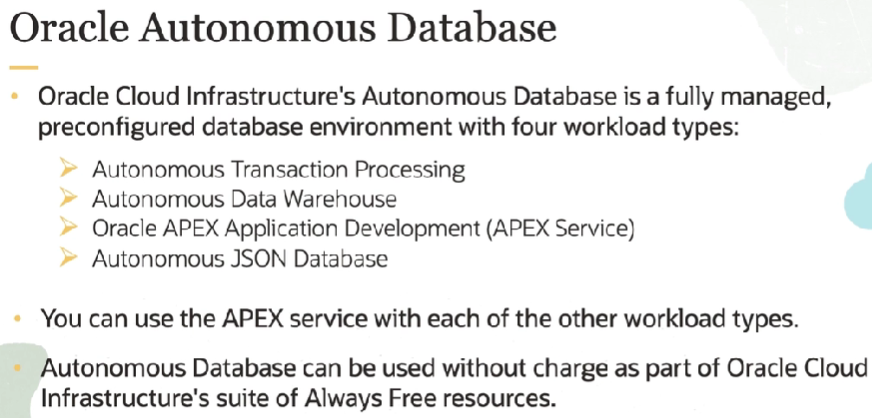
# APEX INTRODUCTION:

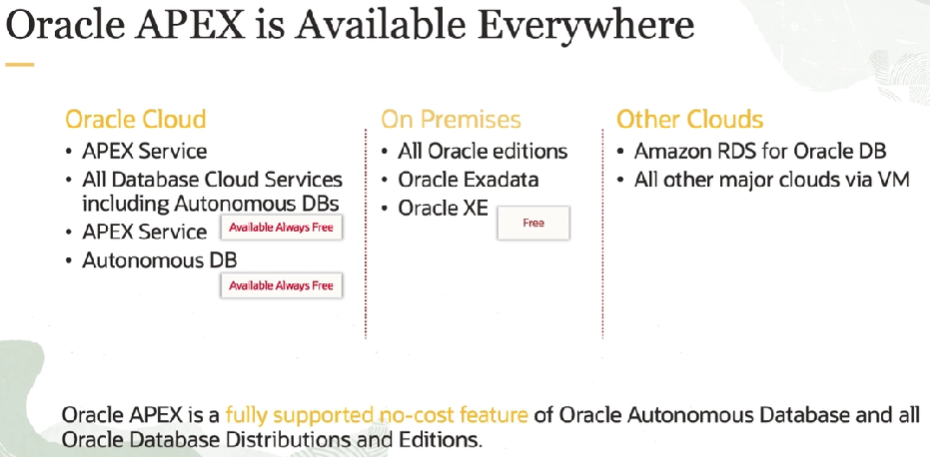
* Oracle APEX is a popular low-code application development platform for enterprise applications.
* Enables the building of scalable, secure enterprise applications with world-class features.
* Designed to create modern, beautiful, and responsive web applications for the Oracle database.
* Accelerates application development, allowing for 20 times faster development with 100 times less code.
* Operates as a part of the Oracle database, providing a single extensible platform for application development.
* Eliminates the complexities of traditional coding and offers world-class components and customizable user interfaces.
* Empowers developers to solve real business problems efficiently without requiring extensive expertise in various web technologies.

ARCHITECTURE

* APEX's declarative framework allows rapid development without extensive coding.
* APEX supports quick transitions from prototype to production, emphasizing agility.
* Pre-built controls in APEX handle database access, user credentials, business rules, etc.
* APEX runs on-premises, in Oracle Cloud, and elsewhere, providing deployment flexibility.
* Oracle Cloud Infrastructure offers APEX Service as an all-inclusive, fast, and low-cost development option.
* APEX Service simplifies costs without constraints on users, developers, applications, or data transfer.
* Oracle APEX uses a three-tier **architecture**, guaranteeing zero-latency data access.
* Web requests flow through Oracle REST Data Services to the Oracle database for processing.
* APEX allows seamless shifting of applications between different Oracle Database environments.

Top of Form





# RESTful

RESTful services, or Representational State Transfer services, are a type of web service architecture that adheres to the principles and constraints of REST.

RESTful services are commonly used for building web APIs (Application Programming Interfaces) that allow different software applications to communicate with each other over the web. They provide a flexible, scalable, and standardized way to expose and consume data and services on the internet.

Popular approach for designing networked applications. Here are key aspects of RESTful services:

1. **Resources:**
   * In REST, everything is considered a resource, which can be a data object, a service, or even a concept.
   * Resources are uniquely identified by URIs (Uniform Resource Identifiers).
2. **HTTP Methods:**
   * RESTful services use standard HTTP methods (GET, POST, PUT, DELETE) to perform operations on resources.
   * Each HTTP method corresponds to a specific CRUD (Create, Read, Update, Delete) operation on a resource.
3. **Stateless Communication:**
   * RESTful services are stateless, meaning that each request from a client to a server must contain all the information needed to understand and fulfill that request.
   * The server does not store any information about the client's state between requests.
4. **Uniform Interface:**
   * RESTful services have a uniform and consistent interface. This is achieved through a set of constraints, including:
     + Resource Identification: Resources are identified by unique URIs.
     + Resource Manipulation through Representations: Resources are manipulated or accessed using standard methods, and their representations are transferred between client and server.
     + Self-Descriptive Messages: Each message includes information needed to understand and process it.
5. **Representation:**
   * Resources can have multiple representations, such as JSON, XML, or others.
   * Clients can request a specific representation using content negotiation.
6. **Stateless Communication:**
   * Communication between the client and server is stateless. The server does not store information about the client's state between requests.
7. **HATEOAS (Hypermedia as the Engine of Application State):**
   * HATEOAS is a constraint that suggests including hypermedia links in the response. These links guide the client on the available actions or transitions to other resources, making the API self-discoverable.
8. **Scalability and Interoperability:**
   * RESTful services are designed to be scalable and can be easily consumed by a wide range of clients, including web browsers, mobile devices, and other applications.
9. **Resource-Based Architecture:**
   * RESTful services are centered around resources and their interactions. Each resource is identified by a unique URI, and interactions occur through the manipulation of these resources.

# ORDS

Oracle RESTful Web Services (ORDS) is a framework provided by Oracle Corporation to facilitate the development and deployment of RESTful web services for Oracle Database. It allows developers to expose database resources as RESTful services, enabling easy interaction with Oracle Database over HTTP. Here are key aspects of Oracle RESTful Web Services:

ORDS simplifies the process of building RESTful services on top of Oracle Database, making it easier for developers to create modern, scalable, and interoperable applications. It provides a way to expose database functionality as web services, allowing for seamless integration with various client applications.

1. **RESTful Architecture:**
   * ORDS follows the principles of Representational State Transfer (REST), which is an architectural style for designing networked applications.
   * Resources (e.g., tables, views, stored procedures) in the Oracle Database are exposed as URI-addressable entities, and interactions are performed using standard HTTP methods.
2. **HTTP Methods:**
   * ORDS supports standard HTTP methods such as GET, POST, PUT, and DELETE for CRUD (Create, Read, Update, Delete) operations on database resources.
3. **SQL and PL/SQL Support:**
   * Developers can define RESTful services by writing SQL queries or PL/SQL procedures that interact with the database.
   * ORDS provides a SQL Developer-based interface for defining and testing these services.
4. **Security Features:**
   * ORDS includes features for securing RESTful services. This includes support for authentication methods such as OAuth, HTTP Basic Authentication, and custom authentication.
   * Fine-grained access control allows developers to restrict access to specific resources based on user roles and privileges.
5. **Customization:**
   * Developers can customize the behavior of RESTful services through templates and configuration options.
   * URI patterns, response formats (JSON, XML), and other settings can be configured to suit the specific needs of the application.
6. **Swagger Support:**
   * ORDS supports Swagger, which is a tool for API documentation and testing. Swagger provides a standardized way to describe and document RESTful APIs.
7. **Installation and Configuration:**
   * ORDS can be installed as a standalone application or deployed within a Java EE container such as Oracle WebLogic Server or Apache Tomcat.
   * Configuration involves setting up database connections, security settings, and other parameters.
8. **Versioning:**
   * ORDS supports versioning of RESTful services, allowing developers to manage different versions of their APIs.
9. **CORS (Cross-Origin Resource Sharing):**
   * ORDS supports CORS, which is important for web applications running on different domains.

# ORACLE APEX COMPONENTS

Oracle APEX components mentioned:

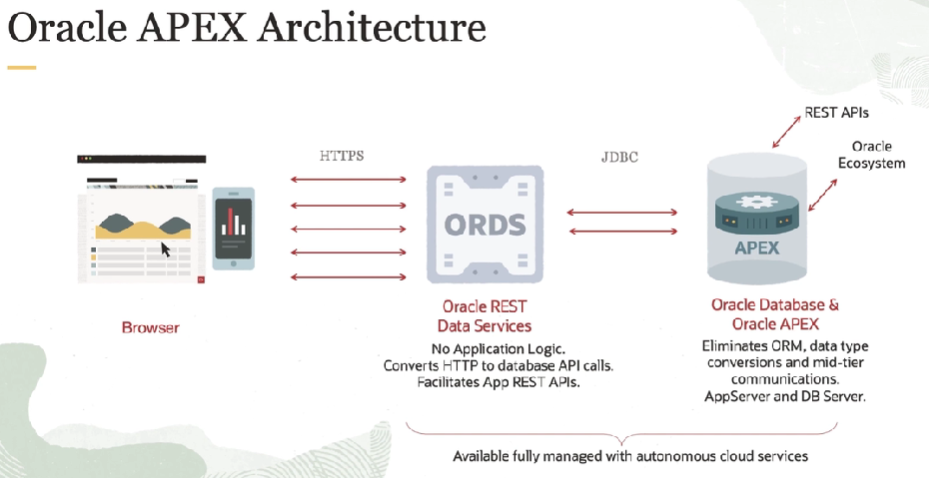
1. **Builder:**
   * The APEX Builder is a user-friendly, web-based development environment.
   * Developers use it to create applications visually, defining pages, data models, security settings, and more.
   * Provides a low-code approach for application design without extensive coding.
2. **Listener:**
   * The APEX Listener, often implemented with Oracle RESTful Web Services (ORDS), acts as middleware.
   * It facilitates communication between user browsers and the Oracle Database.
   * Receives and processes HTTP requests, ensuring smooth interaction between the client-side and server-side of APEX applications.
3. **Runtime:**
   * The APEX Runtime is the environment where APEX applications are executed for end-users.
   * Interprets application logic from the Builder and dynamically renders web pages.
   * Users interact with applications through their browsers, accessing functionalities designed by developers in the APEX Builder.

These components collectively enable efficient development, seamless communication, and user-friendly execution of Oracle APEX applications.

# ORACLE APEX ACRHITECTURE

Oracle APEX follows a three-tier architecture: the client tier, the application server tier, and the database tier. The client tier consists of web browsers, the application server tier hosts the Oracle APEX runtime, and the database tier stores the application data.

Oracle APEX architecture involves the Oracle Database as the backend, the APEX Listener (or ORDS) in the middle for processing and communication, and the end-user's web browser for interaction. This three-tier architecture ensures scalability, separation of concerns, and efficient development and execution of APEX applications.



1. **Database Tier:**
   * At the base of the architecture is the Oracle Database, which stores data and provides the backend processing for APEX applications.
   * The database contains the tables, views, and PL/SQL logic required for the application's data model and business logic.
   * APEX utilizes the Oracle Database as its underlying data store and execution engine.
2. **Application Tier:**
   * The middle tier, also known as the application server tier, includes the Oracle APEX Listener (commonly implemented with Oracle RESTful Web Services - ORDS).
   * The APEX Listener acts as a bridge between the user's browser and the Oracle Database, handling HTTP requests and responses.
   * It interprets APEX applications, communicates with the database to execute SQL and PL/SQL logic, and returns the results to the user's browser.
3. **Web Browser Tier:**
   * The top tier is the web browser tier, where end-users interact with the APEX applications.
   * Users access the applications through web browsers, such as Chrome, Firefox, or Safari.
   * The browser sends HTTP requests to the APEX Listener, and the APEX Runtime dynamically generates and renders web pages based on the application's design.

# SECURITY IN APEX

By leveraging these features and adhering to best practices, Oracle APEX ensures a robust and secure environment for web application development.

1. **Authentication Schemes:**
   * Supports various authentication methods like database, LDAP, and OAuth.
2. **Authorization:**
   * Implements role-based access control for defining user privileges.
3. **Session State Protection:**
   * Includes built-in features to prevent session fixation attacks.
4. **Data Security:**
   * Enables fine-grained access control and row-level security for data protection.
5. **Cross-Site Scripting (XSS) Protection:**
   * Implements measures like automatic escaping and sanitization functions to prevent XSS attacks.
6. **Transport Layer Security (TLS):**
   * Supports secure communication through TLS/SSL protocols.
7. **Security Patching:**
   * Regularly releases security patches and updates.
8. **Security Checklists and Best Practices:**
   * Provides documentation and guidelines for secure application development.

# PAGES IN ORACLE APEX,

In Oracle APEX, a page is a fundamental building block that represents a single user interface screen or component within your application. Each page typically serves a specific purpose, such as displaying data, collecting user input, or performing a particular action.

Each page in Oracle APEX is configured with specific attributes and can be customized to meet the requirements of your application. The navigation between pages is often controlled through buttons, links, or other user interface elements.

There are different types of pages in Oracle APEX, such as:

1. **Interactive Reports**: Pages that allow users to view, filter, and interact with data in a tabular format.
2. **Form Pages**: Pages that facilitate data entry or modification through forms.
3. **Charts**: Pages that display graphical representations of data using various chart types.
4. **Classic Reports**: Pages that show data in a customizable report format.
5. **Master-Detail Pages**: Combining a master report with one or more detail reports to show hierarchical data.
6. **Web Services**: Pages that allow integration with external web services.

calendars, and dashboards, allowing developers to create dynamic and engaging web applications.

