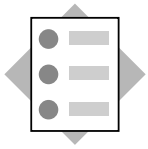
Web Dynpro Introduction

Contents:

- **Introduction**

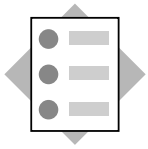
Web Dynpro Introduction: Objectives



After completing this lesson, you will be able to:

- **Understand the basic idea behind Web Dynpro.**
- **Explain basic features of Web Dynpro.**
- **Understand basic concepts of Web Dynpro.**

Overview: Topic Objectives



After completing this topic, you will be able to:

- **Understand the basic idea behind Web Dynpro.**

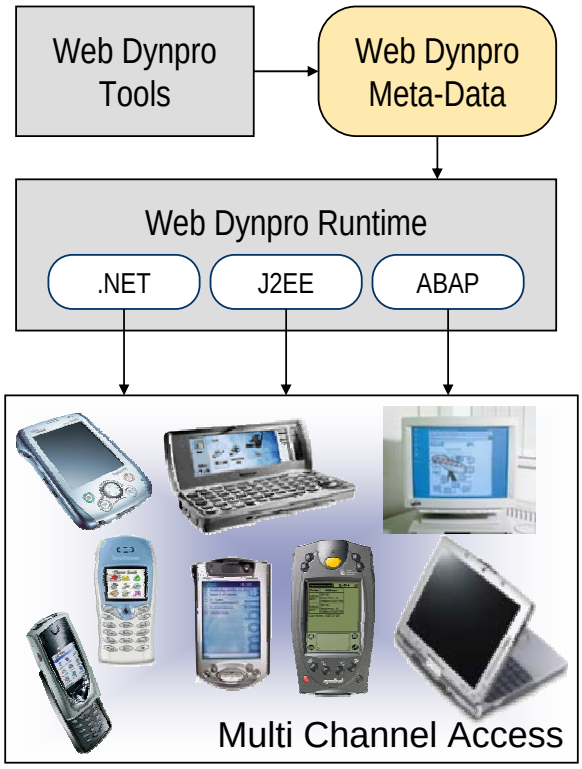
What is Web Dynpro ?

- A Programming Model for User Interfaces
  - Platform-independent (Java, ABAP, ...)
  - Defines a standard structure for user interface applications
    - Derived from the MVC (“model-view-controller”) design pattern
- A Set of Tools for User Interface Design
  - Focus on graphical modeling
    - Code is generated from the meta-model declarations
  - Integrated in SAP NetWeaver Developer Workbench
- A Runtime Environment for Applications
  - Framework running on J2EE server offers common services
  - Client-side technology for browser-based user interfaces
    - XML-based protocol makes alternative clients possible
- A Technology for Software Modularization
  - Components help structure projects and support pattern-based UIs

- **What is Web Dynpro?**
  From a technological point of view, SAP’s Web Dynpro for Java is a revolutionary step in the development of web-based user interfaces. It is completely unlike any design paradigm ever used by SAP before and represents a quantum leap in the development of web-based, ERP applications
- **What is the Design Philosophy Behind Web Dynpro?**
  Web Dynpro applications are built using declarative programming techniques based on the Model View Controller (MVC) design paradigm. That is, you specify what user interface elements you wish to have on the client, and where those elements will get their data from. All the code to create the user interface is then generated automatically within a standard runtime framework. This relieves you from the repetitive coding tasks involved in writing HTML and then making it interactive with JavaScript.

Web Dynpro Main Benefits

- Deliver an Enterprise Quality Web Development Environment
  - minimize coding, maximize design
  - separate layout and logic
  - support arbitrary backends
  - support reuse of components
  - configuration of UI patterns
  - support web services and data-binding
- Achieve Independence
  - run on multiple platforms
- Improve User Experience through a "High Fidelity Web UI"
  - browser based, zero footprint
  - screen updates w/o page reloads
  - client-side dynamics
  - performance through caching
  - 508 accessibility support
  - flicker-free screen, minimal refreshs



- **Web Dynpro Main Benefits**

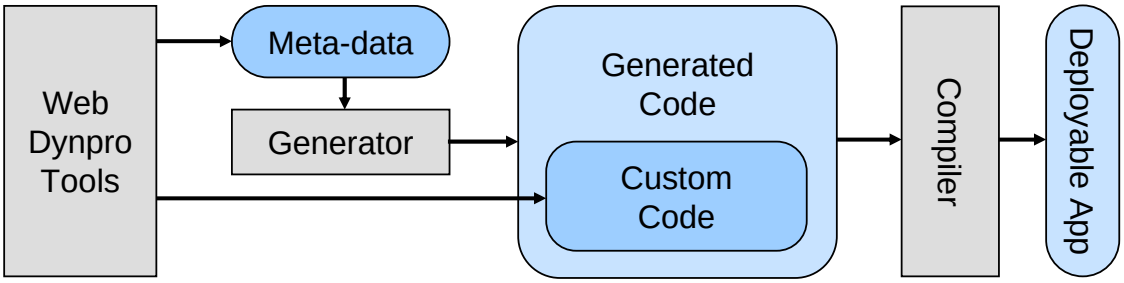
  Web Dynpro's main goal is to enable application developers to create powerful Web applications with a minimum of effort using descriptive tools in a structured design process. The key is to describe an application – any programming will only be a very minor part of application development. The applications should run on a range of devices and on various types of network – this is important for collaboration scenarios. Customers must also be able to personalize them.

  One guiding principle in the Web Dynpro philosophy is: the fewer lines of hand-written code there are in the UI, the better. Web Dynpro pursues this goal in two ways. First, Web Dynpro establishes a declarative language for specifying qualities of a user interface without writing any code. From this abstract definition, Web Dynpro generates code for a ready-to-run program skeleton of the user interface, and it also derives meta-data that a generic engine can interpret at runtime. Hand-written code still has its place, but it is reduced to a minimum.

  Second, it provides technical features such as support for internationalization, flicker-free interaction and a clean separation of the business logic and the user interface. To ensure this separation Web Dynpro employs the Model-View-Controller (MVC) paradigm, first implemented in Smalltalk-80.

  From the meta data created during the design phase, Web Dynpro is able to generate code for runtimes like J2EE and ABAP, and will also support .NET runtime.

Meta-model Declarations vs. Custom Coding



Meta-model Declarations

Guarantees common app design

Good for graphical tool support

- Screen Layout and Nesting
- Navigation and Error Handling
- Data Flow
- Componentization
- ...

Custom Coding

Guarantees universality

Good for data-driven, dynamic apps

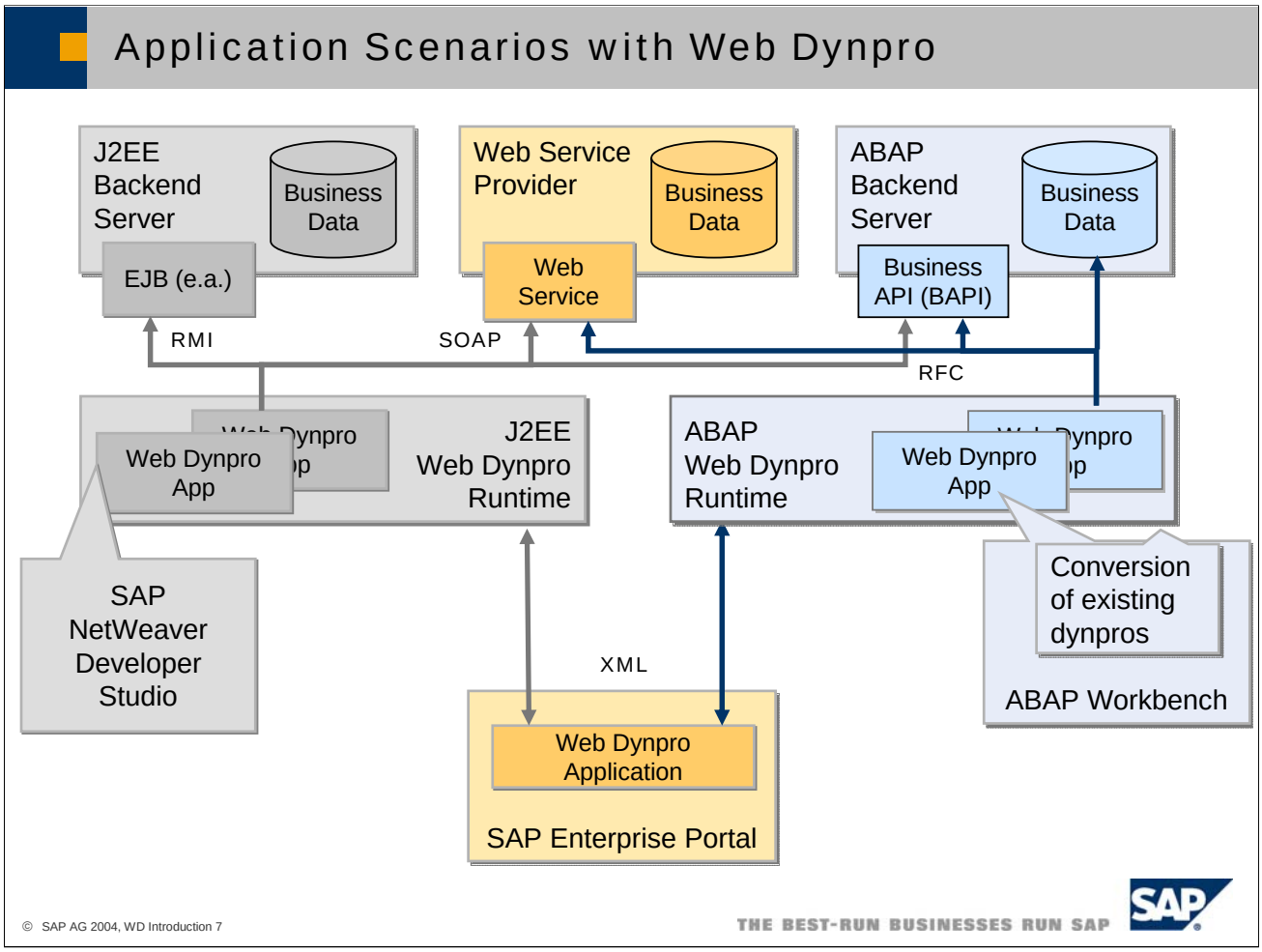
- Implementation of business rules
- Dynamic screen modifications
- Access to services (files etc.)
- Portal eventing
- ...

- **Metamodel Concept and Declarative Programming**

  Web Dynpro provides support for the development of Web applications in the form of a declarative programming approach. You use special tools to describe the properties of a Web Dynpro application in the form of Web Dynpro metadata. The necessary source code is then generated automatically and executed at runtime. In addition to the events provided by the framework, you can also define your own events for a Web Dynpro application. For this purpose, the source code contains separate, specially marked areas.

  In a Web Dynpro application, every user interface is always made up of the same basic elements, with the exception of the individual interface elements for each application. You declare these elements of the metamodel statically using the Web Dynpro tools. It is also possible to implement elements of the metamodel in programs and integrate them dynamically at runtime. Using these implementations, you can make changes or enhancements to a user interface that has been created by declarative methods by generating new interface structures at runtime. This means that you can combine the declarative procedure with the implementation of source code.

  The metadata of a Web Dynpro application is independent of the platform on which the application is executed. It is therefore possible to transport an application to a platform after implementing it on another. The metadata is transported to the platform-specific development environment; the source code required for this platform is generated. Only the source code that you have programmed for the event handling, for example, would have to be adapted to suit the new platform.



■ **Application Scenarios with Web Dynpro**

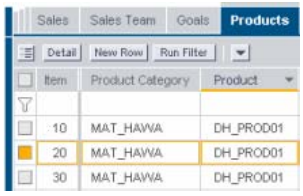
The Web Dynpro programming model supports the following backend systems:

1. BAPIs in the ABAP backend system. With support of the SAP Enterprise Connector, you can generate quickly and comfortably Java proxies which you then use within your Web Dynpro application. The BAPI call is done by an interface which is created from the SAP Java Connector (SAP JCo).
2. Enterprise JavaBeans (EJBs) which encapsulate the application logic
3. Web Services
4. External XMI models (TogetherSoft / Rational Rose): The source code can be generated from an UML definition of the Web Dynpro interface. The Web Dynpro application developer can import a UML definition as XMI file with wizard support (Web Dynpro Model Tools).

Pattern-based UI Design

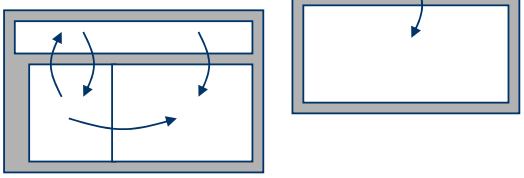
- Three levels of UI patterns
  - Controls
    atomic elements in the layout, constitute the look & feel
  - Components
    reusable, task-oriented building blocks
  - Floor plans
    screen layout, interaction and semantics for a generic application
- Consistent User Interfaces
  - faster learning, less training
  - less specialized knowledge

Controls



Components

Floor plans



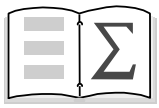
■ Pattern-based UI Design

A UI pattern is a solution to an interaction problem in the context of a user task. Technically, a UI pattern is a generic, configurable Web Dynpro UI element group that has been built to deal with tasks common to a number of applications.

Using UI patterns has the following benefits:

- A unified design both within and across applications
  - A sharp separation between front end and backend so changes to the UI do not affect the business logic
  - More rapid and efficient software development
  - Lower development costs
  - Reuse
  - Reduced maintenance requirements
  - Improvements to patterns can be made once centrally

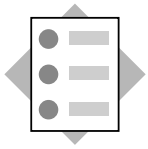
Overview: Topic Summary



You should now be able to:

- **Understand the basic idea behind Web Dynpro.**

Features: Topic Objectives

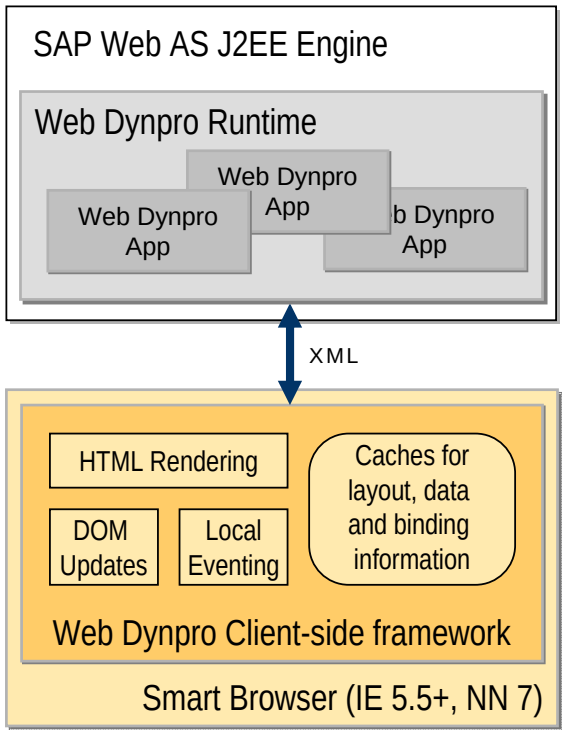


After completing this topic, you will be able to:

- **Explain basic Features of Web Dynpro.**

Web Dynpro Features

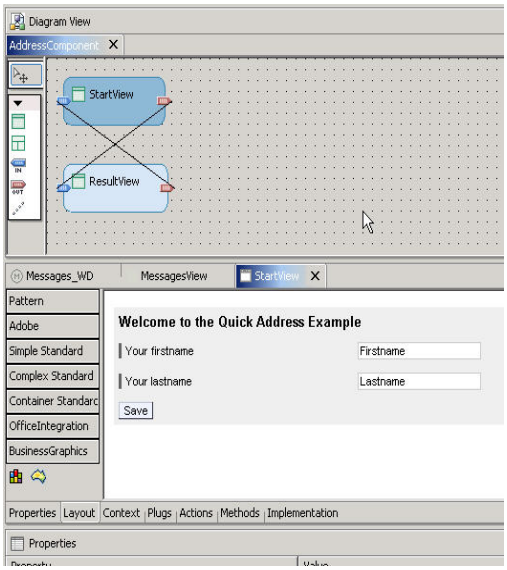
- Client-side framework (CSF) for an advanced user experience
  - Comprehensive UI element library (exceeds HTMLB and CRM UI)
  - Zero footprint
    - JavaScript library < 100 KBytes (compressed)
    - Runs in Internet Explorer 5.5 (or higher)
  - Integration of external components (e.g., Microsoft Office, Adobe Forms)
- Performance-optimized protocol between client and server
  - Load-on-demand for tabular data
  - Delta-transfer
    - layout from server to client, application data back and forth
- Most 508 accessibility features
  - Fully keyboard-enabled



- **Client-Side Framework (CSF)**
  A JavaScript-based client software application, the Client-Side Framework (CSF) runs in the user's browser. The CSF enables the browser to send an http-based abstract definition of the user interface. This information is evaluated by the CSF while the application data is passed separately. The CSF then creates the complete application from the user interface definition and application data.
  CSF offers the following advantages:
- Faster screen generation: Updating of screens is restricted to the changed area
- Flicker-free display of screen output: The end user sees a static screen
- Reduction in the number of server requests through intelligent use of caches
- Higher level of security for the Web application

Web Dynpro Features (2)

- Tools in SAP NetWeaver Developer Studio
  - Integrated as Web-Dynpro-specific perspective in Eclipse
  - Graphical View Designer and AppModeler
    - WYSIWYG, drag & drop, ...
  - Integration with SAP Java Infrastructure
    - Design-time Repository for source code management
    - Access to Java Runtime Data Dictionary
    - Deployment via SDM
- Internationalization
  - Support for SAP translation text format
  - Message pool editor
- Model Interfaces
  - BAPIs via Adaptive RFC
  - Arbitrary models via XMI import (e.g., EJB)
  - . . .



■ **Web Dynpro tools**

To support the declarative programming concept, the SAP NetWeaver Developer Studio contains a range of Web Dynpro tools.

- In a tree structure, the **Web Dynpro Explorer** provides a logical overview of the Web Dynpro application.
- The **navigation modeler** provides comprehensive graphical support for application design,
  implementation of interface elements and their alignment on the screen, and navigation definition for the flow sequence of the interface elements.
- The **View designer** is a graphical tool that provides support when designing Web Dynpro layouts, including a WYSIWYG function.
- **Model tools:** The data for a Web Dynpro application is provided using models. There are specific model types for all the different back-end scenarios.
- The **data modeler** provides support both for the definition of models and custom controllers and their usage, and for the creation of data links for mapping definitions. You can also use this tool to create views.
- The **controller/context editor** provides graphical support for the implementation of the data flow.
- **Message editor:** A message wizard provides support for the quick definition of user outputs.

■ **Supported back-end systems**

The following back-end systems are supported and can be used by a Web Dynpro application:

- The SAP Enterprise Connector enables quick and simple generation of Java proxies
- Encapsulation of the processing logic in Enterprise JavaBeans (EJBs)
- Use of Web services
- A UML definition can be imported into the Web Dynpro application as an XMI file

Web Dynpro Features (3)

- Comprehensive set of standard UI controls
  - According to Unified Rendering and Unified UI Element standards

Button	Caption	Chart (onSelect events!)
Checkbox [group]	Dropdown list box	Group
HTML Frame	Image	Input field
Label	Link to action	Link to URL
Menu (only in tray)	Progress Indicator	Radio button [group]
Road Map	Scroll bar	Table
Tab strip	Text edit	Text view
Tool bar	Tray	Tree

- Table cells containing nested controls
  - Buttons, drop-down list, links, images, checkboxes, radio buttons
  - In-place editing (with automatic undo support!)

- There are numerous user interface elements available for designing the user interface of a Web Dynpro application. All available user interface elements are divided into categories and can be selected using buttons in the view designer.

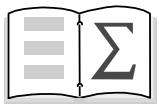
Web Dynpro Features (4)

- Declarative screen management
  - Navigation graphs
  - Nested views and pop-up windows
- Layout managers (Grid, flow, matrix, row) with arbitrary nesting
- Generic UI Services based on meta-data
  - Extended Value Selector (‘F4’)
    - Metadata for value selection from dictionary or defined dynamically
  - Automatic conversion / checks / error handling for basic types
  - Comprehensive application error handling
- Dynamic creation / modification of meta-model elements
  - Views and layout elements
  - Context element (local variables) and data types

Web Dynpro Features (5)

- Component concept for encapsulation and reuse
  - Stand-alone component interfaces
  - Create multiple instances of embedded components dynamically
- APIs for using server interfaces
  - Access to the System Landscape Directory
  - Setting the session time-out
  - Access to URL parameters
- UME integration
  - Includes Single Sign-On (SSO2)
- Portal integration
  - Support for client-side portal eventing
  - Pick up themes and style sheets

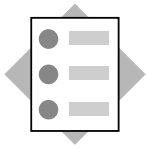
Features: Topic Summary



You should now be able to:

- **Explain basic Features of Web Dynpro.**

Basic Concepts: Topic Objectives

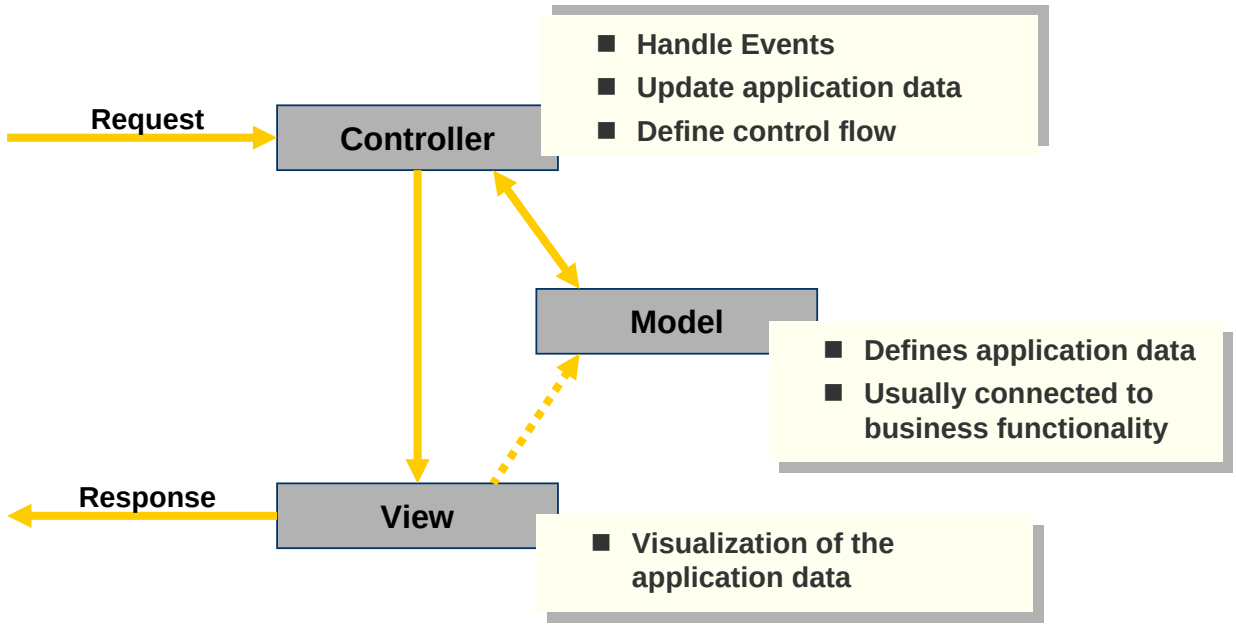


After completing this topic, you will be able to:

- **Understand the basic concepts of Web Dynpro.**

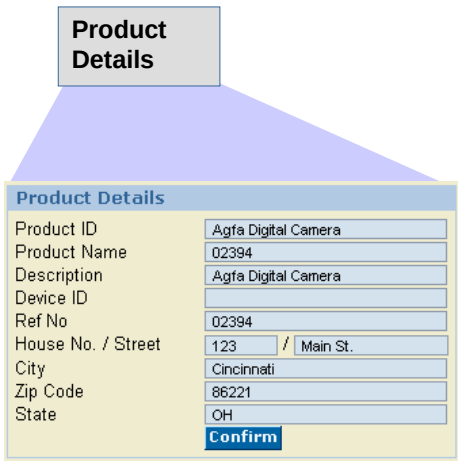
Model View Controller (MVC) Approach

Design pattern for decoupling presentation and logic of an application



- **Model View Controller (MVC) approach**
  Using Web Dynpro technology, you can get a clear separation of application and presentation logic. A Web Dynpro application runs on the front end and has local or remote access to the back-end system via a special service. The complete presentation logic is part of the Web Dynpro application, while business logic and persistence of the business objects run in the back-end system.
  Each Web Dynpro application follows the Model View Controller approach. The **model** is the interface to the back-end system and is responsible for providing data to the entire application. A **view** is the central logical layout element of the Web Dynpro application. It is responsible for processing the presentation logic and for the browser output. A **controller** handles the data flow between the model and the view in both directions.

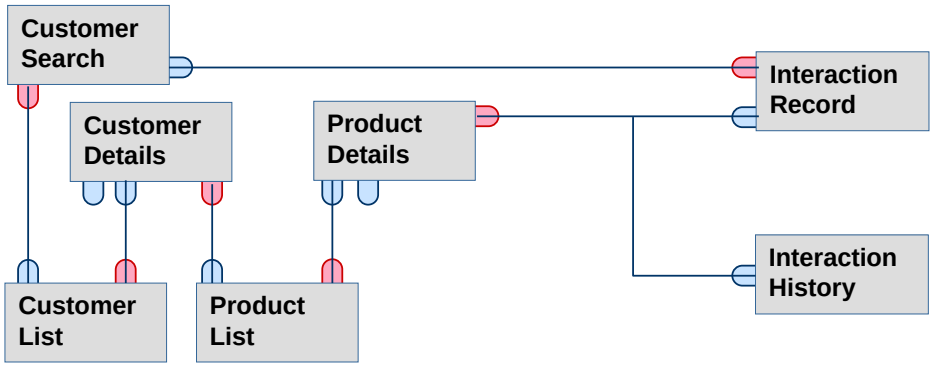
Views, and Layouts



- Each view has its own layout

- **Views and layouts**
  A view describes the layout and behavior of a rectangular area of a user interface.
  Every Web Dynpro application has at least one view. The layout of a view is made up of different user interface elements, which can be nested in each other. The positioning of interface elements in one view is supported by the supplied layout variants.

Views, Layouts, and Navigation



- Each view has its own layout

Navigation links define possible view sequences with plugs (out-bound and in-bound)

■ **Views, layouts, and navigation**

Navigation between different views is enabled by plugs. These can be divided into inbound and outbound plugs. While inbound plugs define the possible entry points of a view, outbound plugs enable navigation to another view.

In general, several views are embedded within a Web Dynpro window, which is why it is necessary to qualify a view as the view that is displayed first when a window is called. This view is assigned the *StartView* property. The subsequent navigation structure is then created using this view.

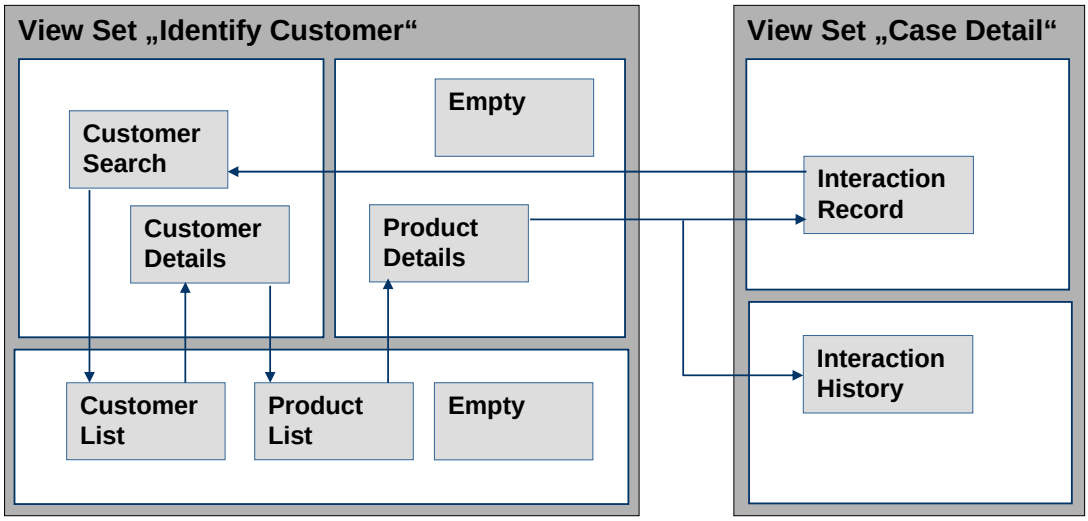
The entering of a view always causes an event handler method to be called. This is why an event handler method is automatically generated for every inbound plug

To navigate from one view to another, each outbound plug from the first view must be linked with an inbound plug from the second view with the help of a navigation link.

Exactly one navigation link can originate from one outbound plug.

In contrast, an inbound plug can be controlled by several outbound plugs.

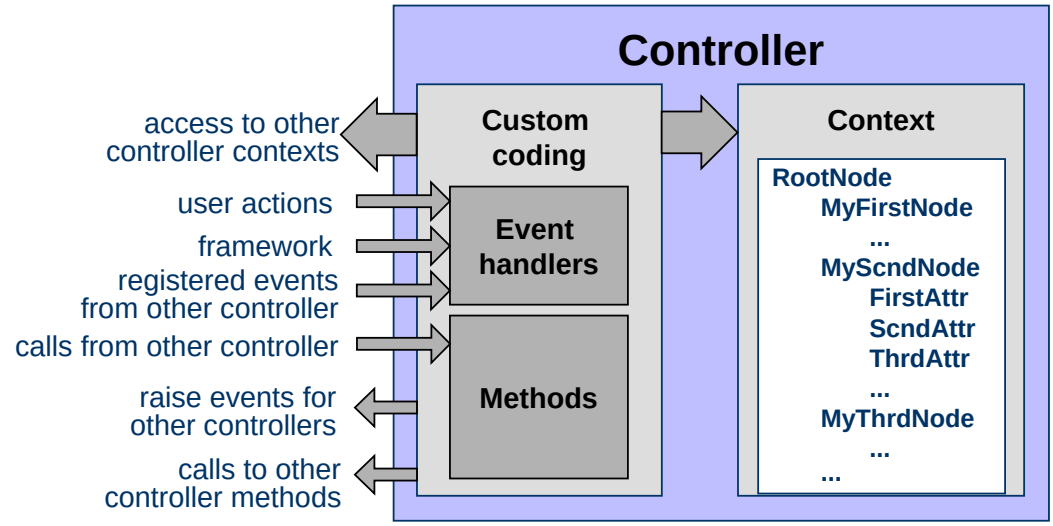
View Sets and View Areas



- Each view has its own layout
- Navigation links define possible view sequences
- A view area can display multiple views, but only one at a time
- View sets are arrangements of view areas

■ **View sets and view areas**
A view set provides a visual frame with predefined subsections (view areas) into which you can embed your views at design time

Web Dynpro Controller



Controller = Context (Local Data) + Custom Coding

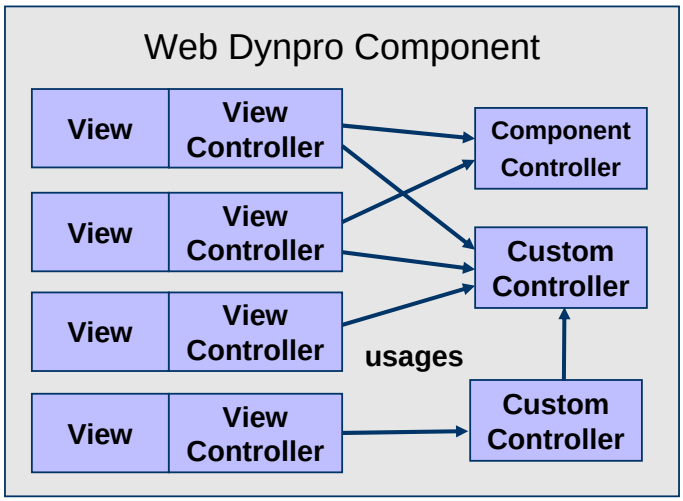
Access to other controllers is ruled by “usage” relations

Custom coding is required for things that can not be expressed in the meta-model

Each controller owns a hierarchically structured set of local data, called the controller’s context

- **Web Dynpro controller**
  Controllers are the active parts of a Web Dynpro application. They define how the user interacts with the Web Dynpro application.
  The data that a controller can access is defined in the corresponding context. The context serves as the local data storage for the corresponding controller.

Views and Controllers



- Views define what the user sees on the screen
- View controllers handle events from the user
- Custom controllers offer global services

■ **Views and controllers**

Different instances of controllers and contexts exist within a Web Dynpro application. In addition to view controllers, which control the behavior of an individual view, there are also global controllers, which offer more general services for all the views of a component.

Each view has exactly one view controller, which processes the actions performed by the user in the view.

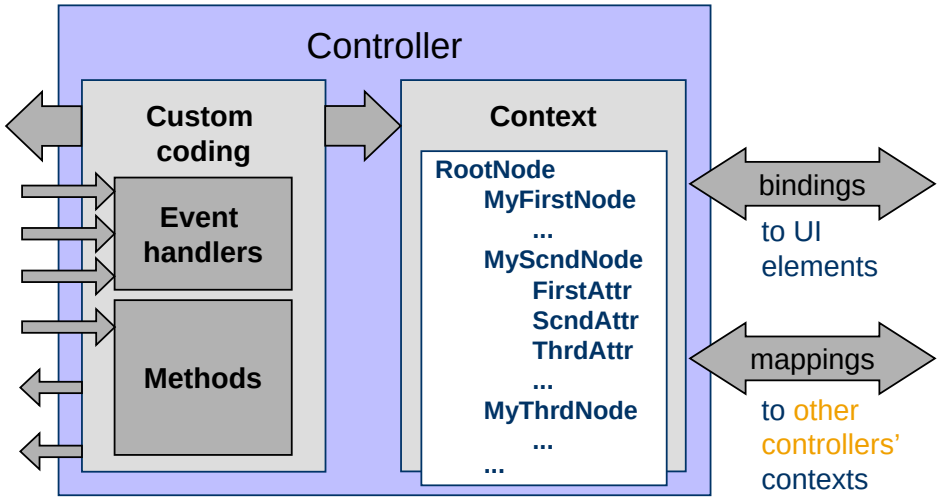
A view also has exactly one view context, which contains the data required for the view.

A view controller and the corresponding context exist for as long as the view is displayed in the browser.

If the view is replaced by a successive view, the local data is also no longer available.

Moreover, each Web Dynpro component has at least one global controller, called the **component controller**. The lifetime of the component controller is determined by the lifetime of the entire application.

Data Binding and Context Mapping



- Access to other controllers is ruled by "usage" relations
- Custom coding is required for things that can not be expressed in the meta-model
- Each controller owns a hierarchically structured set of local data, called the controller's context
- Bindings and mappings are for automatic data exchange

■ **Data binding and context mapping**

Within the Web Dynpro architecture, the contexts of the different controllers can be linked in different ways:

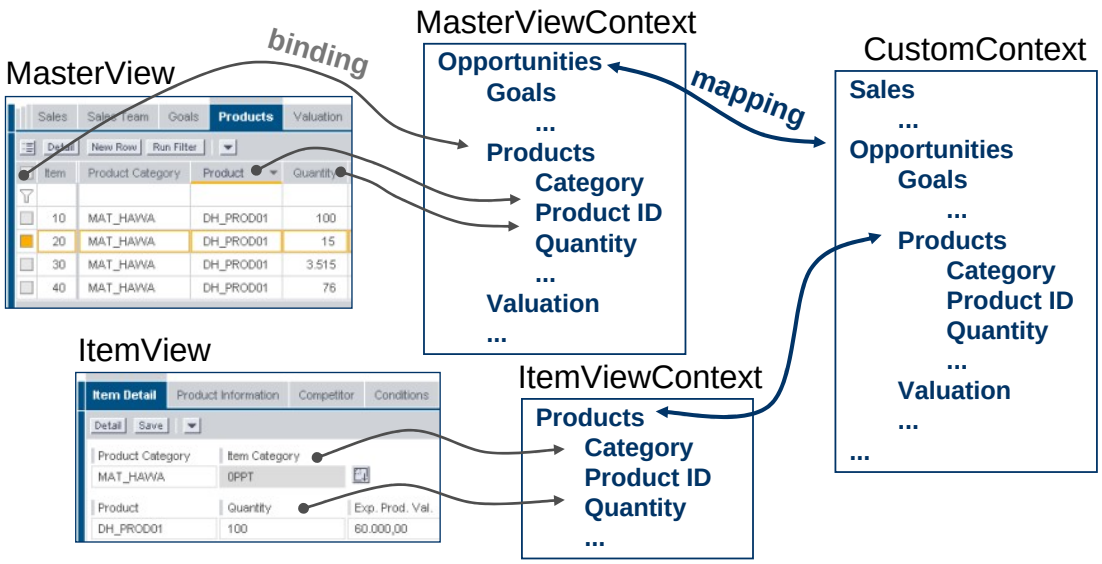
An element of a view context can be linked to a UI element (**data binding**).

A mapping can be defined between two global controller contexts, or from a view context to a global controller context (**context mapping**).

The context of a global controller can be linked to a Web Dynpro model.

Whenever an element of a view context is mapped to the corresponding element of the component context, the data is stored in the global component context, not in the local view context.

Data Binding and Context Mapping (2)



Automatic data transport through binding and mapping

The controls in each view can be bound to the context of the view controller

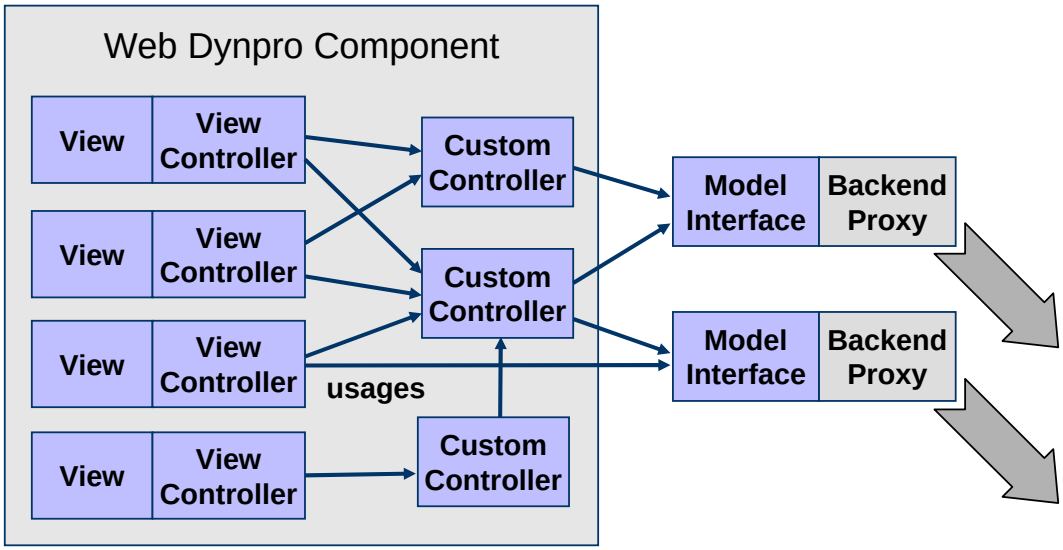
Some controls (e.g., TableView) can be bound to multiple nodes

Context nodes can be mapped to similar nodes in other contexts

Mappings will propagate data and selection state back and forth

- **Data binding and context mapping**
  Every view always possesses a controller, which saves its local data in a **view context**.
  A UI element can be bound to a context only if it belongs to the same view. In general, however, the lifetime of a view context is too short, and its visibility too restricted, for it to be suitable for saving data used across several views. This is where the standard context of the Web Dynpro application comes into play. This standard context belongs to the controller of the Web Dynpro component. Its lifetime is determined by the lifetime of the whole application. Moreover, this context can be made visible to some of the view controllers and not others. So that you do not have to copy data explicitly between two contexts, you can map the relevant elements of the two contexts to each other. This is known as context mapping. Whenever an element of a view context is mapped to the corresponding element of the component context, the data is stored in the global component context, not in the local view context.

Web Dynpro Models



Model View Controller

Views define what the user sees on the screen

View controllers handle events from the user

Custom controllers offer global services

Models provide access to the interfaces in the back end

Proxys connect to the back-end system (mySAP, Web Services, ...)

THE BEST-RUN BUSINESSES RUN SAP

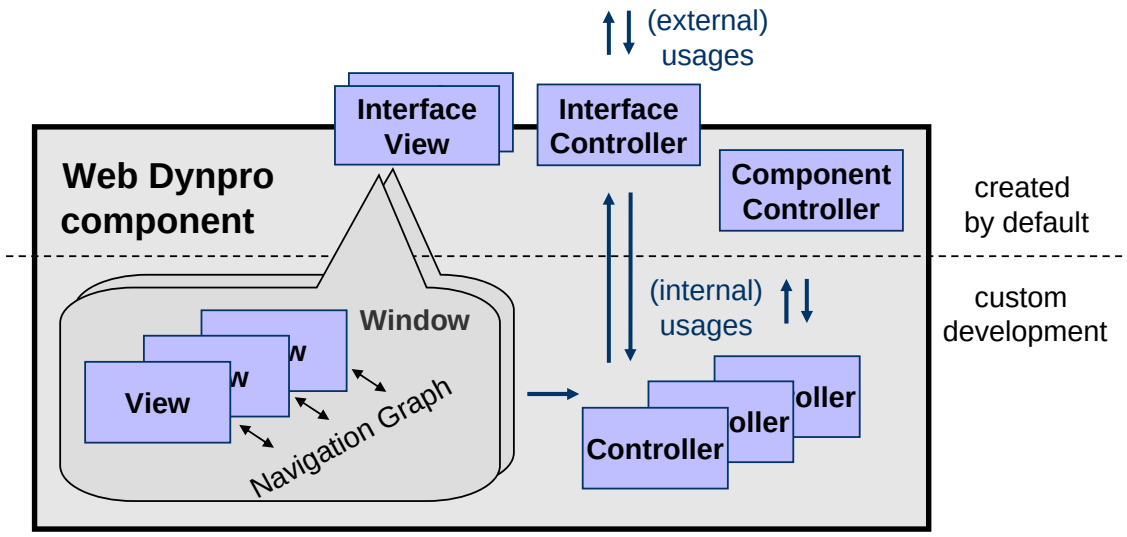
■ **Web Dynpro models**

The model retrieves the application data from the back end.

The data for a Web Dynpro application can originate from different sources. You can:

- Call and use SAP data from a SAP back end using BAPIs
- Define new data
- Call and use Web services
- Combine the three above procedures
- Import an external model using the appropriate tools

The Interface of a Component



A component's interface controller can be used by the embedding component for mappings, eventing, etc.

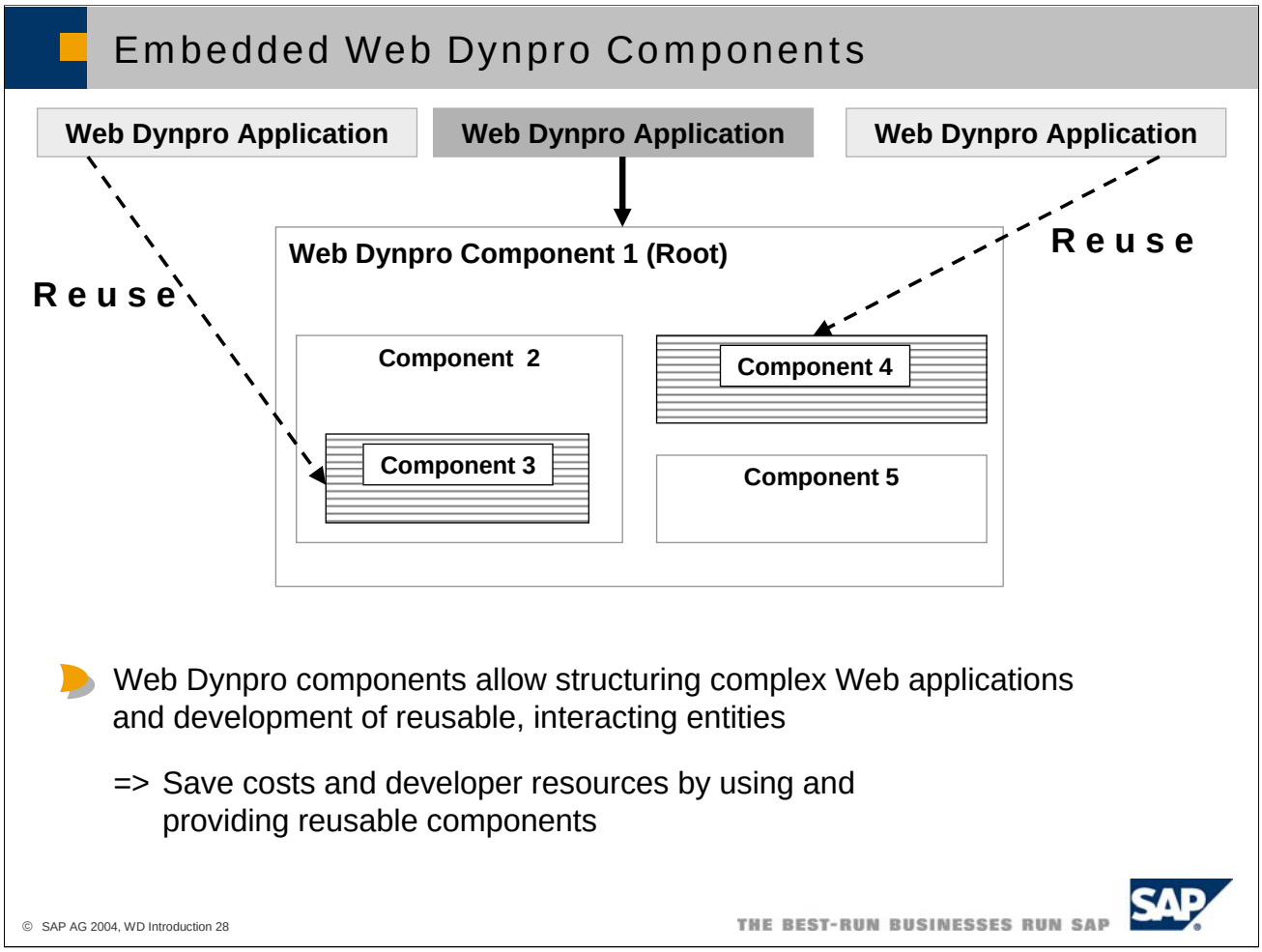
A component's interface view can be used like a normal view in the embedding component's navigation graph

A component can define multiple interface views with different navigations

■ **The component interface**

The component interface consists of two visual parts and one programmatic part.

- The programmatic interface allows an embedding component to interact with an embedded component by calling methods and reacting to events. Currently, the programmatic interface of a component consists of the interface controller. With 6.40, there will also be a configuration controller
- The visual interface allows you to embed the component windows as views in the embedding component. To achieve this, the window has a one-to-one association with an interface view in the component interface. The visual interface of a component is optional.



■ **Embedded Web Dynpro components**

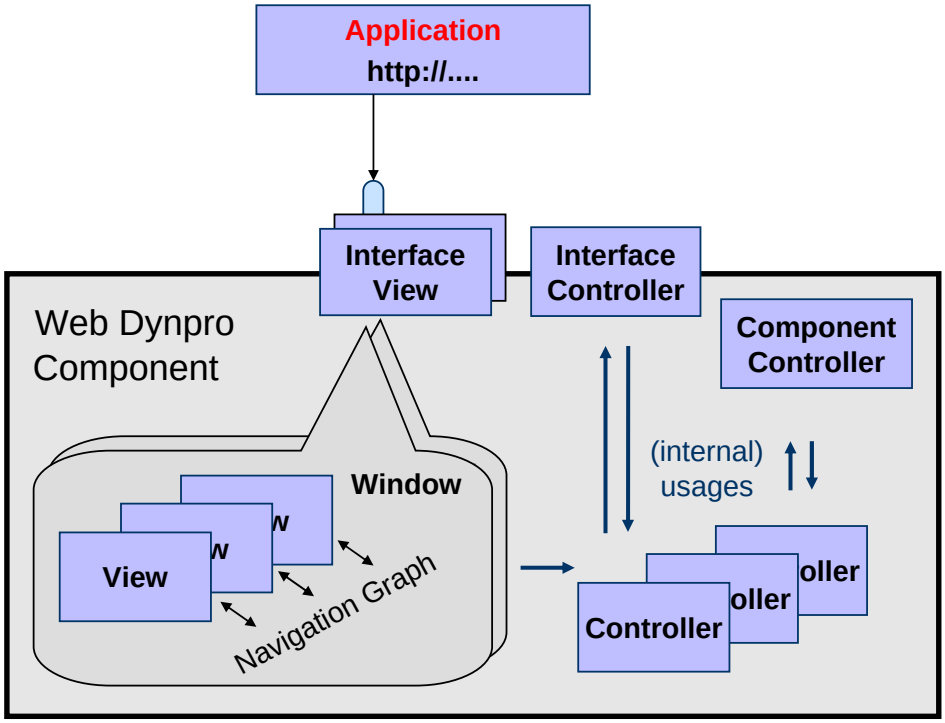
A Web Dynpro component is a reusable entity. It contains all basic Web Dynpro application elements that are needed for a Web Dynpro application, plus the Web Dynpro model and the startup entity, the Web Dynpro application.

You have the option of nesting Web Dynpro components. This is referred to as **embedding**. Embedding Web Dynpro components offers the following advantages:

- The Web Dynpro application is divided into interacting elements.
- The structure of a large Web Dynpro application is clearer.
- Working with reusable elements saves costs and development resources.
- At design time, the embedding Web Dynpro component has no knowledge of the internal structure of the embedded Web Dynpro component; it does not become visible to the embedded Web Dynpro component until runtime.

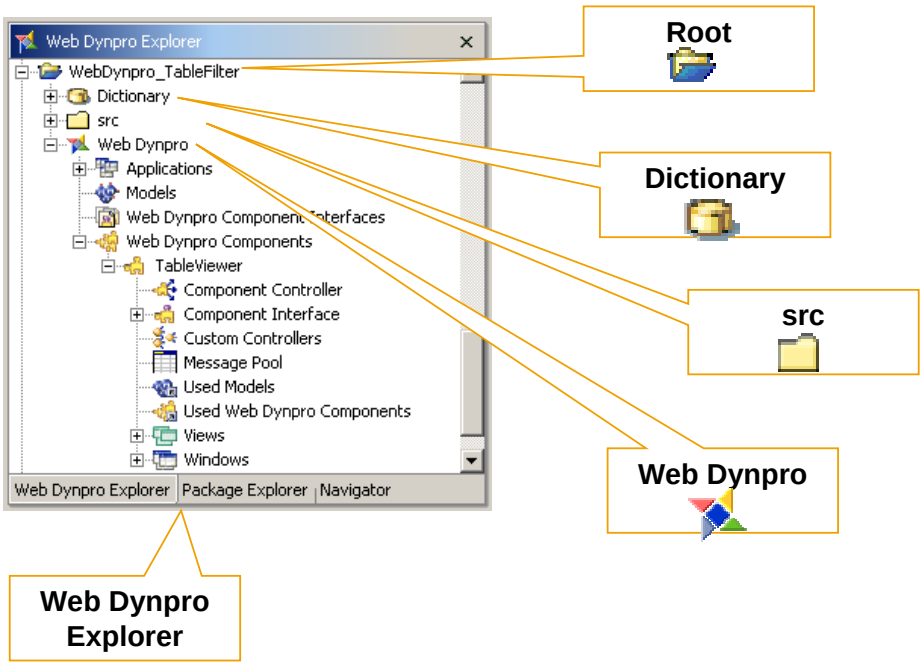
The interaction between nested Web Dynpro components is implemented using method calls that originate from the embedding Web Dynpro component. The Web Dynpro component interface starts these method calls. When nesting Web Dynpro components, this interface function is obligatory and must be implemented in the program logic. However, an optional visual interface is also available

Web Dynpro Application



- **Application**
  An application is like a transaction code in ABAP. It is an entry point to a Web Dynpro component (and its embedded components and used models), addressable via URI.
  - To define an application, the following entities must be named:
  - The root component to be invoked
  - The interface view of the root component (and, therefore, the window to be called)
  - The inbound plug of the specified interface view

Basic Terminology



- **Basic Terms**

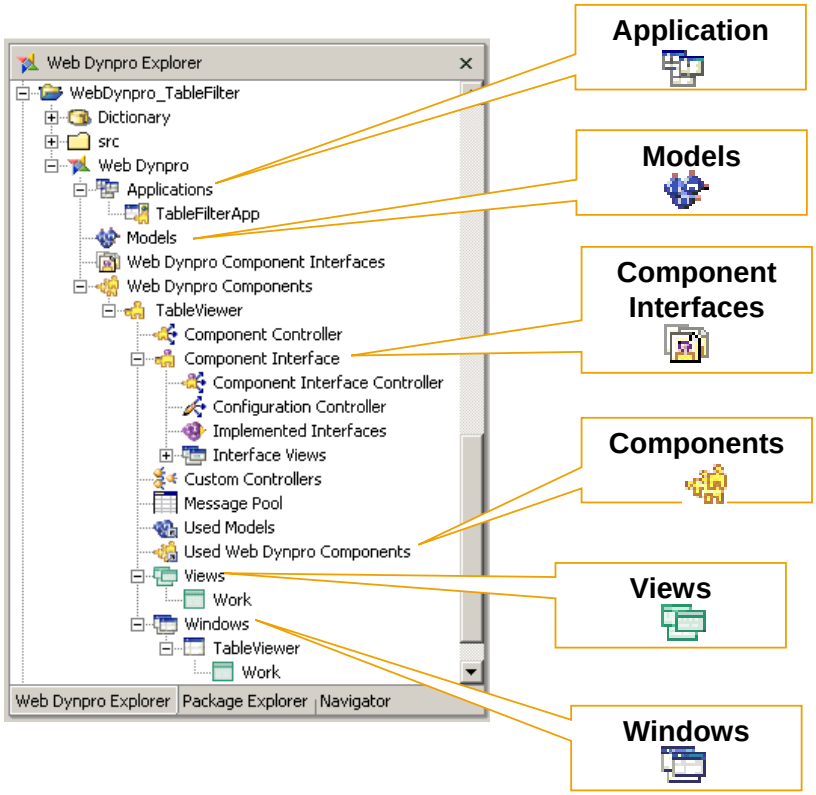
  Root, Web Dynpro project.
  Deployable unit containing on or more Web Dynpro applications
  Dictionary
  Static dictionary definitions of simple (scalar) data types.
  Src, Source tree (references only)

  - Mimes
  - Configuration
  - Packages (metadata, not- generated java-sources)

  Web Dynpro, Main Item of a Web Dynpro project

  - Web Dynpro applications
  - Models
  - Web Dynpro component interfaces
  - Web Dynpro components

Basic Terminology (2)



- **Basic Terms**

  Application

  Entry point to a Web Dynpro component (and its embedded components and used Models) addressable via URI

  Models

  Layer of code encapsulating business functionality external to the Web Dynpro component. Provides access to functionality such as BAPI calls

  Component Interfaces

  The programming interface allowing the embedding code to interact with a Component. Comprises a visual (Interface Views) and a programmatic (controller and context) part

  Components

  Can be called by applications or embedded into other components. Not addressable via URL

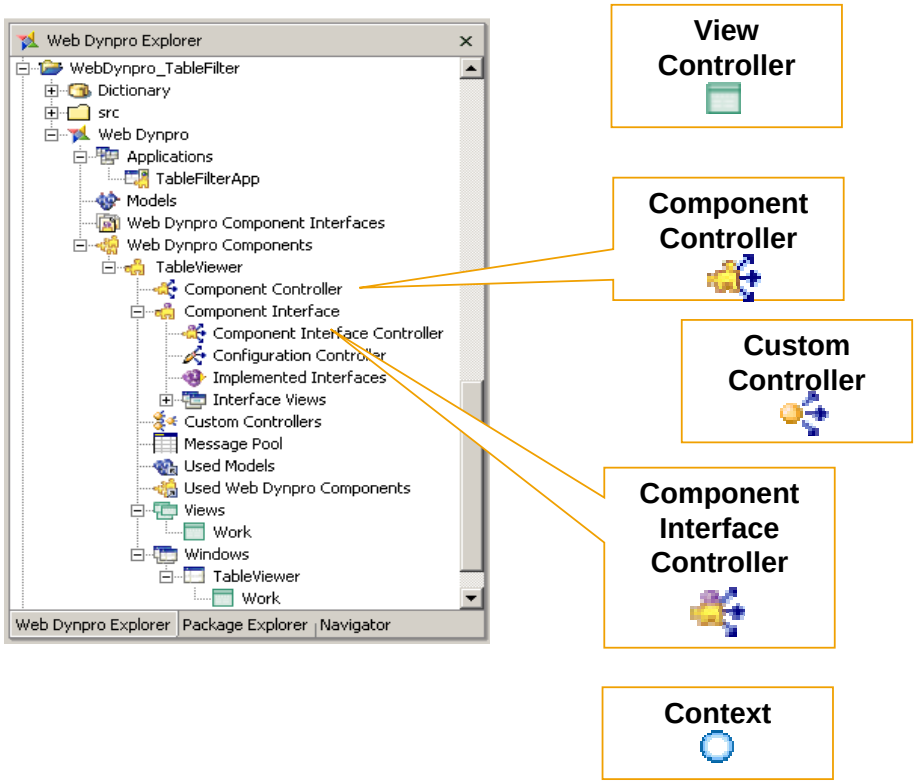
  Views

  Smallest units for representing UI element-sets used to design UI compositions

  Windows

  Single UI unit representing a component's User Interface/view composition needed for embedding purposes (e.g. application uses root component, other View or component embeds component)

Basic Terminology (3)



- **Basic Terms**

  View Controller

  View controllers belong to views and live exactly as long as the views are displayed.

  Component Controller

  Special (default) custom controller which always exists. Custom controllers have to be named and declared.

  Custom Controller

  Needed for storing context data used across several views. A Custom controller has a longer lifecycle than a View controller. Custom controllers belong to the component and live as long as the component lives in an application.

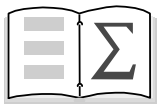
  Component Interface Controller

  Programming interface allowing the embedding code to interact with a component. The component interface comprises a visual (*interface views*) and a programmatic (controller and context) part.

  Context

  Structured storage for the controller

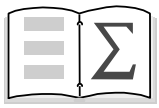
Features: Topic Summary



You should now be able to:

- **Understand the basic concepts of Web Dynpro.**

Web Dynpro Dynamic Programming: Summary



You should now be able to:

- **Understand the basic idea behind Web Dynpro.**
- **Explain basic Features of Web Dynpro.**
- **Understand basic concepts of Web Dynpro.**