Encapsulatin

- A language mechanism for restricting access to some of the object's components, means that the internal representation of an object is generally hidden from view outside of the object's definition.

- A benefit of encapsulation is that it can reduce system complexity, and thus increases robustness, by allowing the developer to limit the interdependencies between software components.

Polymorphism

- is a characteristic of being able to assign a different behavior or value in a subclass, to something that was declared in a parent class.

Delegation

- is the implementation of objects that forward certain method calls to another object, a delegate.

Cohesion

- Is a measure of how the methods of a class or a module are meaningfully and strongly related and how focused they are in providing a well-defined purpose to the system.

- A class is identified as a low cohesive class, when it contains many unrelated functions within it. And that what we need to avoid, because big classes with unrelated functions hamper their maintaining.

- You can read easily a high cohesive class and understand its purpose and role in the system.

- Testing and maintaining a high cohesive class will be easier.

Coupling

- Tight coupling is when a group of classes are highly dependent on one another. This scenario arises when a class assumes too many responsibilities, or when one concern is spread over many classes rather than having its own class.

- Loose coupling is achieved by means of a design that promotes single-responsibility and separation of concerns. A loosely-coupled class can be consumed and tested independently of other (concrete) classes.

- Interfaces are a powerful tool to use for decoupling. Classes can communicate through interfaces rather than other concrete classes, and any class can be on the other end of that communication simply by implementing the interface.

Separation of Concerns (SoC)

- Is the process of separating a computer program into distinct features that overlap in functionality as little as possible. A concern is any piece of interest or focus in a program. Typically, concerns are synonymous with features or behaviors. Progress towards SoC is traditionally achieved through modularity of programming and encapsulation (or "transparency" of operation), with the help of information hiding. Layered designs in information systems are also often based on separation of concerns (e.g., presentation layer, business logic layer, data access layer, database layer).

- Every unit int the system needs to have a cleary dfubed responsability and funcionality. This applies to all levels of system.

Differences of Architecture and Design

- Architecture is of higher level of abstraction and concern realizes nonfunctionality.

- Design is of lower level of abstration and concern to convert the "domain model" into a "technical objetct model".

- Design process, the system components internal details are elaborated based on the architecture.

Code Factoring

- Software applications have a tendency to grow in complexity over time. When that happens, they become difficult to extend, maintain and modify. They also become hard to understand.

- Is a modularization technique that deals with the separation of responsibilities in code.

- Factored code has loose coupling and minimal dependencies, making components and code more reusable

JSP Refactoring

- refactoing a lot of redundant formatting logic (java code) in JSP using simple tag files and expression language.

Java bytecode verification

- done when loading the class files by the class loader.

- makes sure that the code doesn't violate access restrictions.

- checks digital signatures.

- checks acess restriction classes (private, public and protected)

One-Tier Systems

Advantages:

- security cause dont has network calls.

- manageability cause system is in one place.

Disadvantages:

- has no scalability no one component can be increase.

- has no extensibility when add new functionality will impact all system.

- has no maintainability cause tightly coupled - bussines logic, GUI logic and database is totaly mixed.

- has no availability when system fails, the entire system is unavaiable.

- Single point of failure.

- can only scalle vertically.

- the biggest weaknesses of one-tier systems are their extensibility, maintainability and scalability.

Two-Tier Systems

Advantages:

- rapid development.

- aplications "prototype" should be Two-Tier.

- not very availability when client app fails, the entire system is not unavaiable. But it database fails, entire system is unavaiable.

- not very extensibility when add new GUI, will not impact the others.

- not very maintainability Changes in the SGDB will be properly propagated, but changes the GUI should be deployment all client worsktations.

Disadvantages:

- has no scalability cause only component can be increase is database.

- has no manageability cause presentation logic desktop in client worstation can't be monitoraded.

- has no security case sensitive in inicializaion files is spread of many clients.

- has no performance when receive high workload.(database will crash)

N-Tier Systems

Advantages:

- scalability when presentation logic or bussines logic into to a servers can be clusterized.

- availability when clusterized tier provide failover.

- extensibility and maintainability when functionality is separated into to differents tiers.

- reliability assure the integrity an consistency of the application and all data as the load increases.

- performance when spread or processing over many servers.

Disadvantages:

- manageability when ties are deploy on servers as well as not easier to monitor the components.

N-Tier Tips:

- when architected N-Tier is designed well, they do not present any maintenance problems. This is because they are highly modular and it is relatively easy to correct problems in one tier without impacting other tiers.

- can also yield high performance. They can be highly optimized systems generally make extensive use of resource management techniques.

- N-Tier applications are designed to be very modular in nature. The tiers and layers separate roles and responsibilities of each component and container. Hence components and containers can be individually tuning and scaled as needed. This results in better performance.

- the primary advantages a N-tiers system has over Two-tier is that N-tiers is more extensible solution.

Resource Management Techniques

Connection Pooling

- Used to acquire and release connections a process more effective that creation and destruction of connections and so on.

Object Pooling (Flyweight Design Pattern)

- Share resources among multiple users.

- Incresase request handling.

- Allow aplication handles more requests.

- Lowered cost.

Object Passivation

- Used to freed up memory for reuse by other services.

- Best possible use of server resources.

Server clusters

- Used where bottlenecks are encountered.

Browser Base Solution (HTML Thin Client)

Advantages:

- Minimal instalations and easy to deploy.

- UI changes are immediately avaiable for all browser users (pc, smartphones, tablets, game consoles, tv, etc)

- Increase scalability.

- portable to all desktop plataforms when using browser W3C compliance (pc, smartphones, tablets, game consoles, tv, etc)

Disadvantages:

- security concerns, cause is exposet to internet and HTTP session manages wekeness.

- confidentiality of code - client can access system code (html and javascript).

- few GUI features.

- issues with cross-browser compatibility and support

- maintainability of application code is not good, if too much of javascript/DHTML features are used.

Desktop Solution or Thick-client (FAT Client)

Advantages:

- Provides very good client security, persons without a client cannot access system.

- Ability to access the client's PC utilities or Native API's.

- Maintainability of application code is good.

- Rich GUI features.

Disadvantages:

- Managed instalations and distribuion.

- UI changes need redistribuition.

- Not portable to all vendor desktor plataforms (pc, smartphones, tablets, game consoles, tv, etc)

WEB-Centric Applications

Advantages:

- easy development.

- concurrency control.

- web security.

- session management.

- simple CRUD transactions.

- read mostlly transactions.

Disadvantages:

- not support transacions (composto e recursivo).

- not support JMS messagess.

JSF

Advantages:

- mvc.

- ui component model.

- multiples front-ends (desktop and mobiles browsers).

- hides all HTTP infrastructure.

- can use IDE RAD environments.

- offers a clean separation between behavior and presentation.

- Automatic event handling (map http requests to component specific).

- Automatic server side validations and conversion.

- Automatic I18N e I10L.

- All features result in reduces development time.

Disadvantages:

- Standart HTML editor will not render the JSF. Need to deploy JSF app to page rendered.

Life Cicle JSF Request:

- 1.Restore View -> 2.Apply Requests -> 3.Process Validations ->

Life Cicle JSF Response:

- 4.Update Model Values -> 5.Invoke Application -> 6.Render Response.

Ajax

- Is a group of interrelated web development techniques used on the client-side to create asynchronous web applications (HTML, CSS, DOM and JavaScript). With Ajax, web applications can send data to, and retrieve data from, a server asynchronously (in the background) without interfering with the display and behavior of the existing page.

Ajax Advantages

-limits the bandwidth use.

-minimize the page reloads upon user requests.

-improves user-experience.

JPA

Advantages:

- isolate application from the database.

- increase manageability of application.

- improve developer productivity.

- ease of development generated correct and effecient persistence logic.

- can outperform hand-crafted SQL, reducing the number of database round-trips.

- jpa entities can be used outside container environment hence promoting re-usability and coding effort.

- jpa entities an be used on both presentation layer and business logic layer, thus reducing coding effort, size of code and maintenance cost associated

- supports lazy loading of objects hence improving performance.

- best integration with EJB 3.

- Used when model has complex relationships between tables.

Disadvantages:

- imply loss of performance cause include additional layer.

- can not explicitly tunning SQL by hand.

- when datastore is not welll suported by ORM providers.

- Developers need skilled in ORM (learnig curve).

JDBC

Advantages:

- Best performance with explicitly tunning SQL by hand.

- is faster cause acess directly SGDB.

Disadvantages:

- longer, hard development and tests durations with explicit code SQL.

EJB 2x

Advantages:

- scalability (stateless session bean pool).

- component security.

- transactional support.

- concurency controll.

- strong asynchronous or synchronous messaging.

- remote calls (all objectos must implements java.io.Serializable for remote comunications).

Disadvantages:

- takes too long to develops.

- run slowly when compared against non-ejb applications counterpart (remote calls, serializations)

EJB 3 Improvements:

- simplify the process of developing ejb reduze significantly overhead using Java Language Annotations as configurationm, improveing developers productivity.

- specification of programmatic defaults, including for metadata, reducing the need for the developer to specify common, expected behaviors and requirements on the EJB container.

- encapsulation of environmental dependencies and JNDI access through the use of annotations, dependency injection mechanisms, and simple lookup mechanisms.

- simplification of the enterprise bean types.

- ejb support inheritance and polymorphism.

- lightweight CRUD operations with JPA EnityManager API.

- enhamced query JPA capabilities.

- life cicle's callback methods can be defined in ejb itself or in a bean listener class.

- interceptor facility listeners for session beans and message-driven beans. An interceptor method may be defined on the bean class or on an interceptor class associated with the bean.

- It is possible to use both CMP and JPA in one application.

EJB Stateless

- server side proxy for the client side.

- transactional support.

- web services end points.

- consuming web services.

- hight volume.

- can use JDBC.

- Use to provide a service e.g. credit card validation, etc..

- Has no concurency problems, there is no share data to be corrupetd

- Cost of remote lookup is significant and it justify use "cache approach".

EJB Statefull

- server side proxy for the client side.

- spend high server resources (hardware and software).

- dedicated resorces to a user for the life time.

- transactional support.

- can't be web services end points.

- consuming web services.

- can use JDBC.

- Do note survive in case of server crash cause they loose their state.

EJB MDB

- Integration JMS bussines process.

- support synchronous e asynchronous(off-line) processing.

- transactional support.

- consuming web services.

- can use JDBC.

- can't be web services end points.

JPA Entity Bean 3.0

- represent business entities.

- persist state in database.

- not support transacions and web services end points, consume web services and use JDBC.

- bidding ORM represent concurrent use of shared data - non-repeateble read, dirt reads an phanton reads.

- performed automatic DML operations - insert, delete, update and select.

- performed automatic locks pessimistic and optimistic.

- Supports bulk/batch updates e delets.

EntityBeans 2.x CMP

Advantages:

- database independency cause ejb container generated all sql cross sgdb.

- use specific features support by ejb container provider.

Disadvantages:

- can not be portable to another ejb container provider.
- no acess and can not modify sql generated.
- sql generated can be the no most efifcient.

EntityBeans 2.x BMP

Advantages:

- portabled to all ejb container provider.
- can be used to acess nonstandart datatype (legacy) database.
- can use nonstandart SQL features of specific sgdb provider.

Disadvantages:

- not be portable to another sgdb, since bean provider will have to account for this in code.
- knowledge of SQL.
- take much time to delevop.

EntityBeans 2.x Tips

- Does not replace de JDBC API, they merely provide an alternative.
- Does not provide bulk/batch updates e delets.
- Can be created without primary Keys.
- the method findByPrimaryKey() of EntityBeans without primary Keys will be unrealiable.
- After an EntityBean instantiated, is not possible to change your primary key.

EJB CMT versus BMT

- When you need a 'fine level of control' over the transactions with 'complex bussines logic transaction', BMT will give you more control than CMT.

EJB Web Services (JAX-WS e JAX-RS)

Advantages:

- productivity expose ejb as web services end points to meet new requiriments.

- productivity consuming web services to meet new requiriments.

Disadvantages:

- disordered application arquitecture mixing integration layer with business.

- potencial security concerns data and information is acessible on external user without container autenticate.

- potencial data validations broken or circumvented.

State Management

Statefull Session Bean

- ejb container.

- timeout.

- best use of resources (passivation).

- better performace (passivation).

- indicated to huge amount of data.

Servlet HttpSession

- web container.

- timeout.

- not indicated to huge amount of data.

Load Balancer and Servlet HttpSession

- Server instances of same product (all tomcat nodes) when clustered replicates session information across JVM (using common database or any other mechanism), but session replication is not implicit between different products (1 tomcat e 1 websphere).

HTTP (HyperText Transfer Protocol)

- is a protocol with the lightness and speed necessary for a distributed collaborative hypermedia information system. It is a generic stateless object-oriented protocol, which may be used for many similar tasks such as name servers, and distributed object-oriented systems, by extending the commands, or 'methods', used."

- Transport mechanism for MIME (Multipurpose Internet Mail Extensions) documents. MIME documents often contain HTML (HyperText Markup Language) code for display in browser windows. HTTP consists of a request sent by the client to the server, followed by a response sent from the server back to the client. HTTP uses TCP/IP as the underlying transport and network protocols (Connection Base and Stateless).

- HTTP is not secure, but can be made secure by using 'HTTP over SSL' or HTTPS.

HTTPS

- Stands for HTTP over SSL. With HTTPS, SSL sits above the TCP-IP layer and below the application protocol layer

Integration

- Web Services, JCA and JMS are used to integrate JEE-base system with external or legacy system.

Web Services SOAP/REST

- integrating Java System to external system not owned or controled by architect.

- heterogeneous system (any other plataform) -> java to any plataform.

- tecnology independent.

SOAP

- Simple Object Access Protocol, is a protocol specification for exchanging structured information in the implementation of Web Services in computer networks. It relies on Extensible Markup Language (XML) for its message format, and usually relies on other Application Layer protocols, most notably Hypertext Transfer Protocol (HTTP).

- A SOAP message is an ordinary XML document containing the following elements:

  - A required Envelope element that identifies the XML document as a SOAP message

  - An optional Header element that contains header information

  - A required Body element that contains call and response information

  - An optional Fault element that provides information about errors that occurred while processing the message

Advantages:

- interoperability.

- loose coupling.

- well define contracts between messagen producters and consumers, ensure changes of internal implementation without client impact.

Disadvantages:

- expense learnig curve.

- verbose representation.

- higher runtime overhead.

WSDL

- Web Services Description Language is an XML-based language that is used for describing the functionality offered by a Web service.

- A WSDL description of a web service provides a machine-readable description of how the service can be called, what parameters it expects, and what data structures it returns. It thus serves a roughly similar purpose as a method signature in a programming language.

UDDI

- Universal Description, Discovery and Integration is an XML-based language registry for businesses worldwide to list themselves on the Internet and a mechanism to register and locate web service applications.

MTOM (Message Transmission Optimized Mechanism)

- Is a method of efficiently sending binary data to and from web services. It use XOP(XML-binary Optimized Packaging) to transmit binary data.

J2EE JAX-RPC

- has its own data mapping model.

JEE JAX-WS

- supports annotations for deploy web services reducing deployment descriptors.

- supports resource injection of Java EE 5 to shift the burden of creating and initializing common resources in a Java runtime environment.

- support synchronous and asynchronous comunication.

- support MTOM.

- support SOAP 1.1 e 1.2 and WS-I Basic Profile 1.1.

- support (server e client) for both a polling and callback handler mechanism when calling web services asynchronously.

- can be either a remote procedure call (RPC) or document/literal style binding.

- provides handler facility used to logging protocol specific information, adding security information to the message and modifying protocol headers.

- use JAXB or databinding

- desacople data binding tecnology from web services programming model.

- support development of RESTful Web Services.

Dynamic Proxy Client Consumer (-wsimport)

- when you are going to invoke has a well-published WSDL and does not change often.

Dispatch API Client Consumer (dynamic invoke)

- where interoperability with older, non WS-I compliant services is desired.

- where you need to work with a data binding technology other than JAXB.

- document exchange scenarios, where the dynamic nature of dispatches is important.

- where there are specific advantages to be gained by not using SOAP protocol.

REST

- Representational state transfer is a style of software architecture for distributed systems such as the WWW that has emerged over the past few years as a predominant Web service design model cause it simpler style.

- Can send free XML,JSON,etc data over HTTP.

Advantages:

- simples programming model.

- lower runtime overhead.

Disadvantages:

- no contracts between messagen producters and consumers.

ESB (Enterprise Service Bus)

- Is a set of infrastructure capabilities implemented by middleware technology that enable an SOA. The ESB supports service, message, and event-based interactions in a heterogeneous environment, with appropriate service levels and manageability.

J2EE API

- Stax is an API for reading and writing XML documents. This Java API will enable you to create bidirectional XML parsers that are fast, relatively easy to program, and have a light memory footprint.

- JAXP (Java API for XML Processing) enables applications to parse, transform, validate and query XML documents using an API that is independent of a particular XML processor implementation. JAXP provides a pluggability layer to enable vendors to provide their own implementations without introducing dependencies in application code.

- JAXB (Java Architecture for XML Binding) provides an API and tool that allow automatic two-way mapping between XML documents and Java objects. With a given Document Type Definition (DTD) and a schema definition, the JAXB compiler can generate a set of Java classes that allow developers to build applications that can read, manipulate and recreate XML documents without writing any logic to process XML elements.

- JAXR is an api for acessing XML registries.

- SAAJ (SOAP with Attachments API for Java) provides the API for creating and sending SOAP messages by means of the javax.xml.soap package. It is used for the SOAP messaging that goes on behind the scenes in JAX-RPC and JAXR implementations.

JMS

- The Java Message Service is a Java Message Oriented Middleware (MOM) API for sending messages between two or more clients. It is a messaging standard (JCP) that allows application components based on the JEE to create, send, receive, and read messages.

- JMS messages are sent to a 'Message Broker' that routes messages. Multiple brokers may be clustered to scale a message service.

- provides an API for services such as persistence, transactions, and verification of messages.

- Java to Java solution.

- specific java component integration.

- synchronous and asynchronous (offline)

Advantages:

- separating the application from the transport layer of providing data.

- not heterogeneous system, cause depend of messages base systems.

- optimed for Java producers and consumers messagens.

- guaranteed delivery.

- loosely coupled and reliable.

Point-to-Point Messaging (PTP)

- Application is built around the concept of message queues, senders, and receivers. Each message is addressed to a specific queue, and receiving clients extract messages from the queue(s) established to hold their messages. Queues retain all messages sent to them until the messages are consumed or until the messages expire.

- Each message has only one consumer.

- A sender and a receiver of a message have no timing dependencies. The receiver can fetch the message whether or not it was running when the client sent the message.

- The receiver acknowledges the successful processing of a message.

- Use PTP messaging when every message you send must be processed successfully by one consumer.

Publish/Subscribe (pub/sub)

- Application, clients address messages to a topic. Publishers and subscribers are generally anonymous and may dynamically publish or subscribe to the content hierarchy. The system takes care of distributing the messages arriving from a topic's multiple publishers to its multiple subscribers. Topics retain messages only as long as it takes to distribute them to current subscribers.

- Each message may have multiple consumers.

- Publishers and subscribers have a timing dependency. A client that subscribes to a topic can consume only messages published after the client has created a subscription, and the subscriber must continue to be active in order for it to consume messages.

- The JMS API relaxes this timing dependency to some extent by allowing clients to create durable subscriptions. Durable subscriptions can receive messages sent while the subscribers are not active. Durable subscriptions provide the flexibility and reliability of queues but still allow clients to send messages to many recipients.

- Use pub/sub messaging when each message can be processed by zero, one, or many consumers.

- The Publish subscribe model of messaging is a one to many messaging paradigm where the publisher typically sends a message to a centralized node. The node then broadcasts the message to all the topic subscribers, who have registered their interest in the topic. This allows the publisher and the subscribers to be decoupled. Further if the subscriber is registered as a durable subscriber, he will receive the message ultimately even if he is currently inactive. The MOM provider can thus guarantee delivery and any Quality of Service requirements pertaining to the message delivery.

- Subscribers may register interest in receiving messages on a particular message topic. In this model, neither the publisher nor the subscriber knows about each other. A good analogy for this is an anonymous bulletin board.

- Use subject-based adressin and provide location independence for publishers.

JMS Asynchronous Architectures:

- Is more scalable, increase performance, maker beter use of bandwith, desacoples sender and recerivers (...than synchronous).

- When capacities are exeeded, information is no lost, just delayed.

- Is more effective by failures at the hardware, software and network.

- asynchronous messages is better suited with smaller message sizes.

- is more appropriated for "hight volume" transacions processing.

Legacy System

- Screen Scraping, Java-IDL, JCA and JNI are used to integrated java base system to legacy solutions.

Screen Scraping

- A screen scraper emulates a mainframe terminal. Basically the screen scraper logs on to the mainframe like a normal user and sends requests to the mainframe and then reads the response. The problem with a screen scraper is that if you change any of the mainframes code there is always the possibility that the screen scraper will stop working.

- Best alternative only if the existing UI is very tightly coupled with the business tier of the legacy application.

Corba (Common Object Request Broker Architecture)

- is a standard defined by the Object Management Group (OMG) that enables software components written in multiple computer languages and running on multiple computers to work together (i.e., it supports multiple platforms).

- enables separate pieces of software written in different languages and running on different computers to work with each other like a single application or set of services. More specifically, CORBA is a mechanism in software for normalizing the method-call semantics between application objects residing either in the same address space (application) or remote address space (same host, or remote host on a network)

- use ORBs and IIOP to comunication.

Interface Definition Language

- interfaces between CORBA objects can be specified using IDL.

- IDL is mapped to other programing languages.

IIOP (Internet Inter-ORB Protocol)

-Is the abstract protocol by which object request brokers (ORBs) communicate. Standards associated with the protocol are maintained by OMG.

Java IDL

-Adds CORBA capability to the Java platform, providing standards-based interoperability and connectivity.

-Enables distributed Web-enabled Java applications to transparently invoke operations on remote network services using the industry standard IDL (Object Management Group Interface Definition Language) and IIOP (Internet Inter-ORB Protocol) defined by the Object Management Group. Runtime components include Java ORB for distributed computing using IIOP communication.

RMI (Remote Method Invocation)

- Is a Java application programming interface that performs the object-oriented equivalent of remote procedure calls (RPC).

- Depends on Java Virtual Machine (JVM) class representation mechanisms and it thus only supports making calls from one JVM to another.

- The protocol underlying this Java-only implementation is known as Java Remote Method Protocol (JRMP).

- Use stubs e and skeletons to comunication.

RMI-IIOP

- denotes the Java Remote Method Invocation (RMI) interface over the Internet Inter-Orb Protocol (IIOP), which delivers Common Object Request Broker Architecture (CORBA) distributed computing capabilities to the Java 2 platform. It is based on two specifications: the Java Language Mapping to OMG IDL, and CORBA/IIOP 2.3.1

- in order to support code running in a non-JVM context, a CORBA version was later developed. JRMP, whereas the term RMI-IIOP (read: RMI over IIOP) denotes the RMI interface delegating most of the functionality to the supporting CORBA implementation.

- is the protocol used with EJB.

- not support Stub downloads and distributed garbage collection.

Tips:

- Do not suggest a change in architecture (add or exchange of components) of a stable solution without existing needs described.

- There is no requirement more important than another. The most important thing is to meet the requirements of the client / corporation.

- EJB are best used with large and complex enterprise applications with high deployment and transacional requirements. Applications with small-scale and no transactions requirements not benefit from use ef EJB.

Regras Dentro do Banco stored (procedures and triggers)

- violates tier encapsulation (Is not considered a J2EE best practice).

- reduces database portability.

- Writing all the data access logic in stored procedures can affect performance badly because database server may get bogged down with number of requests.

- No manegeability.

- Nâo tem flebilidade em reuso pq procedures and triggers não podem usar principios OOP.

Clusters

- Consists of a set of loosely connected computers that work together so that in many respects they can be viewed as a single system.

- The components of a cluster are usually connected to each other through fast local area networks, each node running its own instance of an operating system.

- Improve performance and availability over that of a single computer, while typically being much more cost-effective than single computers of comparable speed or availability.

- Benefits are replication, Load Balancing and Fault Tolerance.

Load balancing or Load distribution

- is a computer networking methodology to distribute workload across multiple computers or a computer cluster.

- optimal resource utilization, maximize throughput, minimize response time, and avoid overload.

- increase reliability through redundancy. The load balancing service is usually provided by dedicated software or hardware, such as a multilayer switch or a Domain Name System server.

Round robin DNS

- is a technique of load balancing or fault-tolerance provisioning multiple, redundant Internet Protocol service hosts, e.g., Web servers, FTP servers, by managing the Domain Name System's (DNS) responses to address requests from client computers according to an appropriate statistical model.

- round-Robin load distribution is the process of splitting requests evenly irrespective of the request type (i.e. SSL, JSP, HTML). If you have 3 servers, the first request comes in it goes to the first sever, second request to the second server and the third request to the third server. As the fourth request comes in the process starts again and so this request is forwarded to the first server

Fault-tolerance or graceful degradation

- is the property that enables a system computer-based to continue operating properly in the event of the failure of some of its components.

- is particularly sought-after in high-availability or life-critical systems.

Replication

- Is a mechanism to provide fault tolerance.

Active Replication

- Each replica is identical to the main service.

- Each client request is processed by all the servers and an interceptor is used to block extra responses.

- Is similar in concept to Hot Backups.

- Is the preferable choice when dealing with "real time systems" that require quick response even under the presence of faults.

Passive Replication

- There is only one server called primary that processes client requests.

- After processing a request, the primary server updates the state on the other (backup) servers and sends back the response to the client. If the primary server fails, one of the backup servers takes its place.

- The disadvantage of passive replication compared to active is that in case of failure the response is delayed.

- is similar to that of Warm Backups.

Vertical Scalability (|)

- increasing a system's capacity by adding memory, processors and so on.

- easier to achieve because it involves few changes to the existing system's architecture (no impact)

- does not have any impact on reliability or availability because if the system or component fails, in the absence of redundant systems, availability and reliability would suffer.

- more cheaper than Horizontal scaling.

Horizontal Scalability (-)

- adding more servers (inside clusters) to a system.

- hard(tougher) to achieve because it involves a lot of changes to the existing system's architecture (have impact) to support a multi-server environment.

- more expensive than vertical scaling.

- server clustering benefits are replication, load balancinge and fault tolerance.

- increases performance, availability and reliability and provides fault tolerance capabilities but decrease security when additional security measures need to be taken due to addition.

Web Server HTTP

- Support only HTTP protocol.

- faz load balancer para o container adjacente.

- Default error page displays when the container is out.

- Used to store static resources and remove the container. Best performance.

- Used to make front-end SSL connection.

- Support caching, clustering and load balancing.

Application Server

- Support HTTP, TCP/IP e others.

- Support caching, clustering and load balancing.

- Can be configured to work as web server.

Virtual Private Networks

- A virtual private network (VPN) is a secure network that uses primarily public telecommunication infrastructures, such as the Internet, to provide remote offices or traveling users an access to a central organizational network.

- VPNs typically require remote users of the network to be authenticated, and often secure data with firewall and encryption technologies to prevent disclosure of private information to unauthorized parties.

- A network created between two other networks (these are not located in the same place, geographically). Encryption and Authentication are used in the VPN. Normally the VPN is a network on top of an untrusted network (like the Internet).

Transaction Isolation

- The degree to which concurrent parallel transactions on the same data are allowed to interact is determined by the level of transaction isolation configured. ANSI/SQL defines four levels of database transaction isolation as shown in Table 1-21. Each offers a trade-off between performance and resistance from the following unwanted actions:

- Dirty read: a transaction reads uncommitted data written by a concurrent transaction.

- Nonrepeatable read: a transaction rereads data and finds it has been modified by some other transaction that was committed after the initial read operation.

- Phantom read: a transaction re executes a query and the returned data has changed due to some other transaction that was committed after the initial read operation.

Transaction Isolation

- Read Uncommitted - Dirty Read(Yes) Nonrepeatable(Yes) Read Phantom Read(Yes)

- Read Committed - Dirty Read(No) Nonrepeatable(Yes) Read Phantom Read(Yes)

- Repeatable Read - Dirty Read(Yes) Nonrepeatable(No) Read Phantom Read(Yes)

- Serializable - Dirty Read(No) Nonrepeatable(No) Read Phantom Read(No)

I18N and I10L

- To identify the default character enconding for file operations, JDBC querys etc by checking the system property named file.properties as System.out.println(System.getProperty("file.enconding"));

- Java I18N not support calendar and planetary variants.

- Java recognizes fives font names standard - Serif, San Serif, Monospaced, Dialog and DialogInput.

- Java recognizes font styles standard - plain, bold, italic and bolditalic. Mapping is handling by font.properties in JDK/lib/fonts/

- NumberFormat and DecimalFormat classes handles default locates in use decimal symbols (333,33) parse() and format().

Unicode

- Is a computing industry standard for the consistent encoding, representation and handling of text expressed in most of the world's writing systems.

- Provides a standard enconding for the chacacter sets of different languages.

- Provide unique number for every character.

- Independent of plataform, program or language.

UTF

- Stands for Unicode Transformation Formats.

- UTF-16 represents every character using two-bytes.

- UTF-8 represents every character using one-bytes for ASCII characteres.

- UTF-8 has become the dominant character encoding for the WWW.

Security

- Adding new security measures typically affect the application performance negatively. Security techniques such as encryption/decryption, SSL, authentication and authorization checks make our application execution slower than if we don't have all these security protections.

- You have a JAR file that has been signed by a third-party vendor. A Trusted Certificate Authority (CA) has signed the third-party vendor's certificate. It is possible to add more classes to the JAR file.

JSSE (Java Secure Socket Extension)

- enables secure Internet communications. It provides a framework and an implementation for a Java version of the SSL and TLS protocols and includes functionality for data encryption, server authentication, message integrity, and optional client authentication. Using JSSE, developers can provide for the secure passage of data between a client and a server running any application protocol, such as Hypertext Transfer Protocol (HTTP), Telnet, or FTP, over TCP/IP.

JCE (Java Cryptography Extension)

- provides a framework and implementations for encryption, key generation and key agreement, and Message Authentication Code (MAC) algorithms. Support for encryption includes symmetric, asymmetric, block and stream ciphers.

DMZ

- Demilitarized Zone is a physical or logical subnetwork that contains and exposes an organization's external services to a larger untrusted network, usually the Internet - HTTP, FTP, POP3, STMP

- The purpose is to add an additional layer of security to an organization's local area network (LAN); an external attacker only has access to equipment in the DMZ, rather than any other part of the network.

- Is the zone between two firewall.

Encryption Algorithms

Symmetric key algorithms

- Algorithms for cryptography that use the same cryptographic keys for both encryption of plaintext and decryption of ciphertext.

- The keys represent a shared secret between two or more parties that can be used to maintain a private information. This requirement that both parties have access to the secret key is one of the main drawbacks of symmetric key encryption, in comparison to public-key encryption.

Asymmetric key cryptography or Public-key cryptography

- Refers to a cryptographic system requiring two separate keys, one to lock or encrypt the plaintext, and one to unlock or decrypt the cyphertext. Neither key will do both functions. One of these keys is published or public and the other is kept private.

Assymmetric encryption and One-way hash/encrypton

MD5 e SHA-1 e 2

Symmetric encryption.

- Blowfish

- 3DS

- RC4

- RSA

Phishing

- Is a way of attempting to acquire information such as usernames, passwords, and credit card details by masquerading as a trustworthy entity in an electronic communication. Communications purporting to be from popular social web sites, auction sites, online payment processors or IT administrators are commonly used to lure the unsuspecting public.

- Typically carried out by e-mail spoofing or instant messaging, and it often directs users to enter details at a fake website whose look and feel are almost identical to the legitimate one. Phishing is an example of social engineering techniques used to deceive users, and exploits the poor usability of current web security technologies. Attempts to deal with the growing number of reported phishing incidents include legislation, user training, public awareness, and technical security measures.

Denial-of-service

- DoS attack is an attempt to make a machine or network resource unavailable to its intended users.Generally consists of the efforts of one or more people to temporarily or indefinitely interrupt or suspend services of a host connected to the Internet.

Cross Site Scripting (XSS)

- occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user.

SQL Injection

- Is a technique that enables an attacker to perform unauthorized SQL queries in web applications using dynamic SQL statements. Using SQL Injection, attackers might be able to retrieve data you didn't intend for the user to see.

- Measures to prevent it are: escape user input, validate user input, use Java PreparedStatements than Statements.

A Man-in-the-Middle (MitM)

- Is a technique where an attack intercepts another user's session, inspects its contents and tries to modify its data or otherwise use it for malicious purposes. Measures to prevent these attachs are to use encryption of sensitive data and prevent the data being read. Some examples are using SSL, avoiding Frames/IFrames, avoid URL rewriting (SessionId is exposed).

Applets

- Applets that are loaded from the local file system user's CLASSPATH have none of the restrictions that applets loaded over the network do.

- The security manager does not monitor the memory, CPU or network bandwidth usage of an applet.

- when running applets from the command line a different security manager will be used (assuming a typically configured JRE) than when an applet is invoked in a web browser

Remote Unsigned Applets

- can make network connections to the host they came from.

- can send email to the host they came from.

- can utilize only its own code.

- can display HTML documents using the showDocument method of the java.applet.AppletContext class.

- can invoke public methods of other applets on the same page.

- can read all "system properties" unless java.home, java.class.path, user.name, user.home and user.dir

- can not do anything else..

Remote Unsigned Applets VIA JNLP

- can open, read, and save files on the client.

- can access the shared system-wide clipboard.

- can access printing functions.

- can store data on the client, decide how applets should be downloaded and cached, and much more.

Remote Signed Applets

- Do not have the security restrictions that are imposed on unsigned applets and can run outside the security sandbox.

Digital Certificate

- the code has been signed using a digital certificate and packaged in a jar file. Which of the following describes what you definitely know about the third-party code?

- All you definitely know is that the code has been signed using the third-party vendor's private key. You do not know whether it was actually signed by the third-party vendor or an impersonator who stole the private key. So the jar file contents may contain malicious code.

JAAS (The Java Authentication and Authorization Service)

- implements a Java version of the standard Pluggable Authentication Module (PAM) framework.

- can be used for two purposes:

1.for authentication of users, to reliably and securely determine who is currently executing Java code, regardless of whether the code is running as an application, an applet, a bean, or a servlet, ejb , whetever.

2.for authorization of users to ensure they have the access control rights (permissions) required to do the actions performed.

In Servlet and JSP

- HttpServletRequest.getUserPrincipal() returns Principal object. Principal.getName() returns role name.

- HttpServletRequest.isUserInRole("Role_Name") return boolean to determine if callers is within yeh specified role name

In EJB

- javax.ejb.EJBContext.getCallerPrincipal() returns Principal object. Principal.getName() returns role name.

- javax.ejb.EJBContext.isCallerInRole("Role_Name") return boolean to determine if callers is within yeh specified role name

Annotations for Controlling Security Permissions of web componentes and EJBs:

- Use @DeclareRoles to define which security roles to check permissions against. If no @DeclareRoles is present, the list is built automatically from the @RolesAllowed annotation.

- Use @RolesAllowed to list which roles are allowed to access a method or methods. Use @PermitAll or @DenyAll to either permit or deny all roles from using a method or methods.

- Use @RunAs to specify a role a method will always be run as.

- Security roles can be defined by both mechanisms annotations & Deployment descriptor but settings in deployment descriptor override annotations.

GOF Patterns:

Criational

Virtual Factory Method or Constructor - its intention is to define a point of creation for an object type in which there is no knowledge of the concrete type to use.

- Used jaxax.ejb.EJBHome, jaxax.ejb.EJBLocalHome, jaxax.jms.QueueConnectionFactory, jaxax.jms.TopicConnectionFactory, java.text.Collator, java.net.ContentHandlerFactory, javax.naming.spi.InicialContextFactory, javax.net. Socket, java.sql.DriverManager.

Abstract Factory or kit - its intention is to define a point of creation for a family of related or dependent objects in which one has no knowledge of specific types of all these objects to be used. The Abstract Factory is basically a "massive extension" of the Factory Method pattern.

- Used java.awt.Toolkit, DAO, and Transfer Object Assembler

Prototype - its intention is to define a point of creating objects that need to be instantiated based on copying using another object such referred to as "prototype". All objects are created to exist containing values ??copied this supposed prototype object.

- Used java.lang.Object.

Singleton - its intention is to define a point of creating objects that need to be instantiated only once throughout the execution cycle of the solution and provide a single point of access for this reference.

- Used java.lang.Runtime.

Builder - its intention is to define a point of creating a complex object (compound) separated from its representation so that this point of construction can be parameterized to construct different variations of the same object.

Strutctural

Adapter or Wrapper - its intention is to convert the interface of one class to another required interface, defining an intermediate point of connection with the objective of promoting communication between two incompatible interfaces.

Used - JCA, java.awt.event.ComponentAdapter

Bridge or Handle / Body - its intention is to separate the abstraction of an action of its different implementations, so that both can be flexibly interchangeable.

Composite - its intention is to make an object can generally be operated in a manner that represents a dynamic hierarchical compositions of objects.

Wrapper or Decorator - its intention is to define an alternative means of adding the heritage responsibilities to an object in a flexible and dynamic run-time.

- Used javax.ejb.EJBObject, java.io.BufferedReader and java.awt. *

Facade - its intention is to define a user-friendly interface, simple and unified to one or more sets of operations related to internal subsystems.

- Used java.net.URL.

Flyweight - its intention is to define an efficient point of sharing that supports a large number of manipulations of fine-grained reusable objects.

- Used java.lang.String Pooling and Resouces (EBJ Stateless).

Proxy or Surrogate - its intention is to define an object-an object-substitute for real, so the object-replacement can transparently mediate the interactions for the object-real.

- Used javax.ejb.EJBObject rmi stub and proxy.

Behavior

Chain of Responsibility - its intention is to decouple the object-sender of a request of the alleged objects-receptors, allowing multiple objects that can be dynamically-receivers lined up to handle the request transparently.

Used - Servlet / JSP Request Dispatcher.

Command or Action or Transaction - its intention is to make a request for a command may be encapsulated as a uniform object, allowing objects-that clients can be parameterized flexibly with different objects, commands interchangeable.

Used - JEE MessageDrivenBean invoke business logic based on content of messages and Servlet / JSP are invoked based on type of HTTP request (put, get, post, etc.)

Interpreter - its intention is to define objects that can represent and interpret a given interchangeably any textual grammar.

Iterator - its intention is to define a way that the aggregated objects within an object can be accessed by other objects sequentially without exposing the internals of their relationships and implementations.

- Used java.util.Iterator, java.util.Enumeration.

Mediator - its intention is to define an object used to encapsulate the relationship between two other objects, causing the particular relationship of these two target objects can be implemented indirectly, flexible and intercambial.

Memento - its intention is to define a way to capture and store the state of an object so that this object can be restored to its original state before.

- Used EntityBeans using Bean-Managed Persistence (BMP).

Observer - its intention is to define a form that enables an object and automatically update aggregator to notify other objects dependent aggregates when a change occurs in the state in the object aggregator.

- Used JMS Publish / Subscribe Model, and java.lang.Obsersable java.lang.Observer.

State - its intention is to define a form that enables an object interchangeably vary their behavior when internal changes happen in your state.

Strategy - its intention is to define an object that has certain behaviors that suffer variations interchangeable depending on other objects that manipulate customers.

Template Method - its intention is to define objects that implement an algorithm previously formatted within a behavior, so that other objects can interchangeably replace portions of the features of this algorithm.

Visitor - its intention is to define a form that enables various operations that can be dynamically applied to an object, without changing its structure definition.

J2EE Patterns:

Presentantion Tier

Intercepting Filter - facilitates preprocessing and post-processing of a request.

Front Controller - provides a centralized controller for managing the handling of requests.

Composite View - creates an aggregate View from atomic subcomponents (Facelets, Struts Tiles).

View Helper - encapsulates logic that is not related to presentation formatting into Helper components.

Dispatcher View - combines a Dispatcher component with the Front Controller and View Helper patterns, deferring many activities to View processing.

Service to Worker - combines a Dispatcher component with the Front Controller and View Helper patterns.

Business Tier

Business Delegate - reduces coupling between presentation-tier clients and business services. It hides the underlying implementation details of the business service, such as lookup and access details of the EJB architecture.

Service Locator - multiple clients can reuse the Service Locator object to reduce code complexity, provide a single point of control, and improve performance by providing a caching facility.

Session Facade - encapsulate the complexity of interactions between the business objects participating in a workflow. The Session Facade manages the business objects, and provides a uniform coarse-grained service access layer to clients.

Transfer Object - used to send and retrieve datas inside a single method call. When the client requests the ejb for the business data, the ejb can construct the Transfer Object, populate it with its attribute values, and pass it by value to the client.

Transfer Object Assembler - uses Transfer Objects to retrieve data from various business objects and other objects that define the model or part of the model.

Composite Entity - it model, represent, and manage a set of interrelated persistent objects rather than representing them as individual fine-grained entity beans. A Composite Entity bean represents a graph of objects.

Value List Handler - suggests an alternate approach of using ejb-finder methods, controling the search, cache the results and provide the results to the client using a lightweight mechanism. Critical concern in a distributed paradigm is the latency time.

Integration Tier

Data Access Object - abstracts and encapsulate all access to the data source. The DAO manages the connection with the data source to obtain and store data.

Domain Store - provides a powerful mechanism to implement transparent persistence for your object model. It combines and links several other patterns including Data Access Objects. Patter using in ORM frameworks.

Web Service Broker - exposes aplication services using XML and web protocols.

Service Activator - enables asynchronous access to enterprise beans and other business services. It receive asynchronous client requests and messages. On receiving a message, the Service Activator locates and invokes the necessary business methods on the business service components to fulfill the request asynchronously. In EJB2.0, Message Driven beans can be used to implement Service Activator for message based enterprise applications. The Service Activator is a JMS Listener and delegation service that creates a message façade for the EJBs.

Another Patterns:

Application Service/Controller

- improves reusability of business logic and centralizes reusable business and workflow logic. Like a Facade [GOF] patters applying in separation of layers.

- centralize and modularize action and view management.

The MVC architecture has the following benefits:

- multiple views using the same model: the separation of model and view allows multiple views to use the same enterprise model.

- support for new types of clients: you simply write a view and controller for it and wire them into the existing enterprise model.

- efficient modularity of the design: changes to one aspect of the program aren't coupled to other aspects.

- ease of growth: controllers and views can grow as the model grows; and older versions of the views and controllers can still be used as long as a common interface is maintained.

Design Patterns Tips:

- Use of patterns let code be more maintainable and make faster software development cause reuse already solutions.

- Use of patterns have no impact of performance, size of the program final code or loose coupling code.

Layers Approach:

- Developers of each unit can work on implementations parallely.

- Upgrades on individual unit will not lead to recompilation of code in other layers.

- Eases testing as developers of a unit can develop proxies/stubs with hard-coded data for their testing.

1. Software Architecture and Design: The process of modeling a business such that all of its functional and non-functional service level

requirements are adequately addressed.

The Non-functional Quality of Service (QoS) requirements are:

1. Performance – A measure of the system in terms of response time or number of transactions per unit time. Load Distribution (e.g.

DNS Round Robin) and Load Balancing are two techniques that aid in higher performance. Other development and deployment related

tasks such as Application Tuning, Server Tuning, and Database Tuning also help the system perform better.

DNS Round Robin: A process for distributing load in a system. If we have ten web servers that can service HTTP requests, the first

request is directed to server 1, the second to server 2 and so on. When all ten servers have serviced one request each, the process

starts all over again. Note that this is only a load distribution technique. DNS Round Robin does not balance the load.

Reverse proxy load balancing: Reverse proxy load balancing is generally used when you have servers with different amounts of CPUs

and Memory. You might have some really powerful servers just to be used for SSL sessions and others to handle static html. Using

this will maximize the performance of your application

2. Scalability – The ability of a system to perform and behave in a satisfactory manner with increases in load.

Scalability can be achieved in two ways – Vertical (adding additional processors, memory or disks to existing hardware) and

Horizontal (adding more machines to the system.)

Vertical scalability is easier to implement than Horizontal scalability. Many J2EE vendors do however support Horizontal scaling as

well.

3. Reliability – The ability of a system to assure the integrity and consistency of the application and all its data as the load increases.

4. Availability – The ability of a system to assure that all services and resources are always accessible. This can be achieved through

fault tolerance (the ability to prevent system failures in the event of service(s) / component(s) failures, commonly implemented via

redundancy) techniques such as Active and Passive Replication.

Active Replication: This is comparable to 'hot backups.' All redundant machines / processes / servers / components are constantly

updated. All replicas handle all requests. An interceptor mediates by sending only one response.

Passive Replication: This is similar to ‘warm backups.’ The primary server handles all requests. It also synchronizes its state with the

secondary replicas. Should a fail over occur, a secondary replica takes over.

5. Extensibility - The ability to easily add new functionality to the existing system. This can be achieved by using best practices and

well-defined architecture and design techniques.

6. Maintainability - Ability to easily correct flaws in the existing system.

7. Security - The ability to protect a system and all its components and services against potential attacks. Security attacks generally

try to compromise confidentiality and integrity of the system. Sometimes they also take the form of ‘Denial of Service’ (DoS) attacks

that bring down a system by flooding it with messages. Security can be addressed by the use of technologies (firewalls, DMZ, data

encryption, Digital Certificates and so on) and methodologies (good security policies and procedures.)

8. Manageability - The ability to monitor and perform preventive maintenance on a system.

2. Common Architectures

1. 1-Tier Architecture: Legacy systems such as those based on Mainframe technology. Dumb terminal clients directly connecting to the

mainframe characterize these systems.

1-Tier systems are not very scalable or extensible. They typically involve single points of failure. They are also difficult to maintain.

1-Tier systems may be easy to manage because of the limited distribution of components and services.

2. 2-Tier Architecture: Client / Server based models where both clients and servers are intelligent machines and the workload is

distributed among them.

Typically clients maintain individual connections to the server and are responsible for data presentation and the processing of

validations / business logic.

Servers are generally responsible for managing data but may perform some business logic via database stored procedures and

triggers.

In the strictest sense, 2-Tier models may not always represent a single point of failure, though that is most often the case. For

example, if servers in a server farm of databases hold 1 million accounts each, the failure of one of these servers need not

constitute a system failure. However if the client uses a catalog server, which re-routes requests appropriately via a Database Link,

the failure of the catalog server would constitute a total system failure.

2-Tier systems are also not very extensible, maintainable, manageable, or secure. Any client changes involve deployment to all

client workstations. Since client machines reside at various locations, they also constitute a management nightmare. Most clients

also use initialization files that may contain sensitive information such as user id, password and so on.

3. N-Tier Architecture: Distributed architecture involving multiple tiers, each performing a specialized function.

J2EE based architectures are good examples. Typically made of:

a. Client Tier: The tier with which the end user interacts. Clients can be ‘thin clients’ as in the case of Browser based applications

or fat clients as in the case of client Java applications.

b. Web Tier: Decouples the client tier from the Business tier. Java Servlets and JSPs reside in this tier. Servlets act as Controllers,

they translate incoming requests, and dispatch them to components that can invoke the necessary business events in the

Business Tier. JSP combine static templates with dynamic data to create dynamic output that the client tier uses for

presentation to the user.

c. Business / Application Logic Tier: Generally implemented using Enterprise Java Beans (EJB) that act as business process

objects and business domain objects. EJB containers provide various services such as Object Distribution, Persistence,

Transaction, Resource Management, Security and so on.

d. Enterprise Information System Integration Tier: The EIS Integration tier interfaces between the Business (and sometimes

Web tier) objects and Enterprise Information Systems. Example, Data Access Objects (DAO) decouple Enterprise beans (typically

Session Beans or BMP Entity Beans) with Enterprise Data.

e. Enterprise Information System Tier: This tier represents all the Enterprise data. This could be in many forms including

Relational Databases, XML Databases, ERP Systems and so on.31/07/12 OCMJEA 5 Exam Simulator - Full Version: Quick Revision Notes

2/9 www.whizlabs.com/examprep/mod/resource/view.php?id=941

N-Tier Systems (if well architected and designed) can help achieve all non-functional service level requirements of the system.

Note: Due to a highly distributed model, manageability may suffer a little in N-Tier systems, but since all systems are highly modular,

the issue is generally fairly easy to address. It is also notable that Architects have to sometimes compromise a little on one service

level requirement, to achieve the desired effect on another. For example Performance and Security show an inverse proportional

relationship.

II. Legacy Connectivity

Screen Scraper: A screen scraper emulates a mainframe terminal. Basically the screen scraper logs on to the mainframe like a normal user

and sends requests to the mainframe and then reads the response. The problem with a screen scraper is that if you change any of the

mainframes code there is always the possibility that the screen scraper will stop working.

Java Native Interface: JNI is used to allow Java to communicate with programs written in languages like C++. In effect you are wrapping

the C++ code to make it available to Java. For example you will wrap a C++ method called debitAccount (int amount) with a similar Java

method, the Java method will just call the C method. This means you can now make the method accessible via RMI

JAVA IDL: Java IDL adds CORBA (Common Object Request Broker Architecture) capability to the Java platform, providing standards-based

interoperability and connectivity. Java IDL enables distributed Web-enabled Java applications to transparently invoke operations on remote

network services using the industry standard IDL (Object Management Group Interface Definition Language) and IIOP (Internet Inter-ORB

Protocol) defined by the Object Management Group. Runtime components include Java ORB for distributed computing using IIOP

communication.

Go to: http://java.sun.com/j2se/1.3/docs/guide/idl/index.html for more information.

Java Connector Architecture:

The following is taken from:

http://java.sun.com/j2ee/connector/

“The J2EE Connector architecture provides a Java solution to the problem of connectivity between the many application servers and EISs

already in existence. By using the J2EE Connector architecture, EIS vendors no longer need to customize their product for each application

server. Application server vendors who conform to the J2EE Connector architecture do not need to add custom code whenever they want

to add connectivity to a new EIS.”

A transaction attribute must be specified for the methods in the component interface of a session bean and for the methods in the component

and home interfaces of an entity bean.

Required - If the transaction attribute is Required, the container ensures that the enterprise bean's method will always be invoked with a JTA

transaction. If the calling client is associated with a JTA transaction, the enterprise bean method will be invoked in the same transaction context. However, if a client is not associated with a transaction, the container will automatically begin a new transaction and try to commit

the transaction when the method completes.

RequiresNew - If the transaction attribute is RequiresNew, the container always creates a new transaction before invoking the enterprise bean

method and commits the transaction when the method returns. If the calling client is associated with a transaction context, the container

suspends the association of the transaction context with the current thread before starting the new transaction. When the method and the

transaction complete, the container resumes the suspended transaction.

NotSupported - If the transaction attribute is NotSupported, the transactional context of the calling client is not propagated to the enterprise

bean. If a client calls with a transaction context, the container suspends the client's transaction association before invoking the enterprise

bean's method. After the method completes, the container resumes the suspended transaction association.

Supports - It the transaction attribute is Supports and the client is associated with a transaction context, the context is propagated to the

enterprise bean method, similar to the way the container treats the Required case. If the client call is not associated with any transaction

context, the container behaves similarly to the NotSupported case. The transaction context is not propagated to the enterprise bean method.

Mandatory - The transaction attribute Mandatory requires the container to invoke a bean's method in a client's transaction context. If the

client is not associated with a transaction context when calling this method, the container throws

javax.transaction.TransactionRequiredException if the client is a remote client or javax.ejb.TransactionRequiredLocalException if the client is a

local client. If the calling client has a transaction context, the case is treated as Required by the container.

Never - The transaction attribute Never requires that the enterprise bean method explicitly not be called within a transaction context. If the client calls with a transaction context, the container throws java.rmi.RemoteException if the client is a remote client or javax.ejb.EJBException

if the client is a local client. If the client is not associated with any transaction context, the container invokes the method without initiating a

transaction."

IV. EJB Container Model

The EJB Container sits between an EJB Server and EJBs. An EJB Server can have one or more EJB Containers and each container can manage

one or more components.

The EJB Container manages the EJBHome and EJBObject implementations. Via these objects, the container decorates Enterprise Bean classes

and provides various services such as:

Life cycle management

Naming

Object Distribution

Persistence

Security

Transactions and

Concurrency

The EJB Container also does resource management through Instance Pooling and swapping (in the case of Stateless Session Beans) and

Passivation / Activation, in the case of Stateful Session Beans and Entity Beans.

Note: Even while the Bean is passivated, the client's connection to the EJBObject is maintained. Before Passivation, all non-transient non-

serializable fields must be set to NULL.

BMT/CMT

The following is taken from:

http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Transaction3.html

In an enterprise bean with container-managed transactions, the EJB container sets the boundaries of the transactions. You can use container-

managed transactions with any type of enterprise bean: session, entity, or message-driven. Container-managed transactions simplify development because the enterprise bean code does not explicitly mark the transaction's boundaries. The code does not include statements

that begin and end the transaction.

http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Transaction4.html

In a bean-managed transaction, the code in the session or message-driven bean explicitly marks the boundaries of the transaction. An entity

bean cannot have bean-managed transactions; it must use container-managed transactions instead. Although beans with container-managed

transactions require less coding, they have one limitation: When a method is executing, it can be associated with either a single transaction or

no transaction at all. If this limitation will make coding your bean difficult, you should consider using bean-managed transactions.

V. Protocols

Port numbers for some basic protocols:

The following is a list of Protocols and their default Port numbers:

· HTTP – 80

· HTTPS - 443

· FTP - 21

· JRMP - 1099

· IIOP - 535

· Telnet - 23

VI. Applicability of J2EE

Deciding between Java IDL RMI-JRMP and RMI-IIOP:

Java IDL - If you have been developing CORBA applications using IDL for some time, you will probably want to stay in this environment. Create

the interfaces using IDL, and define the client and server applications using the Java programming language to take advantage of its "Write

Once, Run Anywhere™" portability, its highly productive implementation environment, and its very robust platform.

RMI-JRMP - If all of your applications are written in the Java programming language, you will probably want to use

Java RMI to enable communication between Java objects on different virtual machines and different physical

machines. Using Java RMI without its IIOP option leverages its strengths of code portability, security, and garbage

collection.

RMI-IIOP - If you are writing most of your new applications using the Java programming language, but need to maintain legacy applications

written in other programming languages as well, you will probably want to use Java RMI with its IIOP compiler option.

When to use Enterprise Javabeans?

It is recommended that you use Enterprise Javabeans if Transactions are involved in the application. See below for more details.

The application must be scalable. To accommodate a growing number of users, you may need to distribute an application's components across

multiple machines. Not only can the enterprise beans of an application run on different machines, but also their location will remain transparent

to the clients.

Transactions are required to ensure data integrity. Enterprise beans support transactions, the mechanisms that manage the concurrent access

of shared objects. The application will have a variety of clients. With just a few lines of code, remote clients can easily locate enterprise beans. These clients can

be thin, various, and numerous.

VIII. Messaging

Messaging: Enterprise Messaging allows two or more applications to exchange information in the form of messages.

Middleware Architectures:

a. Synchronous, tightly coupled communication between distributed components: This is the model of CORBA, RMI, EJB and so on. The

programming model is called Remote Procedure Call (RPC).

Note: EJB 2.0 has a new kind of bean called a Message Driven Bean, which acts as a message listener for processing asynchronous

requests.

b. Asynchronous, loosely coupled communication between components: This is the Message Oriented Middleware or MOM model. The programming

model is called Messaging.

IX. Internationalization

1. Internationalization: Adapting a program for use in any country is called Internationalization.

2. Localization: The process of adapting a program for use in a particular country is referred to as Localization. During Localization the

language of the text, message icons, colors used, dialogs, number formats, time representation and even sorting algorithms are subject to change.

List of items that may be subject to Internationalization:

· Language for Messages

· Formats – Numeric, Date and so on

· Dictionary sort order

· Currency symbol and position

· Tax and other legal rules

· Cultural preferences

Java support for Internationalization:

Properties

Locale

Resource Bundle

Unicode

Java.text Package

InputStreamReader

OutputStreamWriter

Locale:

The following is taken from:

http://java.sun.com/j2se/1.3/docs/api/java/util/Locale.html

A Locale object represents a specific geographical, political, or cultural region. An operation that requires a Locale to perform its task is called

locale-sensitive and uses the Locale to tailor information for the user. For example, displaying a number is a locale-sensitive operation--the

number should be formatted according to the customs/conventions of the user's native country, region, or culture.

The Properties class represents a persistent set of properties. The Properties can be saved to a stream or loaded from a stream. Each key

and its corresponding value in the property list is a string.

Every Java application has a single instance of class Runtime that allows the application to interface with the environment in which the

application is running. The current runtime can be obtained from the getRuntime() method.

Properties:

The following is taken from:

http://java.sun.com/j2se/1.3/docs/api/java/util/Properties.html

The Properties class represents a persistent set of properties. The Properties can be saved to a stream or loaded from a stream. Each key

and its corresponding value in the property list is a string.

A property list can contain another property list as its "defaults"; this second property list is searched if the property key is not found in the

original property list.

Because Properties inherits from Hashtable, the put and putAll methods can be applied to a Properties object. Their use is strongly

discouraged as they allow the caller to insert entries whose keys or values are not Strings. The setProperty method should be used instead. If

the store or save method is called on a "compromised" Properties object that contains a non-String key or value, the call will fail.

java.text package:

The following is taken from:

http://java.sun.com/j2se/1.3/docs/api/java/text/package-summary.html

Provides classes and interfaces for handling text, dates, numbers, and messages in a manner independent of natural languages. This means

your main application or applet can be written to be language-independent, and it can rely upon separate, dynamically linked localized

resources. This allows the flexibility of adding localizations for new locations at any time.

ResourceBundle:

The following is taken from:

http://developer.java.sun.com/developer/technicalArticles/Intl/IntlIntro/

Resource Bundles - This internationalization feature of the JDK provides a mechanism for separating user

interface (UI) elements and other locale-sensitive data from the application logic in a program. Separating

locale-sensitive elements from other code allows easy translation. It allows you to create a single code base

for an application even though you may provide 30 different language versions. Although you might be

predisposed to think of text only, remember that any localizable element is a resource, including buttons,

icons, and menus.

The JDK uses resource bundles to isolate localizable elements from the rest of the application. The resource bundle contains either the

resource itself or a reference to it. With all resources separated into a bundle, the Java application simply loads the appropriate bundle for the

active locale. If the user switches locales, the application just loads a different bundle.

Resource bundle names have two parts: a base name and a locale suffix. For example, suppose you create a resource bundle named MyBundle.

Imagine that you have translated MyBundle for two different locales, ja_JP and fr_FR. The original MyBundle will be your default bundle; the one

used when others cannot be found, or when no other locale-specific bundles exist. However, in addition to the default bundle, you'll create two

more bundles. In the example these bundles would be named MyBundle_ja_JP and MyBundle_fr_FR. The ResourceBundle.getBundle method relies

on this naming convention to search for the bundle used for the active locale.

The java.util.ResourceBundle class is abstract, which means you must use a subclass of ResourceBundle. The JDK provides two subclasses:

PropertyResourceBundle and ListResourceBundle. If these don't meet your needs, you can create your own subclass of ResourceBundle.”

X. Security

1. Security Basics: Identification refers to identifying who someone is. Authentication refers to verifying that a person / application is

indeed who he / she / it is claiming to be. Authorization is verifying if that someone has access to a particular resource. Confidentiality

means that the data has not being read by anyone other than the intended recipient. (i.e. the data is encrypted). Data Integrity means

that the data hasn’t been altered in transit (i.e. the data is sealed).

2. Virtual Private Networks (VPN): A network created between two other networks (these may not be located in the same place,

geographically). VPN authenticates the user and uses Encryption for communication. Normally the VPN is a trusted network on top of an

untrusted network (such as the Internet).

3. Tunneling: Tunneling is used to pass one protocol through a port that it does not, by default run on. For example if the only free port on

the firewall was port 80 and you needed to pass JRMP through the firewall you would “tunnel” JRMP through the firewall. (JRMP by31/07/12 OCMJEA 5 Exam Simulator - Full Version: Quick Revision Notes

9/9 www.whizlabs.com/examprep/mod/resource/view.php?id=941

standard runs on port 1099). Basically JRMP would run on top of HTTP. However Tunneling should generally be avoided and should only be

used as a last resort.

Sorry for language mistakes, english is my second language.

Fernando Franzini - fernandofranzini@gmail.com / http://fernandofranzini.wordpress.com.