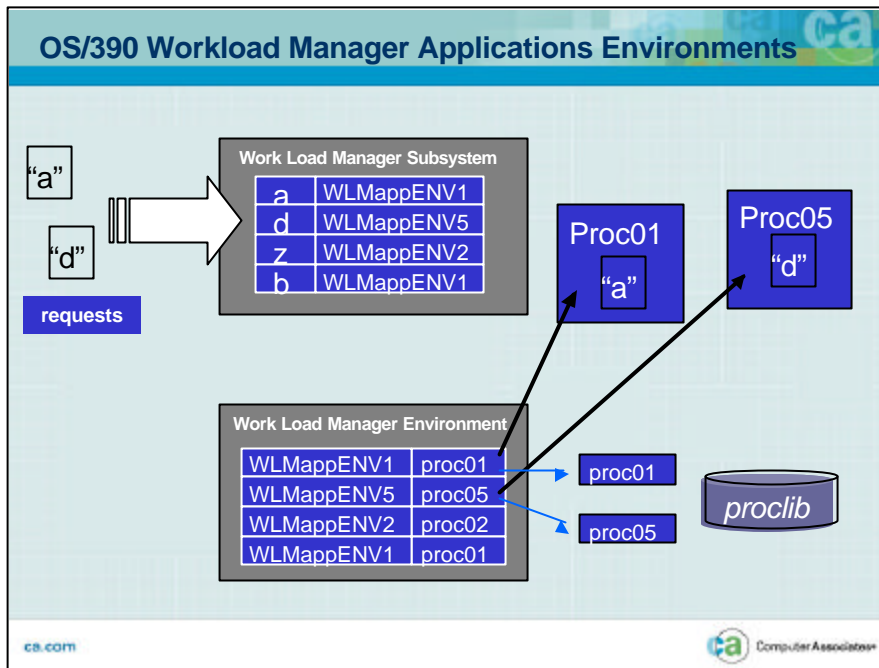
Application servers extend the capabilities of a Web server to handle application requests. Given an application comprised of HTML pages, servlets, and enterprise beans, the application server makes the following exchange possible:

1. A user at a Web browser on the public Internet visits a company Web site. The user requests use of an application that provides access to data in a database.
2. The user request flows to the Web server.
3. The Web server determines that the request involves an application containing servlets and enterprise beans. It forwards the request to IBM WebSphere Application Server.
4. The IBM WebSphere Application Server product forwards the request to one of its application servers on which the application is running.
5. The application processes the user request.
6. The application server collaborates with the Web server to return the results to the user's Web browser.

The WebSphere Application Server provides a servlet engine that implements the Java Servlet API 2.1. It includes its own packages that extend and add to the

Java Servlet API, and leverage JDK 1.2 across all supported operating systems. The extensions and additions make it easier to manage session state, create personalized Web pages, generate better servlet error reports, and access databases. The Application Server supports JSPs, a powerful approach to dynamic Web page content. One of the advantages of JSPs is that they enable you to effectively separate the HTML coding from the business logic in your Web pages. The IBM extensions to the JSP specification include HTML-like JSP tags that make it easy for HTML authors to add the power of Java to their Web pages.

WebSphere Application Server Enterprise Edition offers leading-edge Web services and Java 2 Platform Enterprise Edition (J2EE) programming model extensions which speed application development and deliver the application flexibility necessary to compete in an ever-changing business environment. In addition, Enterprise Edition provides sophisticated integration capabilities which enable messaging to seamlessly connect multiple applications and adapt existing Microstate, C++ and CORBA assets for use within the J2EE application server environment.



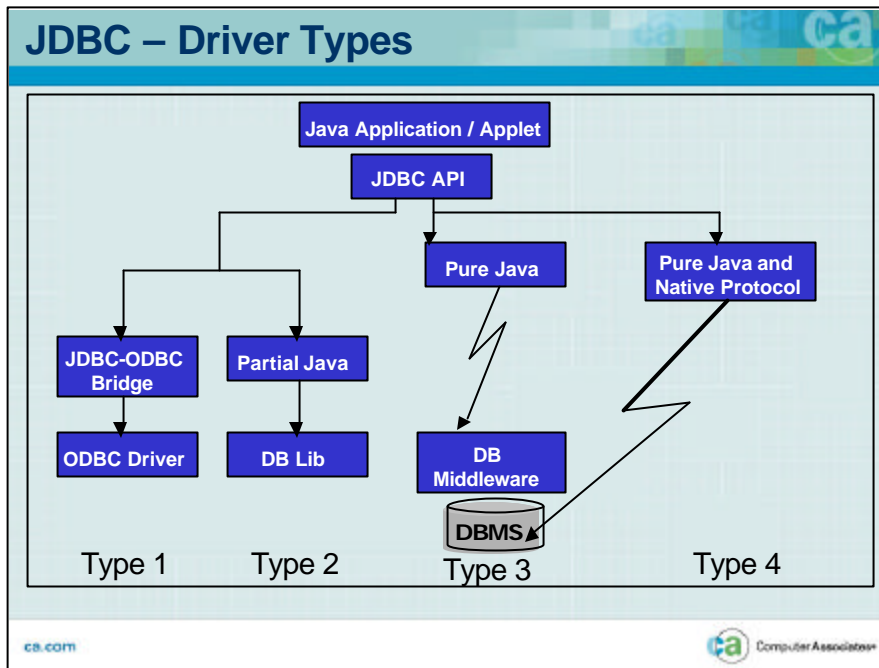
- Work Manager System used by the web server to assign requests to Application Environments
- Server Address Spaces controlled dynamically by WLM
- Uses Enclave SRBs (similar to DB2 – Stored Procedure Address Spaces)

The OS/390 platform is interesting from the perspective of web server management in that it offers unique scaling and workload prioritisation capabilities. This may be important because in an Internet application the user community may be anywhere on the Internet and peak usage may be much higher than in traditional in-house applications.

OS/390 offers the ability to dynamically process application workload using the facilities of Workload Manager (WLM). WLM supports a construct called Application Environments. With WLM managed Application Environments server address spaces may be started and stopped dynamically as deemed necessary by WLM. One usage of the facility is WLM managed stored procedures. Another usage of this facility is for the web server address space management

JDBC (Java Database Connectivity)

- JDBC is a set of APIs
- Provide a generic interface for writing platform-independent applications that can access ANY SQL database
- APIs are defined within 16 interfaces and classes that support
 - Connecting to the database
 - Executing SQL statements
 - Processing Results
- JDBC is similar to DB2 CLI – you do not need precompile or bind a program
- An application written using JDBC uses only dynamic SQL
- JDBC – program executes SQL statements with privileges assigned to the userid running the program



JDBC defines four different ways a JDBC driver can be implemented :

JDBC Driver Type 1 (JDBC-ODBC bridge drivers) was the first driver available on the market. It was provided by Sun itself and is just using an existing ODBC driver. It is the slowest and should only then be used if there is no real driver available for that database product. This type of driver, packaged with JDK, requires an ODBC driver and was introduced to enable database access for Java developers in the absence of any other type of driver.

JDBC Driver Type 2 (Native API partly Java drivers) is available for most databases. It sits on top of a database API (like CLI for DB2). This means it is similar to type 1, but it is vendor specific. The database access code is implemented in a shared library (usually written in C) and loaded by the Java-part of the driver via JNI. This driver requires database software installed on the machine where the application executes. This type of driver uses the client API of the DBMS and requires the binaries for the database client software. This driver offers performance advantages but introduces native calls from the Java Virtual Machine (JVM).

JDBC Driver Type 3 (Net-protocol all Java drivers) is similar to type 2. The difference is that the C-part of the driver is here implemented as a daemon which is accessed via TCP/IP sockets from the Java-part. The advantage compared to type 2 is that there is no database software required on the local machine. This is important for Applets where type 2 cannot be used. The drawback is a worse performance compared to type 2. A generic network protocol is used with this type of driver. Portability is a major advantage of this type of driver, but it has the limitation that it requires intermediate middleware to convert the Net-protocol to the DBMS protocol.

JDBC Driver Type 4 (Native-protocol all Java drivers) is pure Java. It talks over a database native network protocol directly with the database server. It acts like a full database client. This type of driver is portable and uses the protocol of the DBMS.

Driver Types 3 and 4 are well suited for applets that access a database server on an intranet, because they only require Java code to be downloaded. The most common drivers today are Type 2 (generally the best performer) and Type 4 (the best balance of performance and portability).

JDBC

- JDBC - Advantages
 - “Write once, execute anywhere”
 - Ability to change the database the SQL is calling without recoding the Java program
 - JDBC applications do not require pre-compiling

- JDBC – Guidelines
 - JDBC allows getxxx() methods to retrieve a column
 - Using non-matching getxxx() causes a performance overhead
 - Only select/update the necessary columns
 - A Java object is created for each retrieve column
 - Turn autocommit OFF!
 - Else it will force a commit after EVERY SQL statement

JDBC Guidelines

- Use JDBC DataSource connection pooling, thereby
 - Re-using the DB2 connection thread
- Release resources
 - Close ResultSets
 - Close PreparedStatements
 - Close CallableStatements
- Use dynamic SQL caching
 - DYNAMICRULES(BIND)
 - CACHEDYN=YES

SQLJ

- Embeds SQL in Java programs
- An application using SQLJ can only use static SQL
- The SQLJ environment consists of:
 - Embedded SQL
 - Translator (written in Java)
 - A runtime environment
- Set of Java classes that implement SQLJ's runtime support
- SQLJ – when using SQLJ to access a DB2 server, the program will execute SQL statements using the privileges assigned to the user who created the database package

cs.com



SQLJ provides a simplified syntax for JDBC that allows you to write SQL-like statements directly in your Java source code. The SQLJ preprocessor generates static SQL, which provides better performance than dynamic SQL. SQLJ also generates iterator Java classes. These iterators allow you to navigate query results using a very simple “get next” protocol.

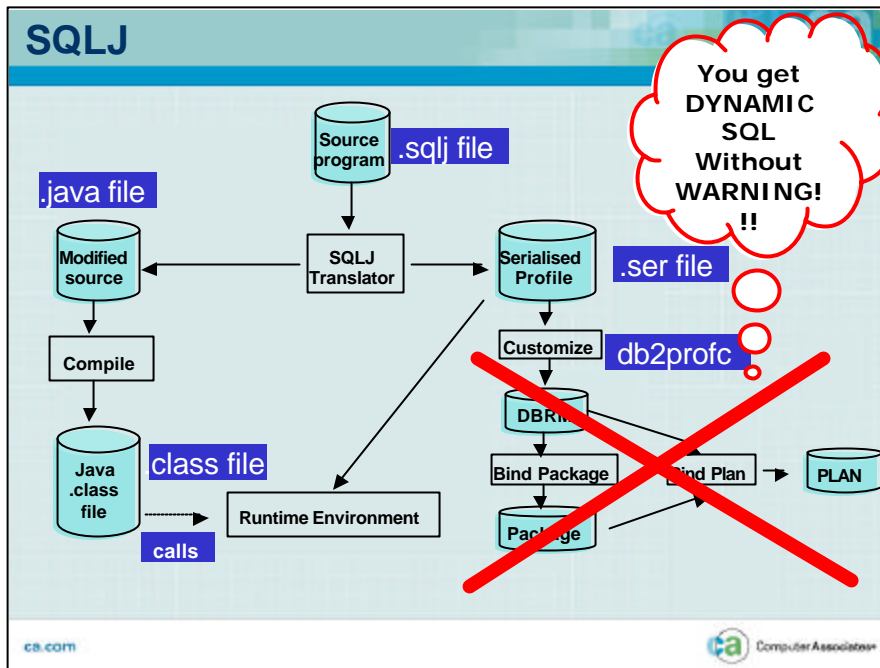
SQLJ support. Provides you with a standard way to embed SQL

statements in Java programs. The SQLJ standard has three components:

• *Embedded SQLJ*

• *A translator:* The translator translates SQLJ files that contain embedded SQLJ to produce .java files and profiles that use the runtime environment to perform SQL operations.

• *A runtime environment.* The runtime environment usually performs the SQL operations in JDBC and uses the profile to obtain details about database connections.



The SQLJ ANSI standard defines a set of clauses to include static SQL in Java programs. It is implemented as extensions to the core Java classes. SQLJ and JDBC are complementary and can be used in the same program.

The standard specifies three components:

- The database language.
- The SQLJ translator. The SQLJ clauses are transformed by the translator into calls to the runtime environment. After translation, the Java program can be compiled normally. The translator produces profile files which contain information about the database schema that must be accessed. The profiles are used at execution time by the SQL runtime environment. The profiles must be customized if the use of vendor-specific features or optimization is needed. With DB2, the profiles can be used to create DBRMs that must be bound into DB2 packages to have a true static SQLJ implementation.
- The runtime environment. This is a thin layer of pure JAVA code that communicates with the DBMS. SQLJ programs can access any DBMS for which a SQLJ runtime implementation exists. By default, the SQLJ runtime performs accesses using JDBC. A SQLJ program can therefore access any DBMS for which a JDBC driver exists. SQLJ also relies on JDBC for tasks like connecting to databases and handling SQL errors. Like JDBC, SQLJ on z/OS supports DB2 attachments to CAF or RRSAF.

SQLJ - Guidelines


- Customise SQLJ serialised profile with online checker
 - Without customisation – SQLJ is executed dynamically !!!
 - Online checking
 - Access DB2 catalog to check SQLJ supported compatibility and convertability
 - Determines the length of string columns (Java has no concept of string length), so see if predicates can be considered for index access

JDBC vs SQLJ

<ul style="list-style-type: none"> ▪ JDBC <ul style="list-style-type: none"> ➢ More portable ➢ Dynamic SQL only ➢ SQL prepared at run time ➢ JDBC manages its own connections to DB2 ➢ Security (“uses userid privileges to access DBMS tables”) ➢ Dynamic SQL statements are cached until they are invalidated 	<ul style="list-style-type: none"> ▪ SQLJ <ul style="list-style-type: none"> ➢ Easier to code ➢ Static SQL only ➢ Better performance ➢ Security (“uses bind owner privileges to access DBMS tables”) ➢ Static SQL is “persistent” ie. They last as long as Package exists ➢ Robustness
---	--

- SQLJ and JDBC can be used in the same application
- How often will the SQL statement be executed?

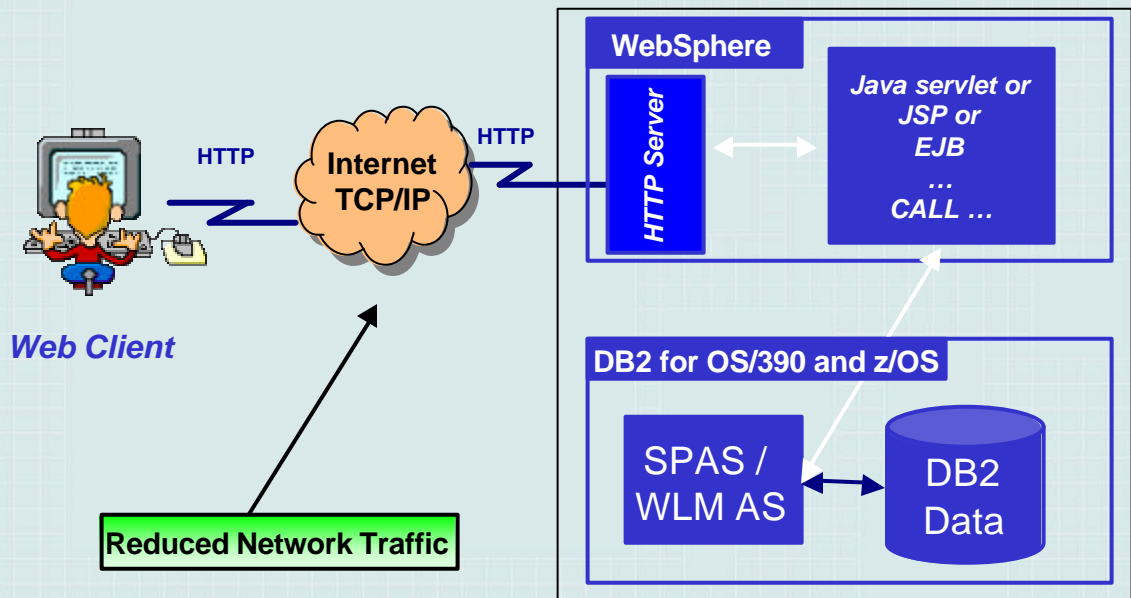
ca.com

 Computer Associates

Most SQL statements in SQLJ are a one-line Java expression. These one line expressions contain familiar “host variable” syntax that make it easy for the programmer to issue SQL statements that reference Java variables

JDBC is a much lower level programming interface. The application programmer has to use “set” and “get” methods to associate Java program variables with SQL statements. A typical SQL SELECT statement can easily take 50-100 lines of code using JDBC. The same SQL statement is a one-line expression in SQLJ.

Stored Procedures



The Active DBMS – SPs

- Reduced network traffic, increased throughput
- If there is a segment of code that is to be executed many times (from many places)
 - A stored procedure is ideal
- They can be executed in parallel by many callers
- Business Logic in Database Server
 - Change management is simplified
 - Easier to control lock contention
 - Increased Security
- STATIC SQL can be used
- Integrates with desktop tools and supported by ODBC, JDBC and SQLJ
- Stored procedures can be written in many languages eg. REXX, COBOL, C, SQL, JAVA etc

ca.com



Reduced network traffic, increased throughput

- A stored procedure can issue many SQL statements, so the number of network send/receive operations are minimised, which improves the elapsed time and CPU time of the application. The SQL issued by the stored procedure uses CALL attach or RRS Attach, so there is no added distributed overhead on the SQL statements. For applications that issue many SQL statements the savings on Elapsed and CPU time can be quite significant.

Application CPU reduced, as the Stored Procedure will run on the database server (which could possibly be OS/390)

STATIC SQL can be used

Lock contention can be more easily controlled

- COMMIT_ON_RETURN (column in SYSIBM.PROCEDURES), allows you to commit as soon as the stored procedure ends. This reduces network messages for a 1-phase commit, by merging the CALL and COMMIT into one network message. For 2-phase commit, two network messages are needed but the COMMIT does not depend on the client program, it happens as soon as the CALL statement has completed

Increased utilisation of database server

What is an EJB ? (Enterprise Java Bean)

- Emerging as the preferred architecture for Java programmers
- EJB extends Java with the following:
 - Security
 - Transaction management
 - Remote method invocation
 - Service improvements
- What do EJB's provide?
 - Unparalleled portability
 - Supports CORBA standard
 - Provides connectors with existing programs/data
 - (with coordinated commit)
 - Access to local and remote objects/methods

cs.com



Since its introduction over two years ago, Enterprise JavaBeans™ technology has maintained unprecedented momentum among platform providers and enterprise development teams alike. That's because the EJB™ server-side component model simplifies development of middleware components that are transactional, scalable, and portable. Enterprise JavaBeans servers reduce the complexity of developing middleware by providing automatic support for middleware services such as transactions, security, database connectivity, and more.

As an example, consider transaction management. In the past, developers have had to either write and maintain transaction management code, or rely on third-party transaction management systems, generally provided through proprietary, vendor specific APIs. In contrast, Enterprise JavaBeans technology enables components to participate in transactions-- including distributed transactions -- simply by specifying which objects and methods are transactional. The EJB server itself handles the underlying transaction management details, so developers can focus specifically on the business purpose of the objects and methods. And because EJB technology is based on the Java programming language, components can be deployed on any platform and operating system that supports the Enterprise JavaBeans standard.

The Enterprise JavaBeans technology model delivers benefits that address the most pressing concerns of enterprise development teams. These include reduced time to market for mission-critical applications, effortless scalability and portability, reduced reliance on hard to find developer skill sets, and an overall increase in developer productivity. EJB technology reduces the cost of developing enterprise scale applications, while protecting an organization's existing investment in IT resources

The EJB Container


Enterprise beans are software components that run in a special environment called an EJB container. The container hosts and manages an enterprise bean in the same manner that the Java Web Server hosts a servlet or an HTML browser hosts a Java applet. An enterprise bean cannot function outside of an EJB container. The EJB container manages every aspect of an enterprise bean at runtimes including remote access to the bean, security, persistence, transactions, concurrency, and access to and pooling of resources.

The container isolates the enterprise bean from direct access by client applications. When a client application invokes a remote method on an enterprise bean, the container first intercepts the invocation to ensure persistence, transactions, and security are applied properly to every operation a client performs on the bean. The container manages security, transactions, and persistence automatically for the bean, so the bean developer doesn't have to write this type of logic into the bean code itself. The enterprise bean developer can focus on encapsulating business rules, while the container takes care of everything else

EJB Architecture – The 4 components

- **EJB client** (*component - No.1*)
 - Process that requires the service provided by the bean
 - Client does not need to store the location of the EJB

- **Application Server** (*component - No.2*)
 - Provides ALL the underlying services required by the EJB. These include:
 - *Transaction Services*
 - *Naming Services*
 - *Database Access Services*
 - *Security Services*
 - *Life Cycle and thread management services*

 Computer Associates

EJB client (*component - No.1*)

Process that requires the service provided by the bean

- Client can be an app on a desktop or a Java Servlet or JavaBean

Client does not need to store the location of the EJB

- ONLY requires the unique name to reference the bean
- EJB is located via a network directory service
- When located a reference to an EJBHome object is returned, which is used to obtain the reference to the EJBObject
- Methods can be called on the EJBObject to access the services provided by the EJB

Application Server (*component - No.2*)

Provides ALL the underlying services required by the EJB. These include:

Transaction Services – provide transaction contracts, handle commits/rollbacks, communicate with underlying OLTP if present

Naming Services – provide a directory service so clients can locate EJBs

Database Access Services – communicate with persistent data store

Security Services – Control access to EJBs through authentication services

Life Cycle and thread management services – provide thread support so that EJBs can be multithreaded

EJB Architecture – The 4 components

- **EJB Container** (*component - No.3*)
 - Provides the environment in which the EJB runs. The container manages the following:
 - *Advertising the EJBs are available in this container*
 - *The Life Cycle of the EJB*
 - *Persistence of the EJBs*
 - *Authenticating Clients*

ca.com Computer Associates

EJB Container (*component - No.3*)

Provides the environment in which the EJB runs. The container manages the following:

Advertising the EJBs are available in this container

- The container registers the EJBs when they are loaded with directory service that can be accessed via JNDI for java clients or CORBA service for non-java clients

The Life Cycle of the EJB

- Creates and removes instances

Persistence of the EJBs

- EJBs can be serialised and stored in a persistent store when not in use

Authenticating Clients

- Container controls access to the EJBs through access control lists

EJB Architecture – The 4 components

- And finally the EJB (component - No.4)
 - The EJB component is the Java class (or classes) that represents the business-logic component
 - There are two types of EJB:
 - **SESSION BEANS**
 - Represent a process that will be performed on the server
 - The Client requests a service from a session bean
 - Each client has its own instance of the bean
 - Instances of session beans cannot be shared among multiple clients
 - Two types of SESSION BEAN
 - **Stateless** – eg. Check whether a stock code is valid
 - **Stateful** – eg. Bean provides next available value in sequence (so it must remember the last value returned)

cs.com



There are two basic types of enterprise beans: **entity beans**, which represent data in a database, and **session beans**, which represent processes or act as agents performing tasks. As you build an EJB application you will create many enterprise beans, each representing a different business concept. Each business concept will be manifested as either an entity bean or a session bean

Session beans represent a set of processes or tasks, which are performed on behalf of the client application. Session beans may use other beans to perform a task or access the database directly. The **entity bean** is used to represent data in the database. It provides an object-oriented interface to data that would normally be accessed by the JDBC or some other back-end API. More than that, entity beans provide a component model that allows bean developers to focus their attention on the business logic of the bean, while the container takes care of managing persistence, transactions, and access control.

Session Type Enterprise Beans

Session beans are used to manage the interactions of entity and other session beans, access resources, and generally perform tasks on behalf of the client. Session beans are not persistent business objects as are entity beans. They do not represent data in the database. Session beans correspond to the controller in a model-view-controller architecture because they encapsulate the business logic of a three-tier architecture.

There are two basic kinds of session bean: Stateless and Stateful. Stateless session beans are made up of business methods that behave like procedures; they operate only on the arguments passed to them when they are invoked. Stateless beans are called "stateless" because they are transient; they do not maintain business state between method invocations. Stateful session beans encapsulate business logic and state specific to a client. Stateful beans are called "stateful" because they do maintain business state between method invocations, held in memory and not persistent.

Stateless Session Beans

Stateless session beans, like all session beans, are not persistent business objects as are entity beans. They do not represent data in the database. Instead, they represent business processes or tasks that are performed on behalf of the client using them. The business methods in a stateless session bean act more like traditional procedures in a legacy transaction-processing monitor. Each invocation of a stateless business method is independent from previous invocations. Because stateless session beans are "stateless" they are easier for the EJB container to manage, so they tend to process requests faster and use less resources. But this performance advantage comes at a price; stateless session beans don't remember anything from one method invocation to the next.

An example of a stateless session bean is a CreditService bean, representing a credit service that can validate and process credit card charges. A Hotel chain might develop a CreditService bean to encapsulate the process of verifying a credit card number, making a charge, and recording the charge in the database for accounting purposes. Below are the remote and home interfaces for the CreditService bean

Stateful Session Beans

Stateful session beans are dedicated to one client and maintain *conversational-state* between method invocations. Unlike stateless session beans, clients do not share stateful beans. When a client creates a stateful bean, that bean instance is dedicated to service only that client. This makes it possible to maintain conversational-state, which is business state that can be shared by methods in the same stateful bean

EJB Architecture – The 4 components

- And finally the EJB (component - No.4)
 - **ENTITY BEANS**
 - **Map a Java class to a data source**
 - Source can be a single row in a database; an entire table; some type of data not stored in a database
 - **Each Entity Bean has a primary key associated with it that identifies the data within.**
 - **As it is difficult to control changes to multiple copies of the same data**
 - So only one instance of an entity bean exists for any given primary key
 - **Two types of ENTITY BEAN**
 - **CMP (Container managed persistence)** – Simplest form and relies on the container to provide all database access calls
 - **BMP (Bean managed persistence)** – provides all the database access calls within the bean itself. The disadvantage this, is that the bean is very closely tied to the underlying architecture

cs.com



There are **two types of entity bean**: **Container-Managed Persistence (CMP)**, and **Bean-Managed Persistence (BMP)**. With CMP, the container manages the persistence of the entity bean. Vendor tools are used to map the entity fields to the database and absolutely no database access code is written in the bean class. With BMP, the entity bean contains database access code (usually JDBC) and is responsible for reading and writing its own state to the database. BMP entities have a lot of help with this since the container will alert the bean as to when it's necessary to make an update or read its state from the database. The container can also handle any locking or transactions, so that the database maintains integrity.

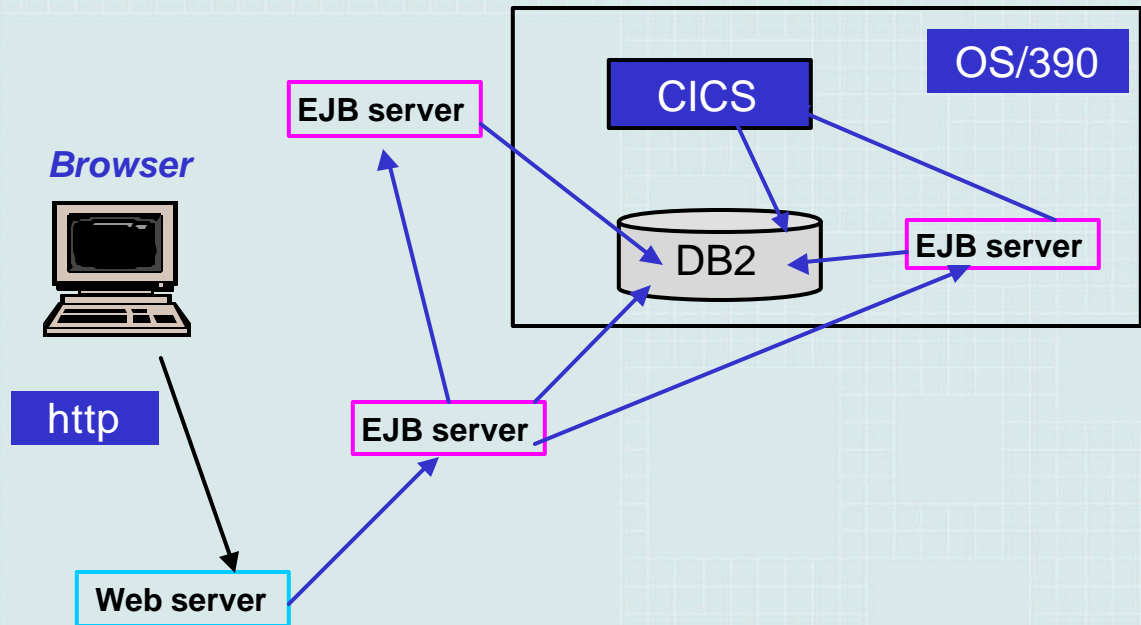
Container-Managed Persistence (CMP)

Container-managed persistence (CMP) beans are the simplest for the bean developer to create and the most difficult for the EJB server to support. This is because all the logic for synchronizing the bean's state with the database is handled automatically by the container. This means that the bean developer doesn't need to write any data access logic, while the EJB server is supposed to take care of all the persistence needs automatically--a tall order for any vendor. Most EJB vendors support automatic persistence to a relational database, but the level of support varies. Some provide very sophisticated Object-to-Relational mapping, while others are very limited.

Bean-Managed Persistence (BMP)

The bean-managed persistence (BMP) enterprise bean manages synchronizing its state with the database as directed by the container. The bean uses a database API (usually JDBC) to read and write its fields to the database, but the container tells it when to do each synchronization operation and manages the transactions for the bean automatically. Bean-managed persistence gives the bean developer the flexibility to perform persistence operations that are too complicated for the container or to use a data source that is not supported by the container--custom and legacy databases for example.

DB2 and EJB Transactions



ca.com

Computer Associates®

EJBs (Enterprise Java Beans) present the idea of global transactions. In the above example, we have a web server that invokes an EJB application on an EJB server. This server itself invokes EJBs at two other servers, and it also issues a call directly to DB2 using JDBC and SQLJ. The other EJB servers also issue SQL against DB2 using JDBC or SQLJ. And there is an EJB servers that calls a CICS transaction that issues SQL.

In EJB all of these DB2 interactions can be part of the same global transaction. DB2 has been enhanced to recognise global transactions and share locks across the branches of a global transaction.

Why integrate EJBs and SPs ?

- An EJB programmer could quite easily use JDBC statements? – why use SP's?
 - *Potential for improved productivity and performance*
- **CMP Entity Beans can eliminate (or minimize) the coding involved with JDBC statements, they are not really a substitute for a SP**
- **SPs usually contain business logic to perform a common service on behalf of many applications so they really have more in common with stateless Session beans**
- **So wrapping SPs as stateless Session beans makes sense**
- **SPs should have more than 'one' SQL statement, otherwise they will cost more than a EJB (cost involved in loading the SPs and communicating with it)**

cs.com



An EJB programmer could quite easily use JDBC statements – why use SPs (Stored Procedures)?

Potential for improved productivity and performance

SPs usually contain business logic to perform a common service on behalf of many applications – so, they really have more in common with stateless Session beans

SPs don't directly represent persistent data (they just access it), so the use of Entity Beans are not really necessary

For MORE info go to :

<http://developer.java.sun.com/developer/technicalArticles/ebeans/sevenrules/>

<http://developer.java.sun.com/developer/onlineTraining/EJBIntro/>

EJB = Enterprise Java Bean

SP = Stored Procedures

Performance

- **Separate excessive SQL from heavily used Java programs**
 - Network delays are a killer
 - Network messaging increases DB2 CPU by more than 3 times
 - Stored procedures, User Defined Functions, Triggers, EJBs
- **Static SQL is faster than Dynamic SQL**
 - If a transaction is being run many times and is slightly different (ie. Caching does not take effect)
 - Dynamic SQL is associated with a user
 - End users must have table privileges
 - If the SQL statements will not change from one invocation to another
- **Java Improvements in V7**
 - DataSource API (eliminates connection URLs and driver names in Java apps)
 - JTA (2-phase-commit)
 - Lock sharing for distributed transactions
 - Connection pooling support
 - Local/global transaction support

ca.com



DataSource API support removes JDBC driver names from your application.

```
Context ctx = new InitialContext()  
DataSource ds = (DataSource)ctx.lookup("jdbc/Mydatabase");  
Connection con = ds.getConnection("userid","password");
```

OR

```
Conenction con = ds.getConnection();
```

Performance

- Optimize Java Environment
 - JVM garbage collection and LE/370 parameter settings
- Make sure Java types match DB2 data types
 - Eg. DB2 (SMALLINT) does NOT have a direct mapping to Java (short or boolean)
 - so use DB2 (INTEGER) and Java (INT) instead
- CHAR vs VARCHAR
 - Char
 - Requires java trim() function to get rid of trailing blanks
 - Higher CPU cost for Java programs
 - Varchar
 - Easier for Java program
 - Higher cost in DB2
 - Updates, inserts, logging
- Keep current with JDK releases and JDBC Driver

Summary

- The “internet” is here to stay
- Understand the impact of the internet to DB2
- Remember the 7 second rule?

Bibliography

- IBM Framework for e-business: Technology, Solution, and Design Overview (SG24-6248)
- DB2 for z/OS and OS/390 : Squeezing the Most Out of Dynamic SQL (SG24-6418)
- DB2 for z/OS and OS/390 Version 7 Selected Performance Topics (SG24-6129)
- DB2 for z/OS and OS/390 Version 7 Application Programming and Reference for Java™ (SC26-9932-02)
- DB2 for z/OS and OS/390 Version 7 Administration Guide (SC26-9931-02)
- An introduction to Object-Orientated Programming with Java (second Edition)
 - ISBN 0-07-118195-4
- Client/Server Survival Guide (third edition) ISBN 0-471-31615-6
- Java for the Web with Servlets, JSP, and EJB (ISBN 0-7357-1195-X)
- DB2 for OS/390 Technology Update – Curt Cotner IBM (IDUG NA – 2001)
- DB2, Java and Design for High Performance – John Campbell IBM (IDUG NA – 2002)
- www.java.sun.com
- www.ibm.com/developerworks/java