Blog 4

on March 08, 2019

Programming Applications and Frameworks

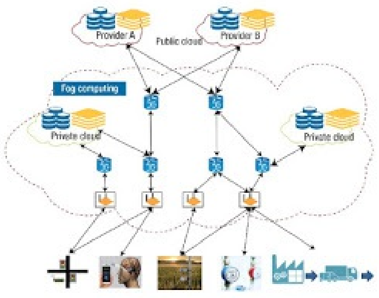
Malith Iddamalgoda | IT17012584

BLOG NO:04

A. Distributed systems

A distributed system is a network that consists of autonomous computers that are connected using a distribution middle ware. They help in sharing different resources and capabilities to provide users with a single and integrated coherent network.

https://www.youtube.com/watch?v=rYK-kTBUrK4



The key features of a distributed system are:

- Components in the system are concurrent. A distributed system allows resource sharing, including software by systems connected to the network at the same time.
- There can be multiple components, but they will generally be autonomous in nature.
- A global clock is not required in a distributed system. The systems can be spread across different geographies.
- Compared to other network models, there is greater fault tolerance in a distributed model.
- Price/performance ratio is much better.

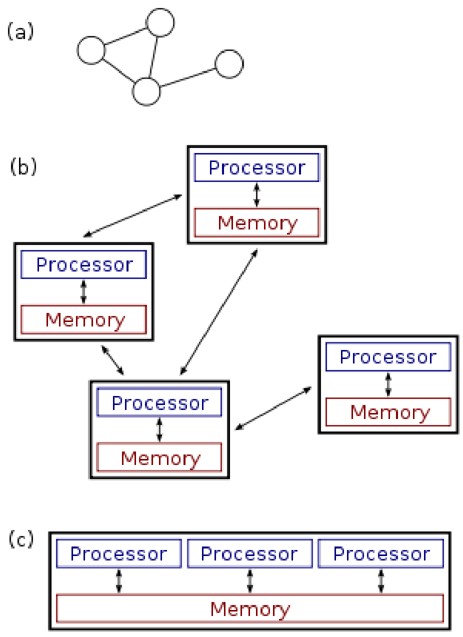
The key goals of a distributed system include:

- Transparency: Achieving the image of a single system image without concealing the details of the location, access, migration, concurrency, failure, relocation, persistence and resources to the users
- Openness: Making the network easier to configure and modify
- Reliability: Compared to a single system, a distributed system should be highly capable of being secure, consistent and have a high capability of masking errors.
- Performance: Compared to other models, distributed models are expected to give a much-wanted boost to performance.
- Scalability: Distributed systems should be scalable with respect to geography, administration or size.

Challenges for distributed systems include:

- Security is a big challenge in a distributed environment, especially when using public networks.
- Fault tolerance could be tough when the distributed model is built based on unreliable components.
- Coordination and resource sharing can be difficult if proper protocols or policies are not in place.
- Process knowledge should be put in place for the administrators and users of the distributed model.

***Distributed computing is a field of computer science that studies distributed systems.**



B. Standalone systems VS Distributed systems

Standalone systems:

All the components are executed within a single device.

Do not need a network.

Usually one or tightly coupled set of technologies are used to develop (JAVA,.NET).

Distributed systems

The components are distributed and executed in multiple devices.

Need a network

Multiple and loosely coupled set of technologies are used to develop(HTML +CSS + JS + PHP)

C. Elements of Distributed systems

•Processing components

•Data networks for components to communicate

•Including the components who are dedicated for processing the communication, called connectors

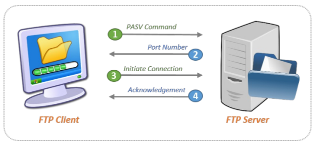
•Data stores (data bases) and Data

•The configuration of the above elements

D. Types of services, which can be gained from Distributed systems

•Mail service (SMTP, POP3, IMAP)

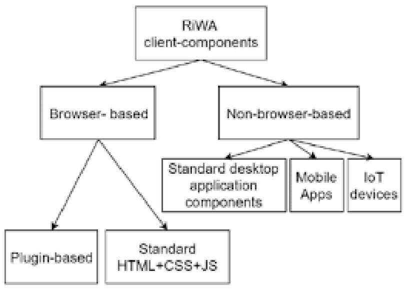
•File transferring and sharing (FTP)

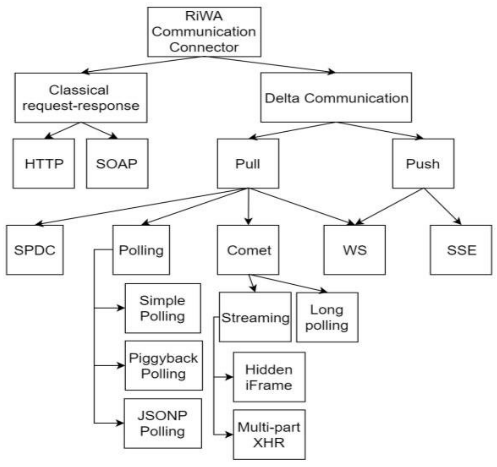


•Remote logging (telnet)

•Games and multimedia (RTP, SIP,H.26x)

•Web (HTTP)





Types of Web-based Systems:

•Web sites•Web applications•Web services and client apps•Rich Internet Applications (RIAs)/Rich Webbased

Applications(RiWAs)

E. Architectures for Distributed systems

Client-server (or two-tier)architecture

What does *Two-Tier Client/Server* mean?

A two-tier client/server is a type of multi-tier computing architecture in which an entire application is distributed as two distinct

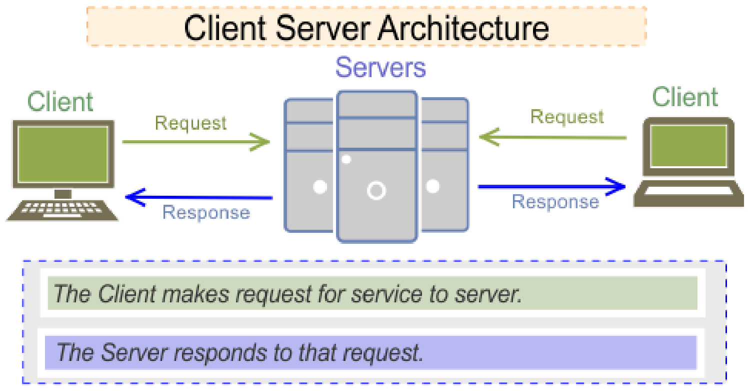
layers or tiers. It divides the application logic, data and processing between client and server devices.

Two-Tier Client/Server

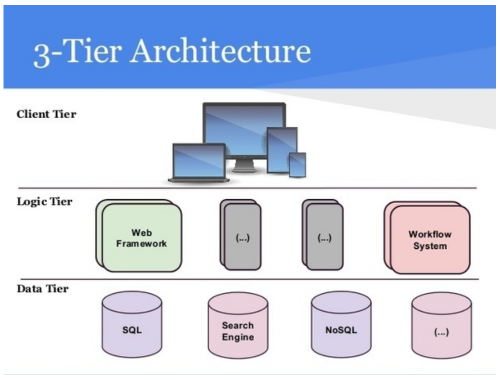
A two-tier client/server works when most or all of the application logic and data is hosted on a server. The client integrates

with the presentation layer and accesses the server for application specific tasks and processing.

The basic architecture of the distributed system is two-tier architecture(client-server).


Client Server Architecture
Servers
Client
Request
Response
Request
Client
Response
The Client makes request for service to server.
The Server responds to that request.

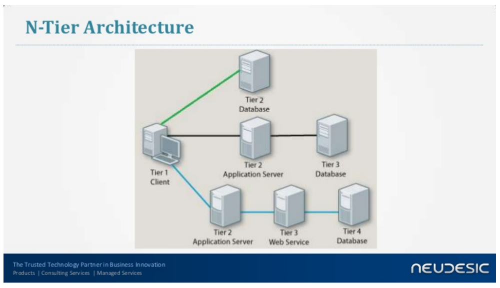
3-tier Architecture:



- A 3-tier architecture is a type of software architecture which is composed of three “tiers” or “layers” of logical computing. They are often used in applications as a specific type of client-server system. 3-tier architectures provide many benefits for production and development environments by modularizing the user interface, business logic, and data storage layers.

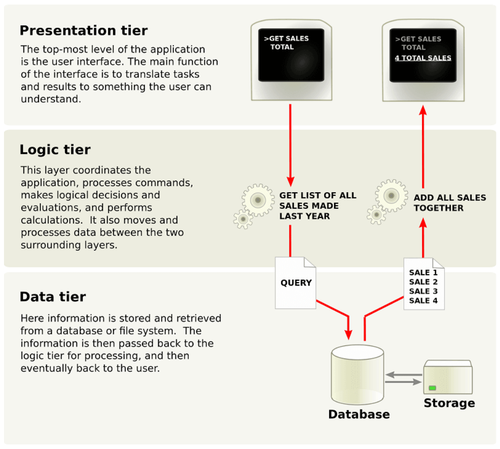
3-tier architecture is used, when there is a need for data persistence and also to separate the application logic from the data.This can be seen as an extension of 2-tier architecture.

n-tier Architecture:


N-Tier Architecture
Tier 2 Database
Tier 1 Client
Tier 2 Application Server
Tier 3 Database
Tier 2 Application Server
Tier 3 Web Service
Tier 4 Database
The Trusted Technology Partner in Business Innovation
Products | Consulting Services | Managed Services
NEUDESIC

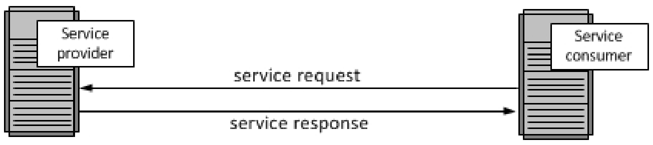
N-tier architecture is also called multi-tier architecture because the software is engineered to have the processing, data management, and presentation functions physically and logically separated. That means that these different functions are hosted on several machines or clusters, ensuring that services are provided without resources being shared and, as such, these services are delivered at top capacity. The “N” in the name n-tier architecture refers to any number from 1.

Not only does your software gain from being able to get services at the best possible rate, but it’s also easier to manage. This is because when you work on one section, the changes you make will not affect the other functions. And if there is a problem, you can easily pinpoint where it originates.


Presentation tier
The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.
>GET SALES TOTAL
>GET SALES TOTAL
4 TOTAL SALES
Logic tier
This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.
GET LIST OF ALL SALES MADE LAST YEAR
ADD ALL SALES TOGETHER
QUERY
SALE 1
SALE 2
SALE 3
SALE 4
Data tier
Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.
Database
Storage

Service Oriented Architecture:

A service-oriented architecture is essentially a collection of services. These services communicate with each other. The communication can involve either simple data passing or it could involve two or more services coordinating some activity. Some means of connecting services to each other is needed.

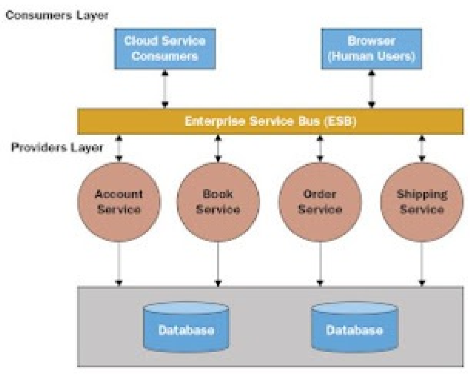


Related Articles

More on the general topic:

Web Services Articles

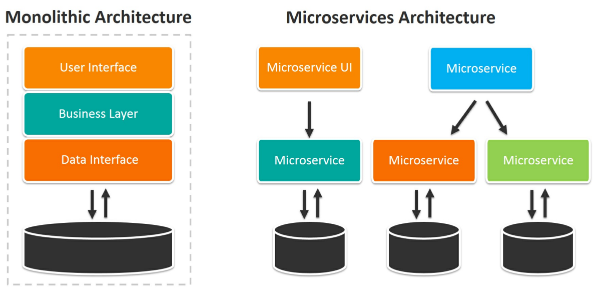
- Web Services Definition
- Web Services Explained
- Application Program Interfaces (APIs)
- Web Services Specifications
- Prior Service-Oriented Architecture Specifications
- Organizations
- Service-Oriented Architecture (SOA) Governance
- Article Suggestions



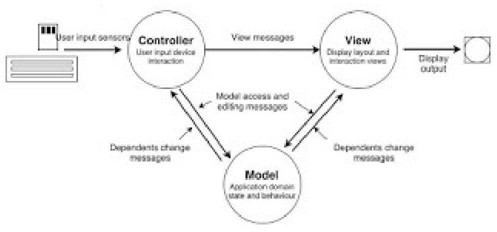
F. Micro-service architecture VS Monolithic architecture

Monolithic vs micro-services

Micro services Architecture. The idea is to split your application into a set of smaller, interconnected services instead of building a single monolithic application. Each micro service is a small application that has its own hexagonal architecture consisting of business logic along with various adapters.



J. MVC style (Model-View-Controller)


User input sensors
Controller
User input device interaction
View messages
View
Display layout and interaction views
Display output
Model access and editing messages
Dependents change messages
Dependents change messages
Model
Application domain state and behaviour

Stands for "Model-View-Controller." MVC is an application design model comprised of three interconnected parts. They include the model (data), the view (user interface), and the controller (processes that handle input).

The MVC model or "pattern" is commonly used for developing modern user interfaces. It is provides the fundamental pieces for designing a programs for desktop or mobile, as well as web applications. It works well with object-oriented programming, since the different models, views, and controllers can be treated as objects and reused within an application.

Below is a description of each aspect of MVC:

1. Model

A model is data used by a program. This may be a database, file, or a simple object, such as an icon or a character in a video game.

2. View

A view is the means of displaying objects within an application. Examples include displaying a window or buttons or text within a window. It includes anything that the user can see.

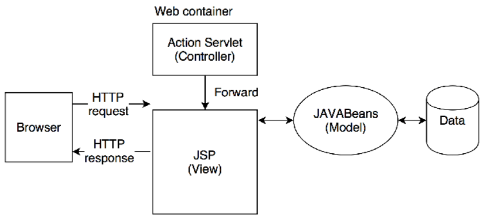
3. Controller

A controller updates both models and views. It accepts input and performs the corresponding update. For example, a controller can update a model by changing the attributes of a character in a video game. It may modify the view by displaying the updated character in the game.

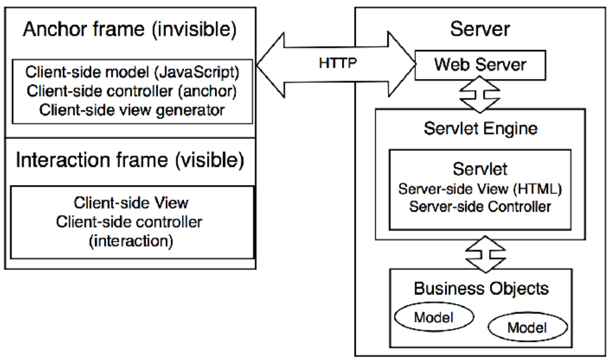
The three parts of MVC are interconnected (see diagram). The view displays the model for the user. The controller accepts user input and updates the model and view accordingly. While MVC is not required in application design, many programming languages and IDEs support the MVC architecture, making it an common choice for developers.

H. MVC for web-based systems

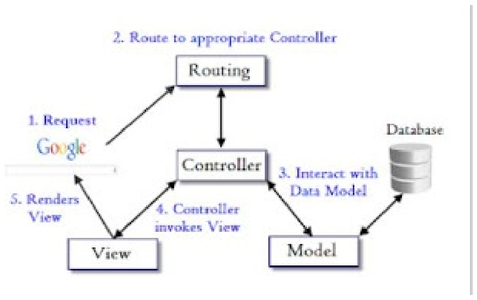
Oracle Model 2 architecture



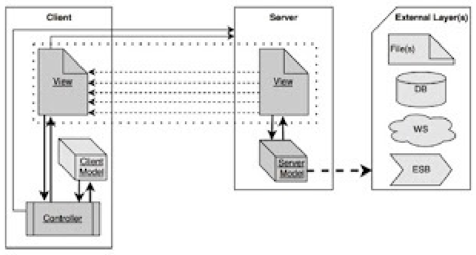
Dual MVC architecture



Model-View-Controller (MVC) –MVC web version


2. Route to appropriate Controller
Routing
1. Request
Google
Database
Controller
3. Interact with Data Model
5. Renders View
4. Controller invokes View
View
Model

Balanced Abstract Web MVC (BAW-MVC) (A distributed mvc version)



Advantages of using MVC architecture:

1. Faster Web Application Development Process:

MVC offers support for rapid and parallel development. So, developing web applications using the MVC model it is possible that one developer work on the view while the another can work on the controller. This helps for easy implementation of the business logic of the web application. It surely benefits developers for completing the web application three times quicker compares with the applications that are developed using other development patterns.

2. MVC Web Application Supports Asynchronous Technique:

.Net developers can integrate MVC architecture with JavaScript Framework. It means that MVC applications can be made to work even with PDF files, site that runs only on the specific browsers, and also for the desktop widgets. MVC also allows using the asynchronous technique, which allows web developers to build faster loading web apps.

3. Offers The Multiple Views:

In the MVC architecture, it is possible to create multiple views for a model. Today, there is a great demand for accessing new ways to access your application and for that MVC development is certainly a great solution. Furthermore, in this method, Code duplication is certainly less as it can separate data and business logic from the display.

4. Ideal for developing large size web application:

It works well for developing web applications which need to be supported by large teams of developers and for Web designers who wants greater control over the application behavior.

5. MVC Model Returns The Data Without The Need of Formatting:

It is helpful for the developers because the same components can be used and called for use with any interface. For example, any types of data can be formatted using HTML, but with MVC framework you can also format using the Macromedia Flash or Dream viewer.

6. The Modification Never Affects The Entire Model:

It is obvious that you make minor changes in a web application such as like changing colors, screen layouts, fonts and adding an extra support for mobile phones or tablets. Furthermore, adding a new type of views is very easy in MVC pattern as Model part does not depend on the views part. So, any changes in the Model will never affect the entire architecture.

Thus, today there are many enterprises which opting for the development of web applications based on MVC architecture to take the above-given advantages. Today you need to find certified MVC .net developer which satisfies your web application development requirement.

Disadvantages of MVC:

1. Increased complexity2. Inefficiency of data access in view3. Difficulty of sing MVC with modern user interface4. Need multiple programmers5. Knowledge on multiple technologies is required6. Developer have knowledge of client side code and html code.

I. Discuss the need for very specific type of communication technologies/techniques for the distributed/web-based systems

Communication techniques/technologies:

1.Functional oriented communication

•RPC/RMI

•CORBA

2.Message Oriented communication

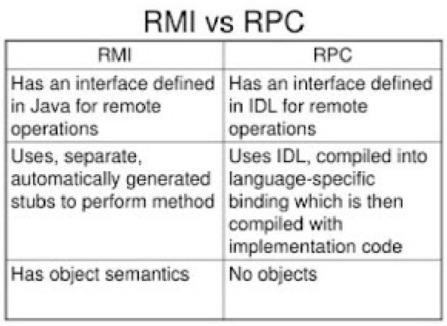
•SOAP

3.Resource oriented communication

•REST

J. RPC vs RMI

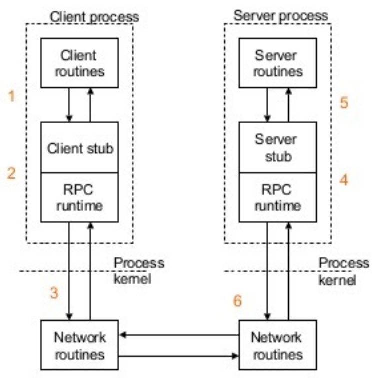
RPC is an old protocol based on C.It can invoke a remote procedure and make it look like a local call. **RPC** handles the complexities of passing that remote invocation to the server and getting the result to client.



RMI vs RPC

RMI	RPC
Has an interface defined in Java for remote operations	Has an interface defined in IDL for remote operations
Uses, separate, automatically generated stubs to perform method	Uses IDL, compiled into language-specific binding which is then compiled with implementation code
Has object semantics	No objects

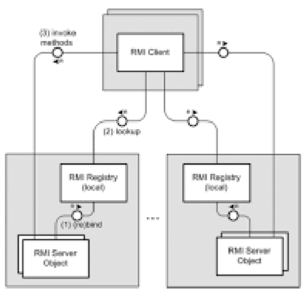
RPC: The basic mechanism



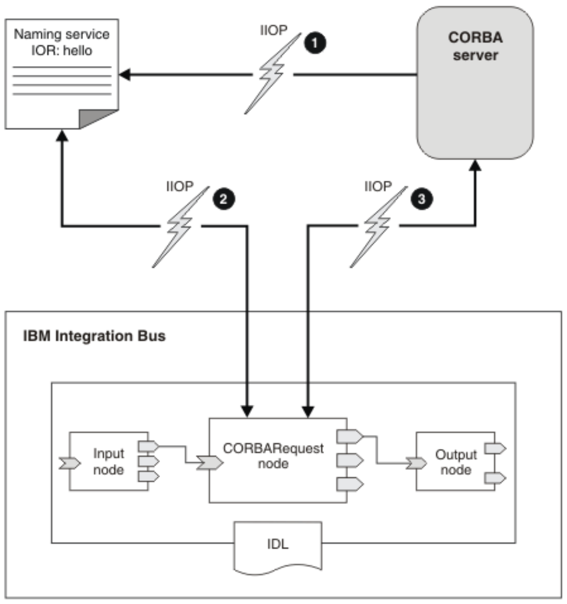
1. Client calls a local procedure on the client stub
2. The client stub acts as a proxy and marshalls the call and the args.
3. The client stub send this to the remote system (via TCP/UDP)
4. The server stub unmarshalls the call and args from the client
5. The server stub calls the actual procedure on the server
6. The server stub marshalls the reply and sends it back to the client

Source: R. Stevens, Unix Network Programming (IPC) Vol 2, 1998

RMI



K. Common Object Request Broker Architecture (CORBA)



Common Object Request Broker Architecture (CORBA) is an architecture and specification for creating, distributing, and managing distributed program objects in a network. It allows programs at different locations and developed by different vendors to communicate in a network through an "**interface** broker."

L. XML specification

What is XML?

- XML stands for eXtensible Markup Language
- XML is a markup language much like HTML
- XML was designed to store and transport data
- XML was designed to be self-descriptive
- XML is a W3C Recommendation

- Designed to store and transport data
- Both human-and machine-readable (self descriptive)
- Often used for distributing data over networks
- Used by may other tools like protocols

•An element has 3 components

1.Start tag

2.Body

3.End tag

```
<name>Saman</name>
```

•There are special type of element with a single self closing tag

```
<price/>
```

For example, XML documents can be very simple, such as the following:

```
<?xml version="1.0" standalone="yes"?>

<conversation>

<greeting>Hello, world!</greeting>

<response>Stop the planet, I want to get off!</response>

</conversation>
```

An element can contain:

- text
- attributes
- other elements
- or a mix of the above

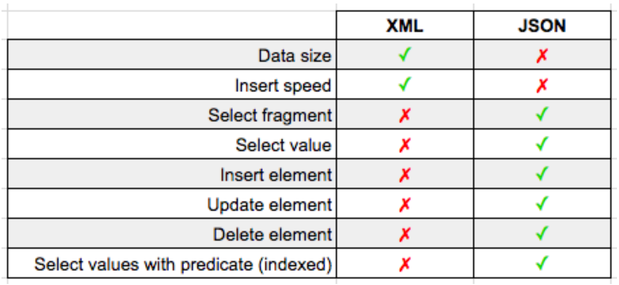
XML Naming Rules

XML elements must follow these naming rules:

- Element names are case-sensitive
- Element names must start with a letter or underscore
- Element names cannot start with the letters xml (or XML, or Xml, etc)
- Element names can contain letters, digits, hyphens, underscores, and periods
- Element names cannot contain spaces

Any name can be used, no words are reserved (except xml).

L. XML vs JSON



	XML	JSON
Data size	✓	✗
Insert speed	✓	✗
Select fragment	✗	✓
Select value	✗	✓
Insert element	✗	✓
Update element	✗	✓
Delete element	✗	✓
Select values with predicate (indexed)	✗	✓

Image result for xml vs json

JSON is Like XML Because

- Both JSON and XML are "self describing" (human readable)
- Both JSON and XML are hierarchical (values within values)
- Both JSON and XML can be parsed and used by lots of programming languages
- Both JSON and XML can be fetched with an XMLHttpRequest

JSON is Unlike XML Because

- JSON doesn't use end tag
- JSON is shorter
- JSON is quicker to read and write
- JSON can use arrays

The biggest difference is:

XML has to be parsed with an XML parser. JSON can be parsed by a standard JavaScript function.

XML is much more difficult to parse than JSON. JSON is parsed into a ready-to-use JavaScript object. For AJAX applications,

JSON is faster and easier than XML:

Using XML

- Fetch an XML document
- Use the XML DOM to loop through the document
- Extract values and store in variables

Using JSON

- Fetch a JSON string

- JSON.Parse the JSON string

JSON is Unlike XML Because

Why JSON is Better Than XML:

XML is much more difficult to parse than JSON.

JSON is parsed into a ready-to-use JavaScript object.

M. Data formatting/structuring

•Plain text

•Files (text, image)

•Query string

•XML

•JSON

Reference

https://www.icar.cnr.it/en/sistemi-distribuiti-e-internet-delle-cose/
http://www.ejbtutorial.com/distributed-systems/service-models-for-distributed-systems
http://www.ejbtutorial.com/distributed-systems/service-models-for-distributed-systems
https://www.google.com/search?q=n-tier+Architecture:&source=lnms&tbm=isch&sa=X&ved=0ahUKEwj38cWx8vLgAhUhiOYKHfYsCk4Q_AUIDigB&biw=1707&bih=803&dpr=1.13#imgrc=AM7DDXKfbBHcTM:
https://stackify.com/n-tier-architecture/
https://techterms.com/definition/mvc
https://download.oracle.com/otn_hosted_doc/jdeveloper/1012/developing_mvc_applications/adf_aboutmvc2.html
https://www.techopedia.com/definition/1293/common-object-request-broker-architecture-corba
https://www.w3schools.com/js/js_json_xml.asp

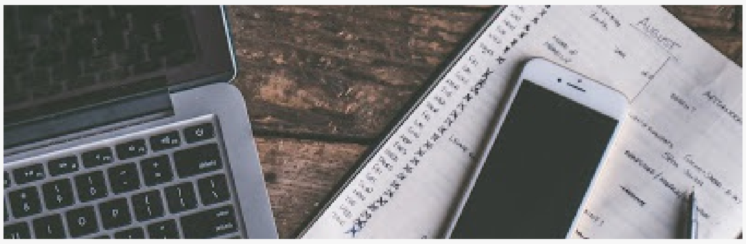
Popular Posts

Programming Applications and FrameworksMalith Iddamalgoda | IT17012584
BLOG NO:03

A. Quality of the code
Code quality matters in many ways.
The long-term usefulness and long-term maintainability of the codeMinimize errors and easily debuggedImprove understandablyDecrease risks

Clear Requirements A project with clear, feasible requirements is much more likely to achieve high quality than ambiguous, poorly specified requirements.The quality of the code can be measured by different aspects Weighted Micro Function Points,Hal stead Complexity Measures,Diplomatic Complexity,Lines of code,Lines of code per method.

B. Explain different approaches and measurements used to measure the quality of code.
Clarity-Easy to read and oversee for anyone who isn't the creator of the code,Maintainable-A high-quality code isn't over complicated,Documented-The best thing is when the code is self-explaining,Refactored-Code formatting needs to be consistent and follow the language's coding conventions,Well-test...



Programming Applications and Frameworks

Programming Applications and Frameworks
A. Compare and contrast declarative and imperative paradigms.*What is Declarative Paradigms?* **Declarative** *Paradigms is refers to code that is concerned higher levels of abstraction and it's Necessary for web development. What is Imperative Paradigms?***Imperative** *Paradigms refers to code that is concerned with lower levels of abstraction and it's Necessary for engineering of algorithms and other low levels.****Declarative and Imperative programming paradigms are describing coding of different levels of abstractions.***

***B. P**rocedural programming and Functional programming.*

What is Procedural Programming?

***Procedural Programming** is a sub category of imperative programming .It is based on the concept of procedural calls. Procedural Programs constitutes of one or more modules. Depending on the programming*