Introduction to controllers

Introduction:-
Controllers are the active parts of a Web Dynpro application. They define how the user can interact with the Web Dynpro application. The data that a controller can access is defined in the corresponding context. Different instances of controllers and contexts exist within a Web Dynpro application. Controllers contain the source code of a Web Dynpro application; they hold the context data and care for event handling and navigation. Any programming of flow logic requires the knowledge of how controllers are defined and what kind of entities they offer.

Controller Types
There are four types of controllers in an ABAP Web Dynpro component. These different controller types differ in the entities they are composed of:
1. **Component controller:-**
   There is only one component controller per Web Dynpro component. This is a global controller, visible to all other controllers. The component controller drives the functionality of the entire component. This controller has no visual interface. The lifetime of the component controller equals the lifetime of the component. When starting a Web Dynpro application, the component controller is instantiated by the Web Dynpro runtime.
2. **Custom controllers**
   Custom controllers are optional. They have to be defined at design time and can be used to encapsulate sub-functions of the component controller. Multiple custom controllers can be defined in a component. Custom controllers are instantiated automatically by the Web Dynpro framework and the instantiation order is undefined; therefore, the coding in a custom controller should make no assumptions about the existence of any other custom controller. The instantiation of a custom controller is delayed until the first method of the controller is called. Custom controller instances cannot be deleted explicitly.
3. **Configuration controller**
   This is a special custom controller. It is only necessary if the corresponding component implements special configuration and personalization functionality. Only one configuration controller may exist in any component. Any other controller can access the configuration controller, but the configuration controller cannot access any other controller. This controller is instantiated as the first controller of the component. It lives as long as the component lives.
4. **View controllers**
   Each view consists of the layout part and exactly one view controller. This controller cares for view-specific flow logic, like checking user input and handling user actions. The instantiation of a view controller is delayed until the first method of the controller is called. The lifetime of a view controller can be controlled by the views properties. If *framework controlled* is selected, the view instance will be deleted with the component. If *when visible* is selected, the view instance is deleted as soon as the view no longer belongs to the view assembly.
5. **Window controllers**
   Each window has exactly one window controller. This controller can be used to store the data passed via the inbound plugs when being reused as a child controller. Methods of this controller can be called from the inbound plug methods of the window. The instantiation of a window controller is delayed until the first method of this controller is called. This is done by starting a Web Dynpro application or by embedding the related interface view in the parent component's window. Window controller instances cannot be deleted explicitly.

Common Controller Entities:-
Each controller has its own context. The context root node already exists. All other nodes and attributes have to be defined statically or by source code. For all controllers, there exist methods that are called by the Web Dynpro framework in a predefined order. These are called hook methods. Depending on the controller type, there are different hook methods available. At least two hook methods are contained in all controller types. These methods are processed only once during the lifetime of a controller instance: when a controller instance is created (*wddoinit()*) and when a controller instance is deleted (*wddoexit()*). Attributes not related to the value or property of UI elements can be declared using the Attributes tab. These attributes are then visible in all methods of this controller. There are two predefined attributes, which are used to access the functionality of the controller (*WD_THIS* and of the context (*WD_CONTEXT*). For information to be shared between different controllers, one controller must declare the use of another controller. This is done on the *Properties* tab of the controller that needs to access another controller. The most frequent requirement for this kind of data sharing is when it is useful to create a mapped context node or we wish to access another controller's instance methods. We may not specify the name of a view controller as a used controller, as this would violate good MVC design principles. A view controller should be written such that it is responsible for nothing more than the display of data and user interaction. A view controller is not responsible for generating the data it displays; that is the role of a custom controller. Business logic, such as function modules, BAPIs, or methods in helper classes can be accessed from the methods of all controllers.

Special Entities of Component/Custom Controllers:-
For both component and custom controllers, events can be defined with arbitrary parameters. Any method of any other controller (also view and window controller) that registers for these events if this method is defined as an event handler method. A typical use of such events is the invocation of processing in a view controller after processing in the component controller has been completed. This can be achieved when a method in the view controller subscribes to an event raised by the component controller. Using our design time declarations, the Web Dynpro framework will automatically manage the definition, triggering, and handler subscription to such events for us. If two or more methods subscribe to the same event, the order in which they will be executed is undefined. Only the component controller has additional standard hook methods. These are processed directly before the navigation requests are processed (*wddobefournavigation()*) and after all views of the view assembly to be sent have been processed (*wddopostprocessing()*). Attributes, methods, context elements, and events can be marked as interface elements. These elements are then exposed to other components by means of the interface controller.

Special Entities of View Controllers:-
A view controller is a visual building block of a Web Dynpro component and is designed to handle all aspects of data display and user interaction. For navigation to take place between different Web Dynpro view controllers, special navigation events and navigation event handlers have been created. These are called navigation plug. A navigation event is raised when an outbound plug is fired. Using the generic name *<Outbound plug>* for an outbound plug, its declaration will cause a method to be generated in the view's component controller, called *FIRE<Outbound plug>*. This method is only visible for the Web Dynpro framework; it is not visible to the developer. An inbound plug is the navigation event handler that can be registered to a navigation request. Using the generic name *<Inbound plug>* for an inbound plug, its declaration will cause a method to be generated in the view's component controller, called...
A static registration of an inbound plug (method HANDLE<Inbound plug>) to the navigation event raised by an outbound plug (method FIRE<Outbound plug>-PLG) is called a navigation link. Navigation links are not part of a view but are defined in a window embedding the view. Thus, in different windows, the event registration can be defined differently. An action links a client-side event to an event handler method defined in the corresponding view controller. When defining an action having the name <Action>, an event handler method (ON ACTION<Action>) is automatically generated. If a view will be replaced as a result of a client event, the related outbound plug can be fired in this action event handler method. There are two special hook methods found in view controller: wddobeforeaction() is processed if a client event is fired in any view. Before the action event handler methods are processed, thewddobeforeaction()_methods are executed for all views that are part of the last view assembly: _wddomodifyview() is a method that allows us to programmatically manipulate the layout of the view. This event can be used to define the UI dynamically. In addition to the standard attributes of all controllers, a view controller has a reference to the component controller of its component: WD_COMP_CONTROLLER. This reference can be used to access functionality related to the component controller. It is used when accessing messages or when calling a method declared in the component controller.

**Special Entities of Window Controllers:**

Window controllers are very similar to view controllers. Technically, it is like a view controller without a UI (view layout). All views that are to be displayed when using a Web application have to be embedded in the window that is referred to by the application or the calling component. The Web Dynpro window contains the structure of all views to be displayed and the navigation links defining the possible view assemblies. Each Web Dynpro window contains outbound plugs and inbound plugs, like views. We can use the plugs to set up cross-component navigation. To expose the plugs to the component interface, select the property Interface for each plug. These plugs will then be part of the related interface view. The window controller also has the special attribute WD_COMP_CONTROLLER, which holds the reference to the component controller.