

SAP Web Dynpro

Erstellen webbasierter
Benutzeroberflächen



- **Einführung**
- **Architektur**
- **Voraussetzungen**
- **Einsatzszenarien**
- **WDA oder WD4J?**
- **Live-Entwicklung Web Dynpro for Java**
- **Live-Entwicklung Web Dynpro for ABAP**
- **Diskussion**



- **Einführung**

- Architektur

- Voraussetzungen

- Einsatzszenarien

- WDA oder WD4J?

- Live-Entwicklung Web Dynpro for Java

- Live-Entwicklung Web Dynpro for ABAP

- Diskussion



Was ist Web Dynpro?

Programmiermodell für Benutzeroberflächen

- Plattformunabhängig (Java, ABAP ...)

Definition einer Standard-Struktur für UI-Anwendungen

- Umsetzung des MVC (“Model-View-Controller”) Design Patterns

Werkzeug für Gestaltung von Benutzeroberflächen

- Fokus auf graphischer Modellierung
 - ◆ Generierung von Quelltexten aus Meta-Model-Deklarationen
- Integration in Entwicklungsumgebungen
 - ◆ SAP NetWeaver Developer Studio
 - ◆ SE80

Laufzeitumgebung für Anwendungen

- Serverseitiges Framework bietet allgemeine Dienste an
- Clientseitige Technologie für browserbasierte Benutzeroberflächen
 - ◆ XML-basiertes Protokoll erlaubt verschiedene Clients

Technologie zur Software-Modularisierung

- Komponenten erlauben
 - ◆ Strukturierung von Projekten und
 - ◆ Pattern-basierte UIs

Entwicklung von Web Anwendungen in Enterprise-Qualität

- **Minimales Codieren, maximales Design**
- **Trennung von Layout und Logik**
- **Unterstützung für**
 - ◆ verschiedene Backends
 - ◆ Wiederverwendung von Komponenten
 - ◆ Web Services & Data-Binding
- **Konfiguration von UI Patterns**

Plattformunabhängigkeit

- **serverseitig durch Unterstützung verschiedener Laufzeitsysteme (Java, ABAP ...)**
- **clientseitig durch**
 - ◆ Java Script-Anwendungskomponenten mit
 - ◆ Lauffähigkeit in verschiedenen Standard-Browsern (IE, Firefox)

Verbesserte Benutzerfreundlichkeit

- **Bildschirmaktualisierung ohne neuerlichen Seitenaufbau**
- **dynamische Darstellung clientseitig**
- **Performanz durch Caching**
- **508 Accessibility Unterstützung**
- **flackerfreie Bildschirme, wenige Refreshes**



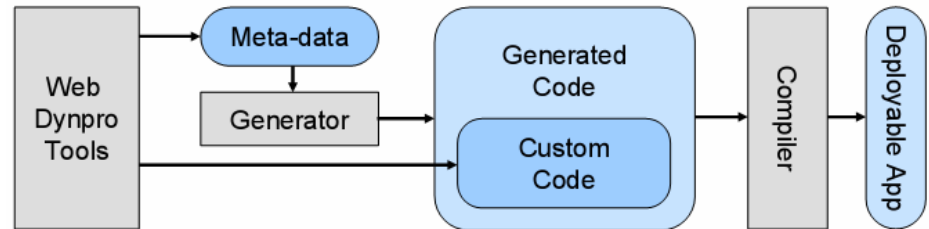
Deklaratives Programmiermodell

Beschreiben der Eigenschaften einer Web Dynpro Applikation

- in Form von Metadaten
- mittels Web Dynpro Toolset

Quellcode wird

- automatisch generiert,
- kompiliert und
- zur Laufzeit ausgeführt



Möglichkeit zur Platzierung eigenen Codes in vorgesehenen Bereichen

UIs bestehen aus

- gleichen Basis- sowie anwendungsspezifischen Elementen, die
 - ◆ statisch deklariert oder
 - ◆ programmatisch implementiert und zur Laufzeit integriert werden können

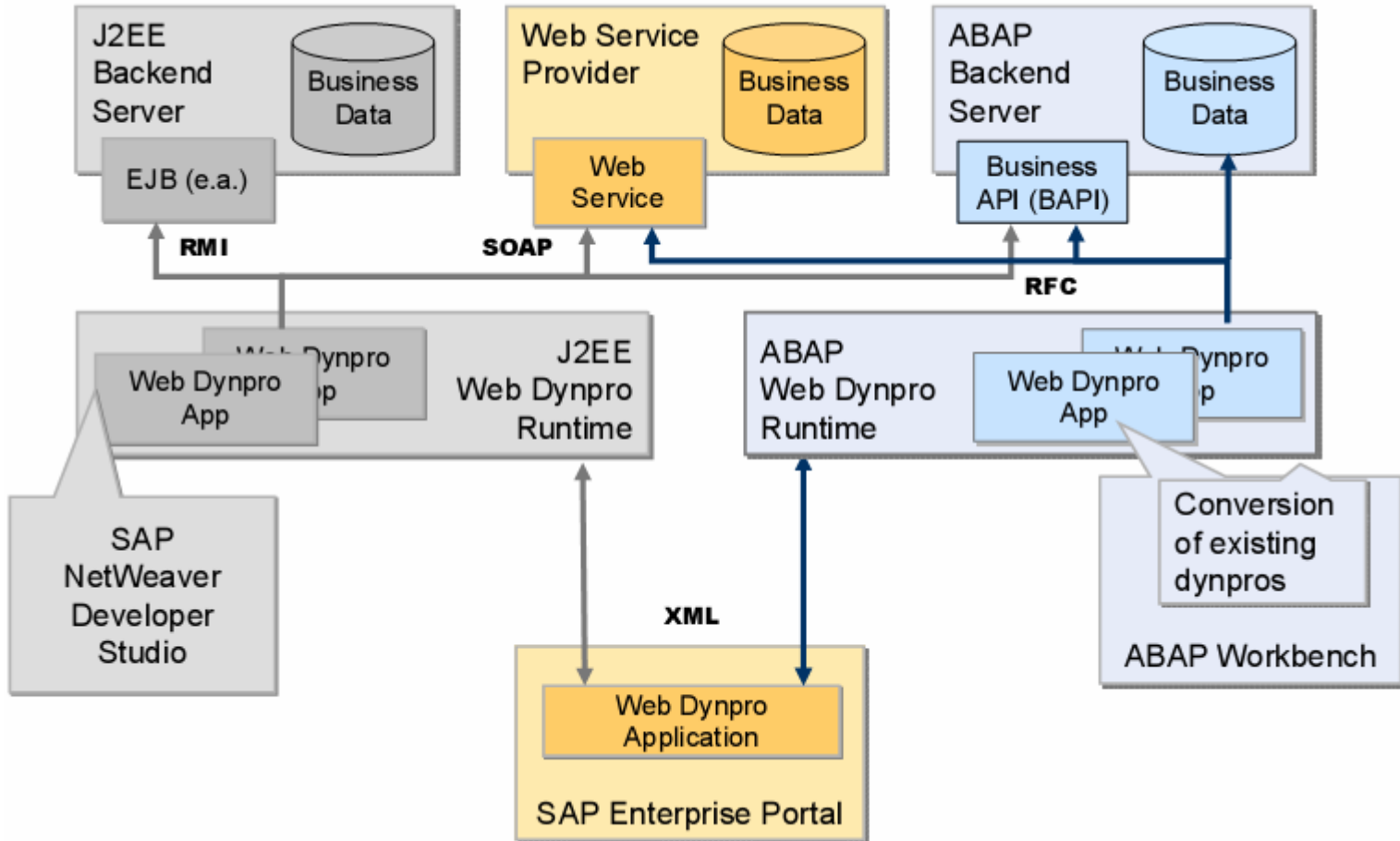
Programmatische Implementierung ermöglicht

- Erweiterung deklarativer UIs mit neuen Interface-Strukturen zur Laufzeit und damit
- Kombination deklarativer und programmatischer Techniken

Web Dynpro Metadaten liegen plattformunabhängig vor

- Anwendungen können von einer Plattform auf eine andere transportiert werden
 - ◆ Quellcode wird aus Metadaten neu generiert bzw.
 - ◆ eigenes Coding entsprechend adaptiert (einheitliche API)

Anwendungsszenarien mit Web Dynpro



Ebenen

- Controls
 - ◆ atomare Layout-Elemente, bestimmen Look & Feel der Anwendung
- Components
 - ◆ wieder verwendbare, aufgabenorientierte Baugruppen
- Floor Plans
 - ◆ Bildschirmgestaltung, Interaktivität und Semantik generischer Anwendungen

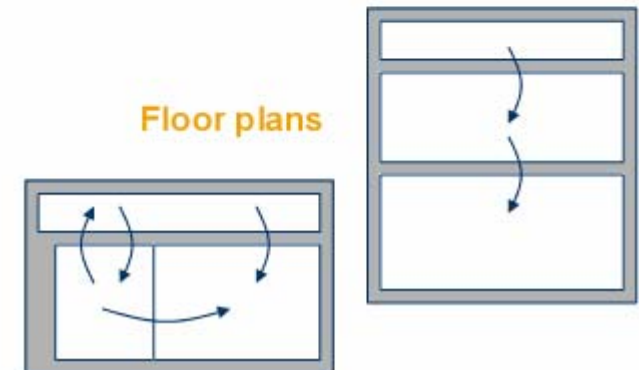
Konsistente Benutzeroberflächen

- geringer Einarbeitungsaufwand
- wenig Spezialkenntnisse erforderlich



Components

Sales	Sales Team	Goals	Products
Detail	New Row	Run Filter	
Item	Product Category	Product	
10	MAT_HAWA	DH_PRODD01	
20	MAT_HAWA	DH_PRODD01	
30	MAT_HAWA	DH_PRODD01	



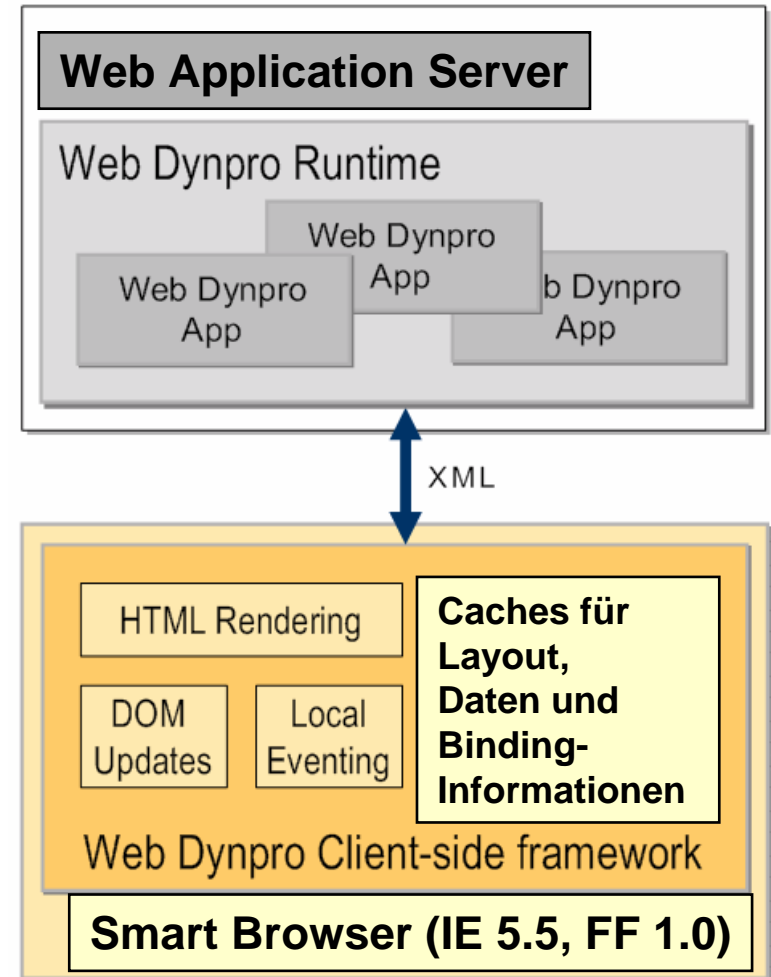
Client-seitiges Framework (CSF)

- umfangreiche Bibliothek von Benutzeroberflächenelementen
- Zero Footprint
 - ◆ Java Script-Bibliothek < 100 KB
 - ◆ Lauffähig auf IE >= 5.0 und Firefox >= 1.0
- Integration mit externen Komponenten
 - ◆ Microsoft Office
 - ◆ Adobe Forms

Performance-optimiertes Client-Server-Protokoll

- Laden von Tabellendaten
- Delta-Transfer
 - ◆ Layout-Informationen vom Server zum Client
 - ◆ Anwendungsdaten in beide Richtungen

508 Accessibility Features



Umfangreiche Bibliothek an UI Controls

- entspricht Unified Rendering & Unified UI Elements Standard

Deklaratives Screen-Management

- Navigationsgraphen
- Verschachtelte Views & Popup-Fenster

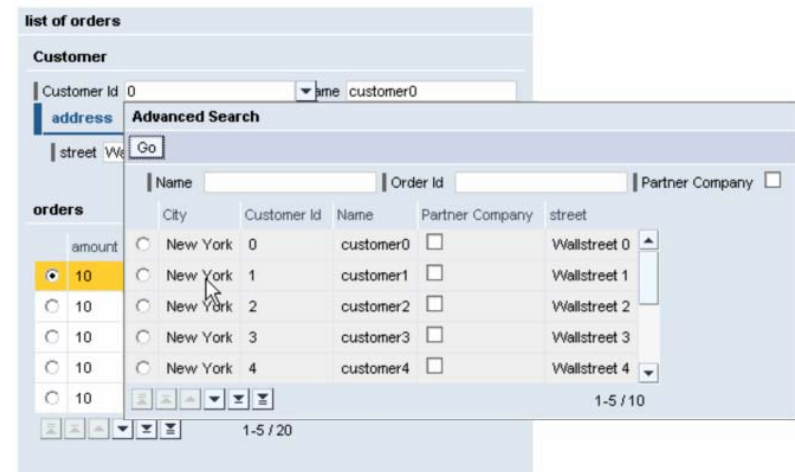
Layout-Varianten

- Grid-, Flow-, Row- & Matrix-Layout
- Verschachtelungsmöglichkeit

Generische UI Services basierend auf Metadaten

- Extended Value Selector ('F4')
 - ◆ Metadaten für Werteselektion
 - aus Dictionary oder
 - dynamisch definiert
- automatische
 - ◆ Konvertierung,
 - ◆ Prüfungen und
 - ◆ Fehlerbehandlungfür elementare Datentypen
- Umfangreiches Fehlerbehandlungskonzept

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



Demo!

Dynamische Erzeugung & Modifikation des Meta-Modells

- Views & Layout-Elemente
- Context-Elemente & Datentypen

Komponentenmodell für Kapselung & Wiederverwendbarkeit

- Standalone-Component Interfaces
- dynamische Mehrfachinstanzierung eingebetteter Komponenten

APIs für Zugriff auf Server-Interfaces

- Zugriff auf
 - ◆ System Landscape Directory
 - ◆ URL-Parameter
- Setzen des Session Timeout

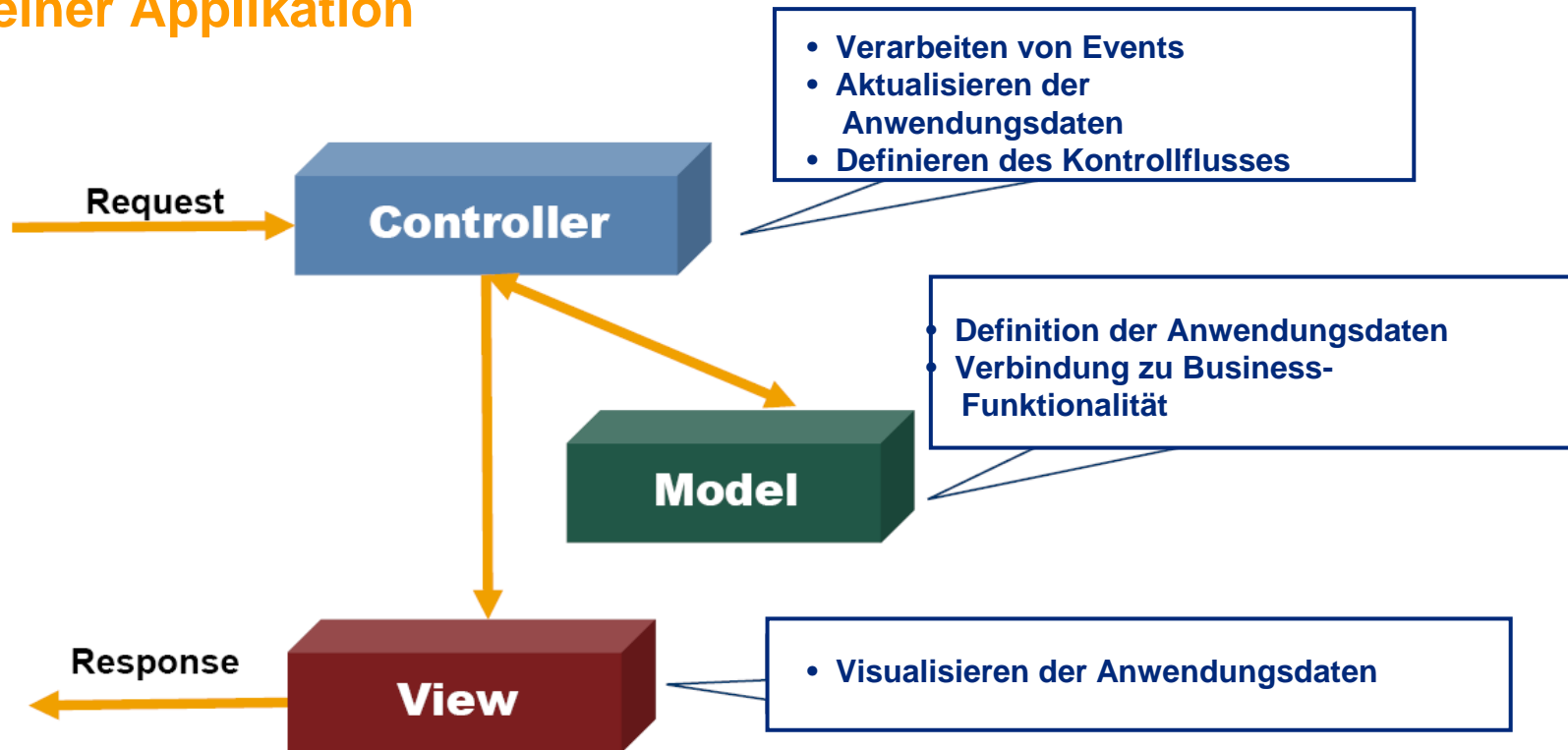
Integration

- UME
- Portal
 - ◆ clientseitige Portal-Events
 - ◆ Übernahme von Themes & Stylesheets

- Einführung
- **Architektur**
- Voraussetzungen
- Einsatzszenarien
- WDA oder WD4J?
- Live-Entwicklung Web Dynpro for Java
- Live-Entwicklung Web Dynpro for ABAP
- Diskussion

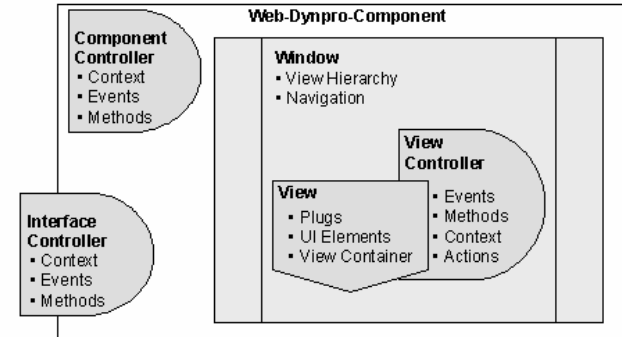


Design Pattern für Entkoppelung von Präsentation und Logik einer Applikation



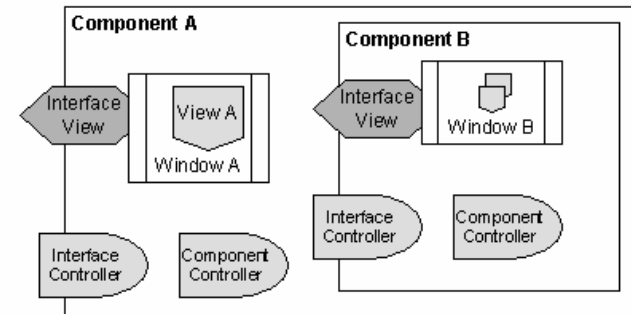
Definition

- wieder verwendbare Einheit
- umfasst alle Bestandteile, die
 - ◆ im Rahmen dieser Programmier-Einheit
 - ◆ für lauffähige Web-Dynpro-Anwendung benötigt werden
- enthält beliebige Anzahl von
 - ◆ Windows und
 - ◆ Viewsmit zugehörigen Controllern
- Referenzierung weiterer Web-Dynpro-Components möglich
- Lebensdauer
 - ◆ beginnt mit erstem Aufruf zur Laufzeit &
 - ◆ endet mit der Laufzeit



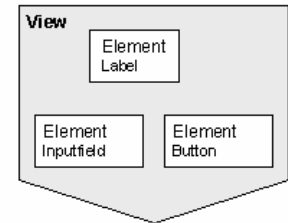
Vorteile

- Strukturieren der Programmierung
- Bilden überschaubarer Anwendungsblöcken
- Wiederverwendbarkeit ganzer Components
- Zeitliche und räumliche Entkopplung von Software-Projekten



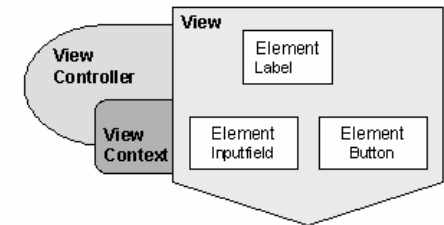
Definition

- beschreibt
 - ◆ Layout und
 - ◆ Verhalten
- rechteckigen Bereichs einer Benutzungsoberfläche









Verwendung

- Web Dynpro-Anwendung besitzt mindestens einen View
- Layout setzt sich aus ineinander verschachtelbaren Oberflächenelementen zusammen
- Positionierung von Oberflächenelementen in View durch Layoutvarianten
- View enthält
 - ◆ Layout (sichtbarer Teil)
 - ◆ Controller
 - ◆ Context (Ablage & Verwaltung von Bildschirmdaten)



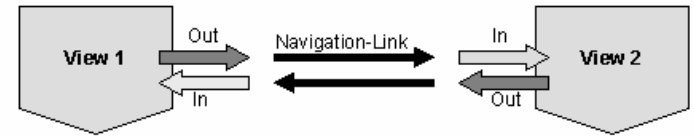
View-Set

- nur in Web Dynpro für Java
- optischer Rahmen
 - ◆ mit vordefinierten Teilbereichen
 - ◆ zur Einbettung von Views
 - ◆ zur Design-Zeit
- Vorteile
 - ◆ Strukturierte Anzeigemöglichkeit mehr als eines View in Bildschirmausgabe
 - ◆ Unterstützung beim Entwurf der Benutzeroberfläche
 - ◆ Einfache nachträgliche Layout-Änderungen durch Nutzen vordefinierter Bereiche
 - ◆ Wiederverwendung von Views innert Web-Dynpro-Window

T-Layout	T-Layout 90°	T-Layout 180°	T-Layout 270°	Grid- Layout	Tab-Strip
					

Plugs

- Unterscheidung:
 - ◆ Inbound-Plugs definieren mögliche Einstiegsstellen eines View
 - ◆ Outbound-Plugs ermöglichen Aufruf Folge-Views
- Bestandteil des Controllers eines View
- Betreten eines View über Inbound-Plug löst Aufruf einer Ereignisbehandlungsmethode aus
 - ◆ zu jedem Inbound-Plug automatisch generiert
 - ◆ Nutzung optional
- Default-View
 - ◆ wird bei Aufruf eines Window als erstes angezeigt & nicht über Inbound-Plug aufgerufen
 - ◆ weiterführende Navigation wird darauf aufgebaut

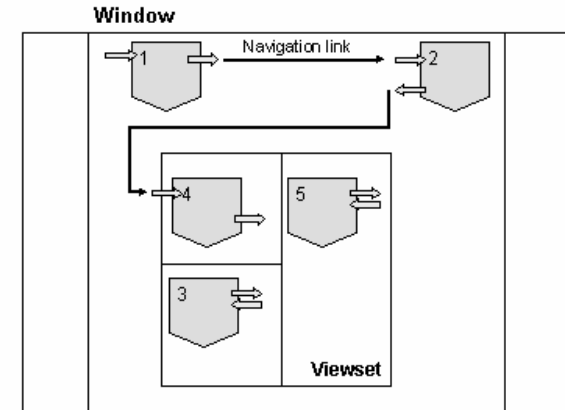


Navigations-Link

- implementiert Navigation zwischen zwei Views durch statisches Verbinden
 - ◆ Outbound-Plug des ersten mit
 - ◆ Inbound-Plug des zweiten View
- Outbound-Plug kann über einen Navigations-Link zu mehreren Ziel-Views führen
- Inbound-Plug kann von mehreren Outbound-Plugs angesteuert werden
- Informationen über Verbindung zwischen Plugs wird in Navigation separat gespeichert

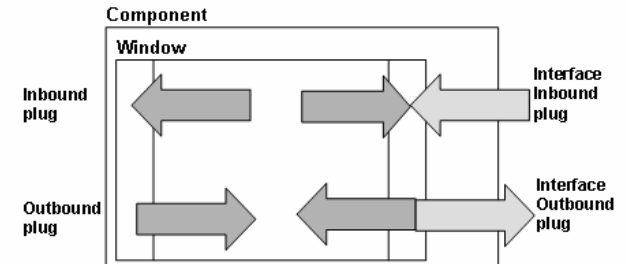
Definition

- bündelt mehrere
 - ◆ Views und
 - ◆ View-Sets
- View kann nur angezeigt werden, wenn in Window eingebettet
- Window enthält immer
 - ◆ einen / mehrere Views, die
 - ◆ untereinander über Navigations-Links verbunden sind
- Startview wird bei erstem Aufruf des Window angezeigt



Plugs

- Window verfügt über einen / mehrere In- bzw. Outbound-Plugs zur Einbettung in Navigationskette

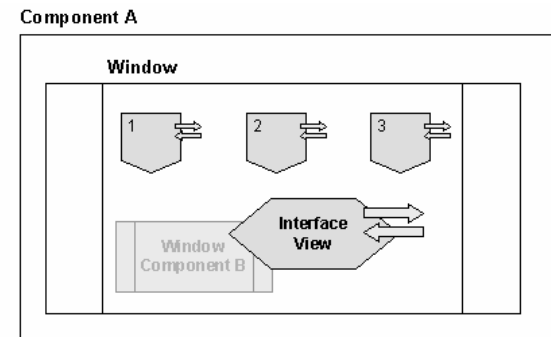
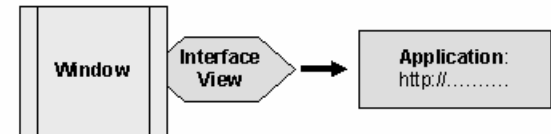


Windows-Controller

- globaler Controller, der
 - ◆ jedem Web Dynpro-Window zugeordnet und
 - ◆ für alle anderen Controller innerhalb der Component sichtbar ist
- nur in WDA verfügbar

Interface-View

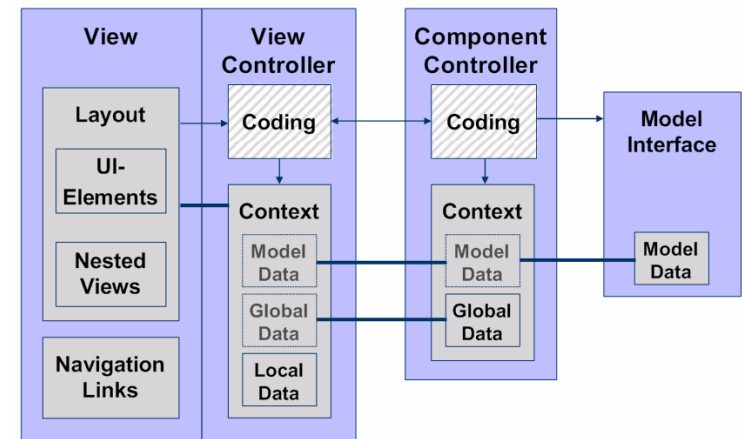
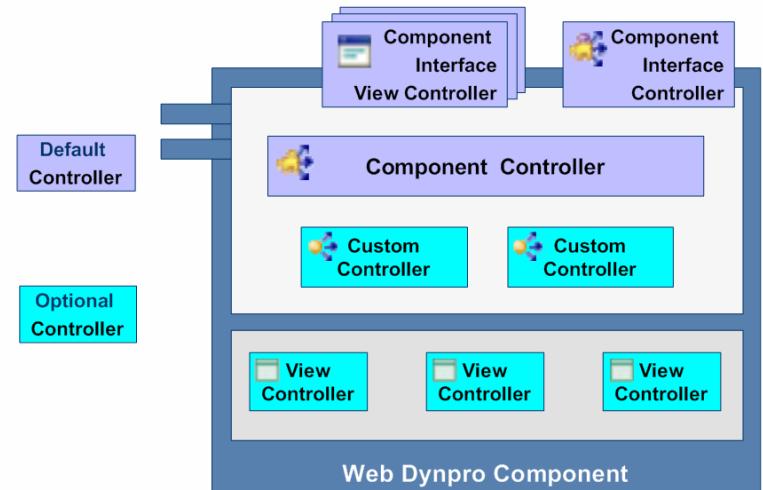
- jedem Window ist ein Interface-View zugeordnet, der
 - ◆ die Außensicht des Windows darstellt
 - ◆ mit Web-Dynpro-Anwendung verknüpft wird, sodass Window über URL aufrufbar wird
- erlaubt Component-übergreifende Wiederverwendung von Windows
 - ◆ in Window können neben Component-eigenen Views auch
 - ◆ Interface-Views der Windows aller Components eingebettet werden, die
 - der aktuellen Component über
 - Component-Verwendungbekanntgegeben wurden
- verfügt über
 - ◆ Inbound-Plugs und
 - ◆ Outbound-Plugs undkann über diese in Navigationsstruktur innert anderen Windows eingebunden werden
- jeder View kann nur einmal im aktuellen Window angezeigt werden



Demo!

Definition

- aktive Teile einer Web-Dynpro-Anwendung
- legen fest, wie Benutzer mit Web-Dynpro-Anwendung agieren kann
- zugreifbare Daten eines Controller sind im zugehörigen Context definiert
- Web-Dynpro-Anwendung besitzt verschiedene Controller & Contexte
 - ◆ View-Controllern steuern Verhalten einzelnen Views
 - ◆ globale Controller bieten allgemeinere Services für alle Views einer Component an

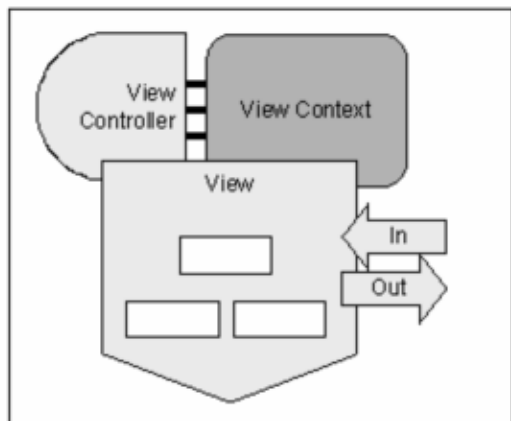
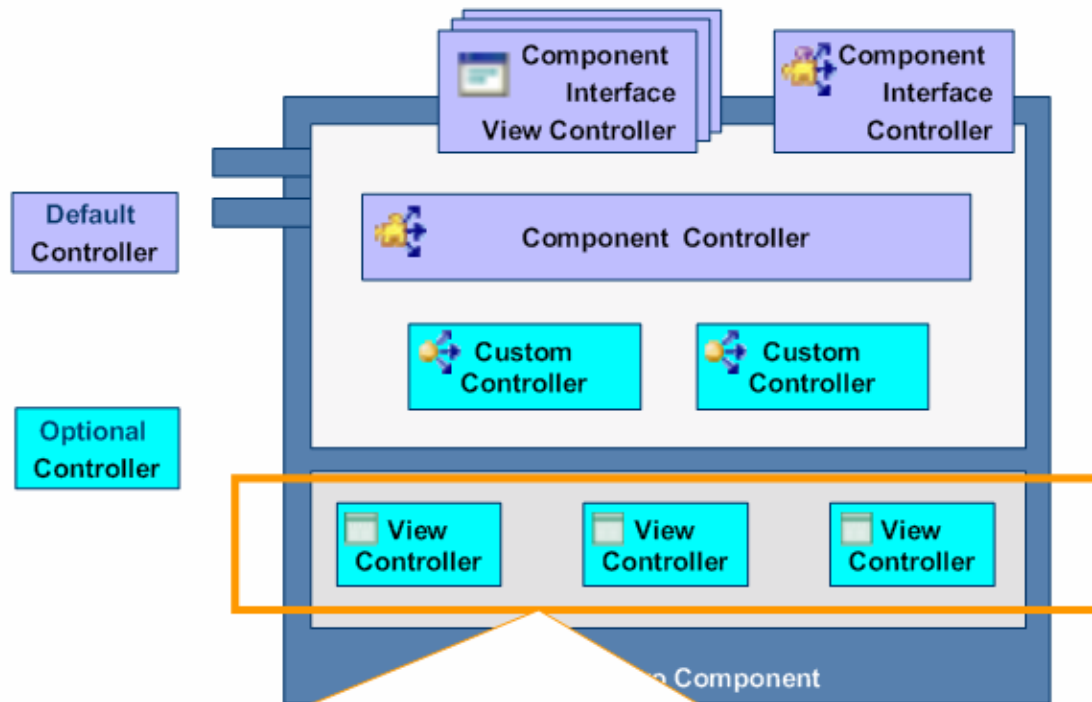


Quelltext

- jeder Controller besitzt Bereiche zum Einfügen von Quelltext
 - ◆ Ereignisbehandler: ausgeführt bei
 - Initialisierung, Betreten & Beenden eines View
 - Auslösen einer Aktion durch ein Oberflächenelement eines View
 - Auslösen eines ein registriertes Ereignis auslösen durch anderen Controller
 - ◆ Methoden: von anderen Controllern aufrufbar
 - ◆ Supply-Funktionen: nach Bedarf zur Initialisierung von Context-Elementen ausgeführt
- Application Programming Interface (APIs) für
 - ◆ Bearbeitung von Context-Knoten, ihrer Attribute und Daten
 - ◆ Zugriff auf Systemumgebung (API der Server-Abstraktions-Schicht)
 - ◆ Nachrichtenbehandlung
 - ◆ dynamische Programmierung

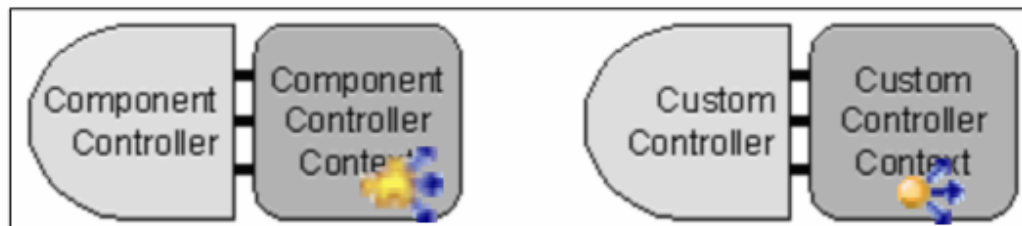
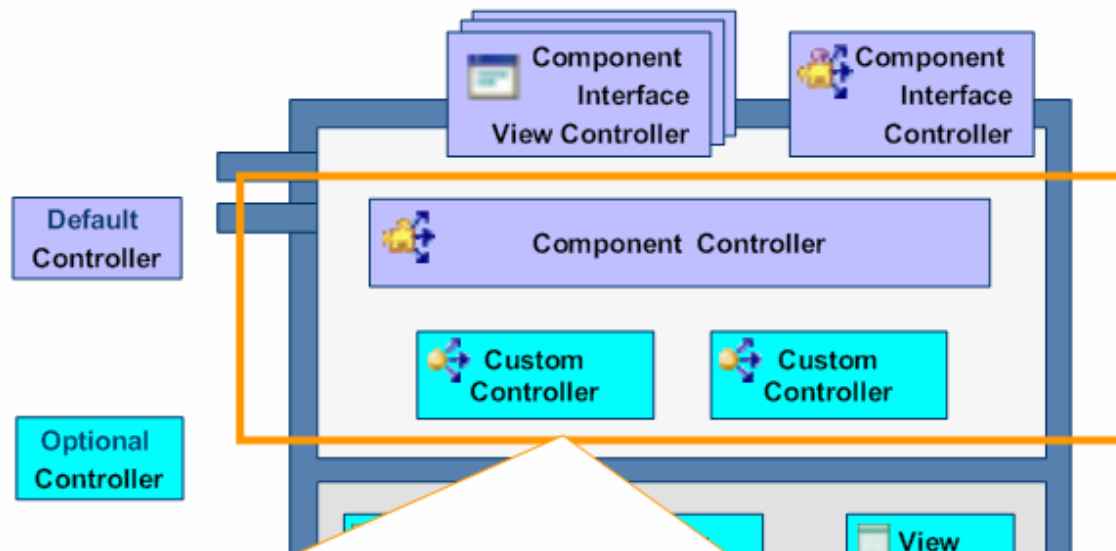
Demo!

View Controller



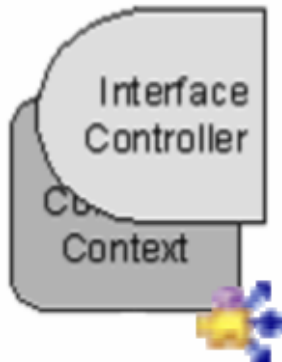
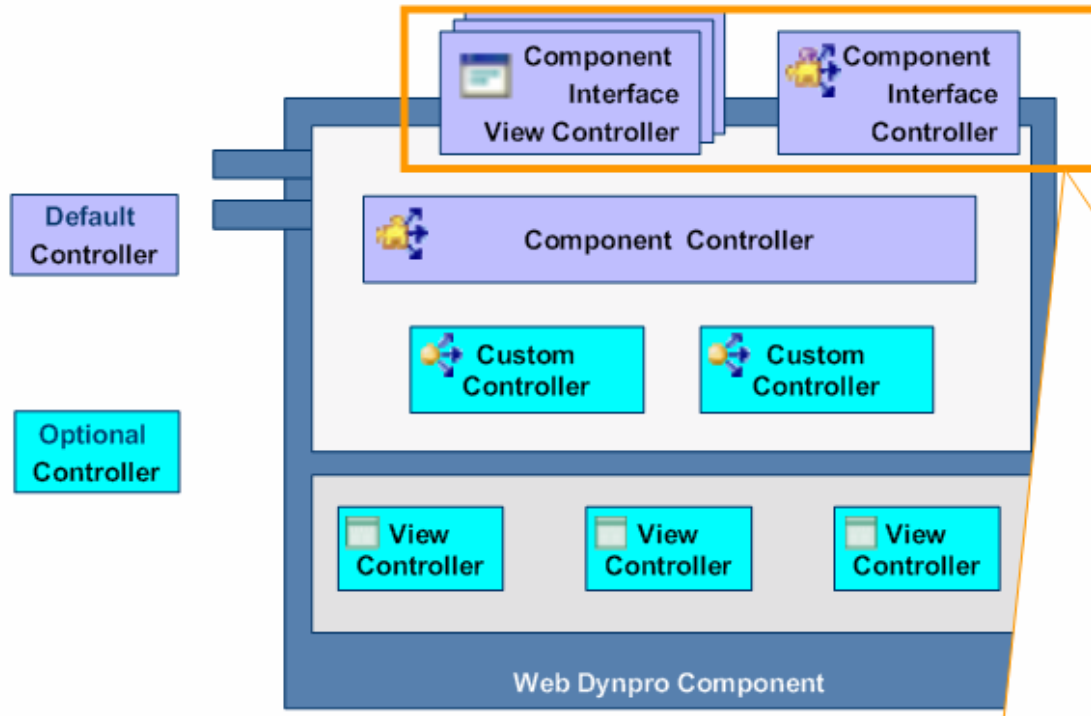
- Jeder View besitzt einen View-Controller, der Benutzeraktionen verarbeitet
- Jeder View besitzt einen View-Context zum Vorhalten benötigter Daten
- View-Controller & Context leben mindestens so lange, wie View im Browser sichtbar ist

globale Controller



- Jede Component besitzt mind. einen innert der Component für alle anderen Controller sichtbaren globalen Controller
Lebensdauer erstreckt sich auf Nutzungsdauer jeweiliger Component
- Weitere globale Controller können als Custom-Controllern hinzugefügt werden

Interface Controller



- Jede Component besitzt einen genannten Interface-Controller, der global und zusätzlich auch außerhalb der Component sichtbar und damit Teil der Schnittstelle ist.
- Kommunikation von einem Controller zum nächsten erfolgt durch Aufruf von Methoden des Controllers oder Auslösen eines Ereignisses, auf das sich andere Controller registriert haben.

Definition

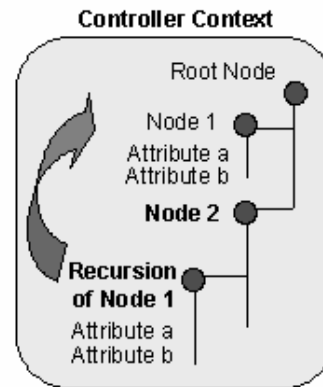
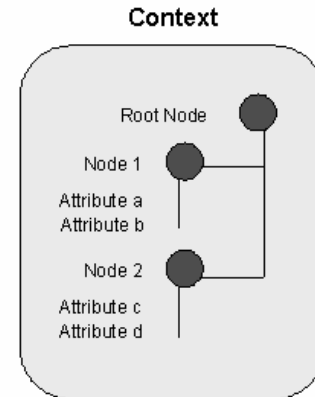
- Ablage der in Component / View verwendeten Daten
- Lese- & Schreibzugriff von zugehörigen Controllern aus

Aufbau

- Verwaltung der Contextdaten in hierarchischer Struktur
- Jeder Context besitzt
 - ◆ einen Wurzelknoten
 - ◆ unterhalb dessen die einzelnen Datenfelder (Attribute)
 - ◆ in Baumstruktur abgelegt werden
- Anlage der Baumstruktur entsprechend der Struktur der Anwendung
- Jeder Knoten enthält Datenfelder, die
 - ◆ Einzelinstanz eines Objekttyps oder
 - ◆ Tabelle von Instanzendarstellen (Kardinalität eines Knotens)

Rekursionsknoten

- dynamische Verschachtelung von Knoten durch Anlage von Rekursionsknoten möglich
 - ◆ zur Rekursion verwendeter Knoten ist immer Vorgänger neuen Knotens
 - ◆ neu angelegter Rekursionsknoten ist Referenz auf Vorgängerknoten
 - kann nicht separat bearbeitet werden
 - übernimmt Struktur zu wiederholenden Knotens



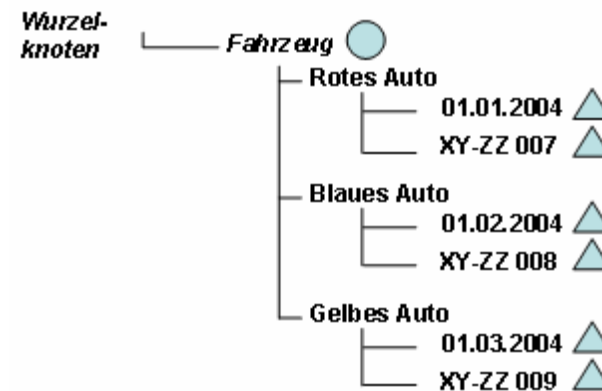
Kardinalität

- bei Anlage eines Knoten bekanntgegeben
- legt fest, wie oft Knoten zur Laufzeit instanziiert wird (wie viele Elemente verfügbar sind):
 - ◆ **1...1**: Instanzieren **genau eines** Elements
 - ◆ **0...1**: Instanzieren **maximal eines** Elements
 - ◆ **1...n**: Instanzieren **von n** Elementen, mindestens jedoch **einem**
 - ◆ **0...n**: Anzahl instanzierter Elemente kann beliebig variieren
- Beispiel:

Design-Zeit

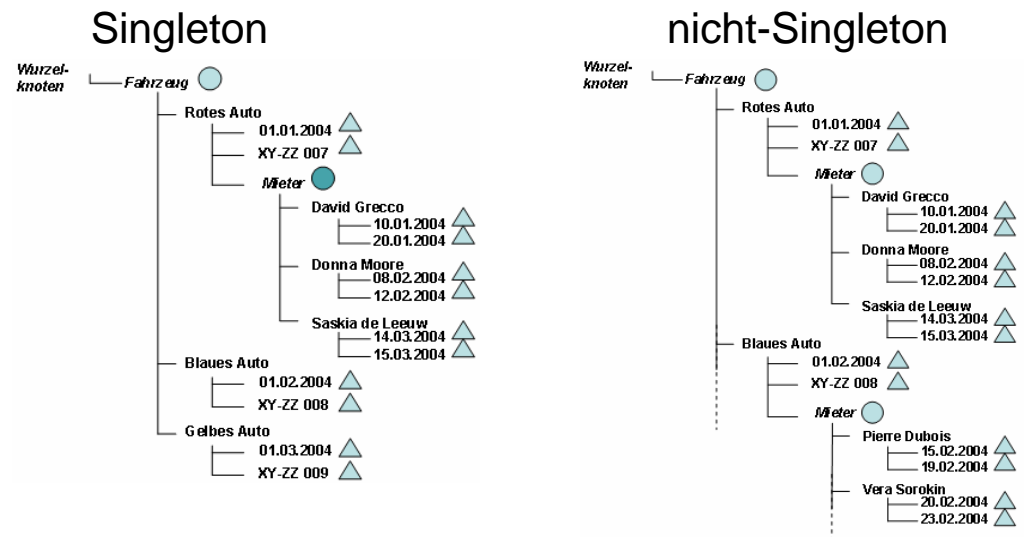
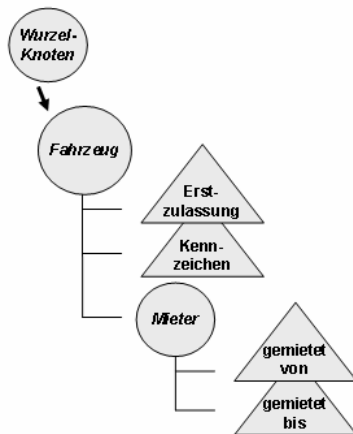


Laufzeit



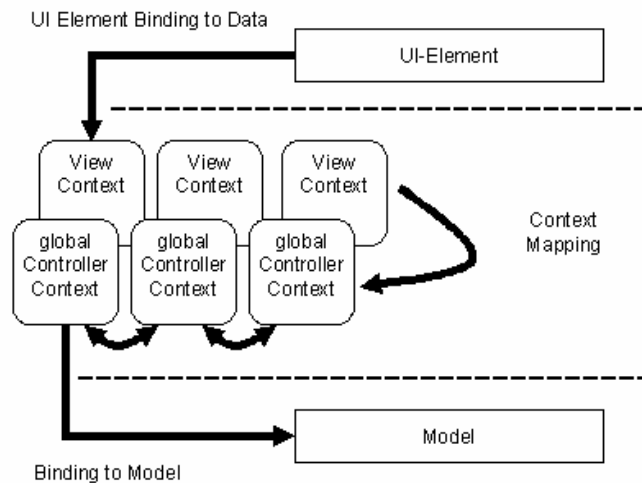
Singleton

- leg fest, ob Kind-Elemente
 - ◆ für alle Elemente des Eltern-Knotens („nicht-Singleton“) oder
 - ◆ nur für genau ein Element des Eltern-Knotens („Singleton“)aufgebaut werden
- Lead-Selection bestimmt, zu welchem Eltern-Knoten ein Singleton Elemente hält
- Beispiel **Demo!**
Design-Zeit Laufzeit



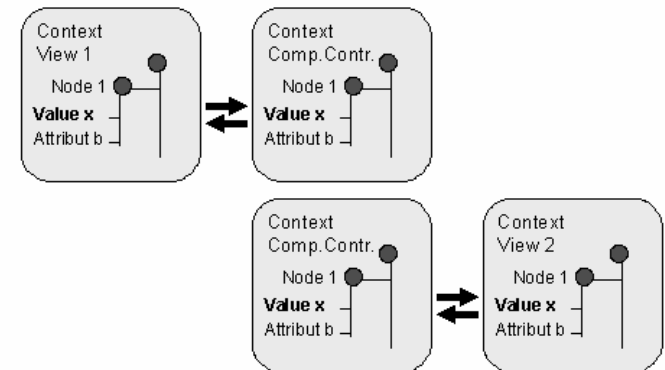
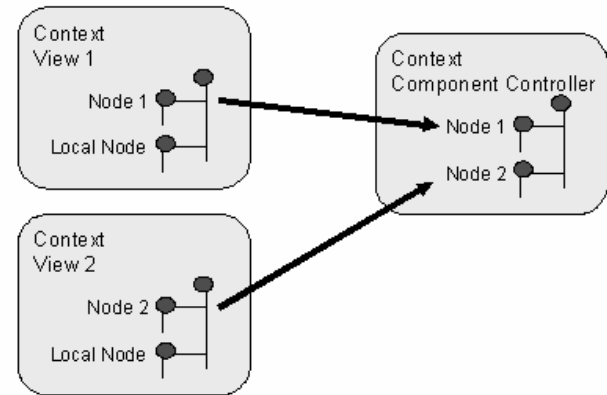
Datenbindung & Mapping

- Möglichkeiten zur Verknüpfung der Contexte verschiedener Controller:
 - ◆ UI-Element des View kann an Element eines View-Contexts gebunden werden
 - ◆ Mapping kann zwischen
 - zwei Contexten globaler Controller bzw.
 - von einem View-Context zu einem globalen Controller-Context definiert werden.
 - ◆ Context globalen Controllers kann an ein Web-Dynpro-Model gebunden werden



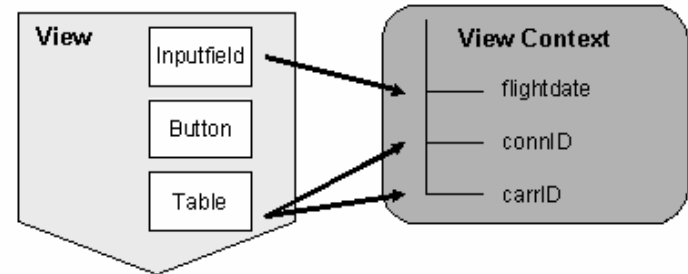
Mapping zwischen Contexten

- Elemente eines View-Contexts können
 - ◆ lokal deklariert werden oder
 - ◆ mit einem Knoten eines anderen Contextes verknüpft (Mapping) werden
- Eigenschaften lokale Context-Elemente
 - ◆ alle enthaltenen Attribute nur innert betreffendem View sichtbar
 - ◆ mit Verschwinden des View werden Attributswerte gelöscht
- Mögliche Mappings:
 - ◆ globalen Controller-Contexte untereinander
 - ◆ View-Context-Elementes auf ein Element globalen Contextes
 - ◆ Mapping von globalem Controller auf Element eines View-Contextes aufgrund kürzerer Lebensdauer letzteren nicht möglich
- Verwendung von Mappings für Datentransport innert Component über verschiedene Contexte hinweg
 - ◆ Einlesen Datenwerts über Eingabefeld des ersten View
 - ◆ Weiterreichen über geeignetes Mapping an den Context des Component-Controllers und Folgeview
 - ◆ wurde Wert von geändert, findet Folgeview aktuellen Wert in Context vor



Bindung Oberflächenelement an Context-Attribut

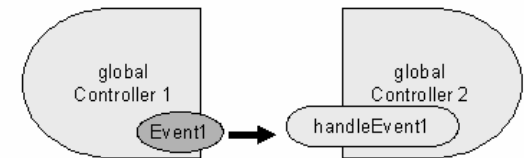
- Binden der Eigenschaften in View enthaltener Oberflächenelemente an Attribute eines View-Contexts
 - ◆ Anzeige der Daten des Contextes im Browser angezeigt
 - ◆ Ablage der Änderungen des Anwenders Browser angezeigtem View im Context
 - ◆ Binden mehrerer Eigenschaften an Context-Element möglich
- zur Definition von Context-Attributen stehen alle elementaren Datentypen zur Verfügung
 - ◆ hinterlegte Informationen werden zur Erstellung von
 - Wertehilfen und
 - Fehlermeldungen bei ungültigen Eingaben genutzt



Bindung an Web Dynpro-Model

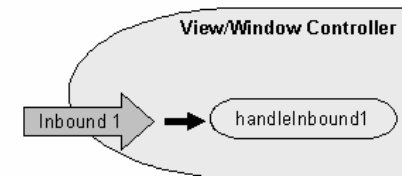
- Bindung zwischen
 - ◆ Context eines Component-Controllers und
 - ◆ Daten eines Web-Dynpro-Models **Demo!**

- Component-Controller erlauben Anlegen von Ereignissen für
 - ◆ Kommunikation zwischen Controllern über Auslösen eines Ereignisbehandlers in anderem Controller
 - ◆ Realisierung Component-übergreifender Kommunikation über Ereignisse des Interface-Controllers kann realisiert werden
 - im Component-Controller angelegte Ereignisse sind nur innerer der Component sichtbar



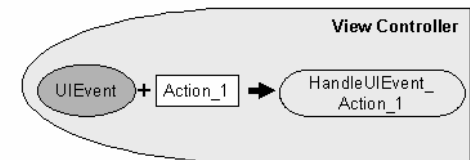
Inbound-Plugs

- Inbound-Plugs von Views bzw. Windows agieren wie Ereignis
 - ◆ Bei Aufruf eines View / Window über Inbound-Plug wird zuerst zum Inbound-Plug gehöriger Ereignisbehandler aktiviert
 - ◆ Ereignisbehandlung findet innerer aktuellem Controller statt



Ereignisse von UI-Elementen

- einige UI-Elemente (z. B. Button) besitzen
 - ◆ spezielle Ereignisse, die
 - ◆ mit Benutzeraktionen verbunden sind
- Ereignisse sind vordefiniert und werden zur Designzeit mit Aktion verknüpft werden
 - ◆ Anlegen einer Aktion bedingt automatische Generierung einer Ereignisbehandlungsmethode

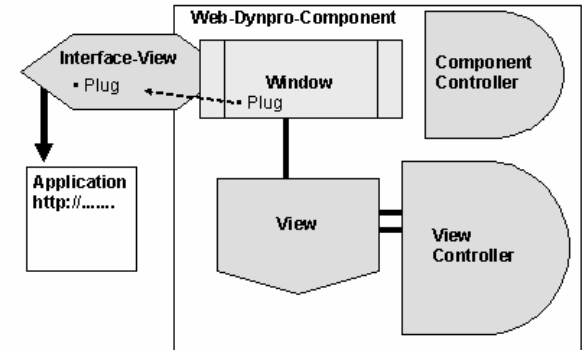


Interface-View in Component enthaltenen Windows

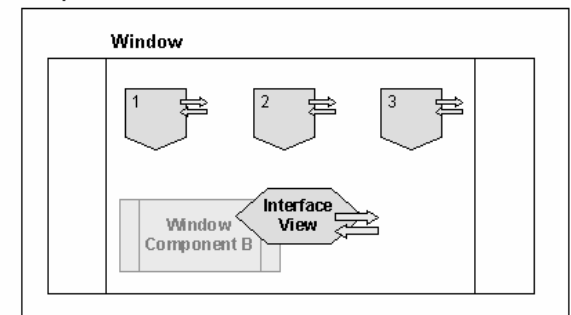
- verbindet Window mit vom Benutzer aufrufbarer Web-Dynpro-Anwendung
- eingebettete Component kann
 - ◆ über die Inbound- und Outbound-Plugs eines ihrer Interface-Views
 - ◆ in eines der Windows der einbettenden Component

integriert werden

- In- bzw. Outbound-Plugs sind als Window-Bestandteil implementiert
 - ◆ Window eingebetteter Component
 - verhält sich bezüglich Navigation wie View
 - muss keine grafischen Elemente besitzen nur & kann
 - nur in Component-Controller enthaltenen Funktionen oder
 - Contexte
- zur Verfügung stellen

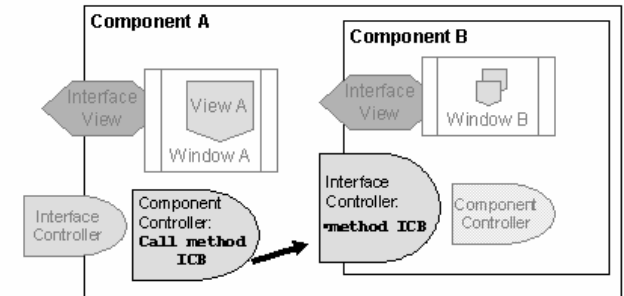


Component A



Interface-Controller einer Component

- dient programmatisch zum Austausch von Business-Daten, ist
 - ◆ innerhalb und
 - ◆ außerhalbder Component sichtbar
- einbettende Component kann eingebettete Component über diesen Controller aufrufen
- kein eigenständig implementiertes Objekt
 - ◆ Sichtbarkeit von Methoden und Ereignissen des Component-Controllers kann
 - ◆ über Grenzen eigener Component hinaus erweitert werden
- Möglichkeit zu Datenaustauschs zwischen
 - ◆ einbettender und
 - ◆ eingebetteter Componentauch über Component-übergreifendes



Context-Mapping

Inteface-Controller einer Component

■ Schnittstelle einer Web-Dynpro-Component kann

- ◆ eigenständig
- ◆ ohne Implementierung

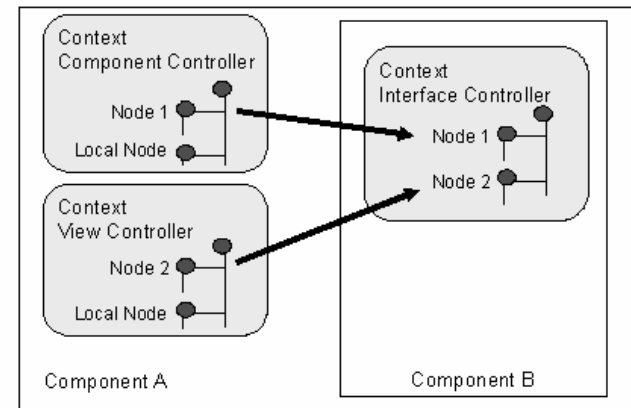
zur Trennung der Entwicklung

- ◆ der Web-Dynpro-Component und
- ◆ ihrer Verwendung

definiert werden

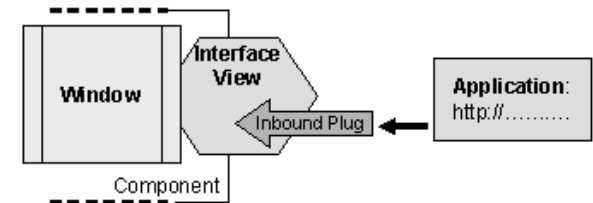
■ Möglichkeit zur Erstellung mehrere Schnittstellen-Implementierungen für eine Web-Dynpro-Component

- ◆ Auswahl gewünschte Implementierung erst zur Laufzeit
- ◆ Kopplung von Schnittstelle und Implementierung erfolgt über Namensgleichheit.



Anwendung

- von der Benutzungsoberfläche aus aufrufbare eigenständige Programmeinheit
- stellt Verbindung zwischen
 - ◆ dem Nutzer zugänglichen URL und
 - ◆ einem Window einer Web-Dynpro-Component her
- nur mit Inbound-Plug des Interface-View eines Web-Dynpro-Windows verbunden
 - ◆ enthält keine Information über hinter Interface-View liegende Teile betreffender Component



Model

- sorgt für Bereitstellung der Anwendungsdaten aus Backend-System
 - ◆ Auslesen vorhandener SAP-Daten mittels BAPIs
 - ◆ Definition neuer Daten
 - ◆ Aufruf von Web Services
 - ◆ Mischform
 - ◆ Import externen Models

Demo!

- Einführung
- Architektur
- **Voraussetzungen**
- Einsatzszenarien
- WDA oder WD4J?
- Live-Entwicklung Web Dynpro for Java
- Live-Entwicklung Web Dynpro for ABAP
- Diskussion



Systemvoraussetzung

- WD4J: SAP Web AS 6.40 Java) Installation und
- WDA: SAP Web AS 7.0 ABAP) initiale Konfiguration

Bedarfsabhängige Konfiguration

- Portal-Integration
 - ◆ diverse Voraussetzungen bezüglich
 - User- & Rollenverwaltung
 - Systemregistrierung
- Adobe-Integration
 - ◆ SAP NetWeaver AS System mit Adobe Document Services (ADS)
 - ◆ Installation des Layout Designer mit SAP GUI
 - ◆ auf Client
 - Adobe Reader (>= 7.0) und
 - für interaktive Formulare ActiveX Control Framework (ACF)
 - ActiveX im Browser aktiviert
- Einsatz von UI-Elementen der
 - ◆ ActiveComponent-Bibliothek
 - am Frontend üblicher Browser und Installation Java-Runtime 1.4 (Sun Plug-Ins)
 - ◆ OfficeIntegration-Bibliothek
 - Installation Microsoft Office
 - ActiveX im Browser aktiviert
 - ◆ BusinessIntelligence-Bibliothek
 - Verbindung zu BI-System bzw. entsprechende BI-Daten im System.
 - ◆ BusinessGraphics-Bibliothek
 - für GeoMap UI-Element Softwarekomponente für geografischen Karten

- Einführung
- Architektur
- Voraussetzungen
- **Einsatzszenarien**
- WDA oder WD4J?
- Live-Entwicklung Web Dynpro for Java
- Live-Entwicklung Web Dynpro for ABAP
- Diskussion



- Einführung
- Architektur
- Voraussetzungen
- Einsatzszenarien
- **WDA oder WD4J?**
- Live-Entwicklung Web Dynpro for Java
- Live-Entwicklung Web Dynpro for ABAP
- Diskussion



Entwicklungskompetenz

- Wo liegt Programmierkompetenz in Entwicklungsabteilung:
 - ◆ ABAP oder
 - ◆ Java?
- Es empfiehlt sich Entwicklung in gewohnt produktiver Umgebung!

Infrastruktur

- Welche Infrastrukturellen Voraussetzungen sind gegeben?
 - ◆ Existiert bereits DEV/QA/PROD ABAP Infrastruktur?
- Entwicklungsprojekt kann einfacher durch Systemlandschaft transportiert werden, falls
 - ◆ UI und
 - ◆ Geschäftslogikin selber Entwicklungsumgebung erstellt wurden!

Geschäftslogik

- Ist Geschäftslogik mehrheitlich als EJBs auf J2EE Servern oder in ABAP basierten SAP Systemen realisiert?
- Es gilt:
 - ◆ Liegt Code-Basis in Java ist WD4J sinnvoll,
 - ◆ umfangreiche ABAP Kundenentwicklungen sind - trotz adaptiven RFCs – einfacher über native ABAP anzusprechen.
- WD4J unterstützt mehrere Models
 - ◆ JavaBeans,
 - ◆ Web Services,
 - ◆ Adaptiven RFC und
 - ◆ XMI,

WDA bietet derzeit nur Funktionsbausteine als Modell (Erweiterung geplant)

Entwicklungsumgebung

- Welcher Editor bieten umfangreichere Entwicklungsunterstützung?
 - ◆ WDA verfügt über Code Wizard
 - ◆ NWDS bietet
 - Code Completion (hilfreich bei Verwendung der WD API) und
 - online Syntax Checking

Mobile Applikationen

- Werden Anwendungen für mobile Geräte entwickelt?
 - ◆ Derzeit bietet nur WD4J Unterstützung für mobile Geräte, für WDA geplant

SAP List Viewer

- Wird ALV benötigt?
 - ◆ Derzeit nur in WDA realisiert, ALV-Funktionalität für WD4J geplant

Integration in NetWeaver und Drittanbieter-Technologien

- Werden Portal & Knowledge Management-Service verwendet?
- Sollen APIs von Drittanbietern für
 - ◆ Rechtschreibprüfung,
 - ◆ leistungsfähigere XML Parser,
 - ◆ Monitoring APIs uämverwendet werden?
- Werden Applikationen für die CAF Laufzeit entwickelt?
- In diesen Fällen
 - ◆ gilt Axiom „Business Logik in J2EE“ und
 - ◆ empfiehlt sich Verwendung von WD4J

- Einführung
- Architektur
- Voraussetzungen
- Einsatzszenarien
- WDA oder WD4J?
- **Live-Entwicklung Web Dynpro for Java**
- Live-Entwicklung Web Dynpro for ABAP
- Diskussion



- Einführung
- Architektur
- Voraussetzungen
- Einsatzszenarien
- WDA oder WD4J?
- Live-Entwicklung Web Dynpro for Java
- **Live-Entwicklung Web Dynpro for ABAP**
- Diskussion



- Einführung
- Architektur
- Voraussetzungen
- Einsatzszenarien
- WDA oder WD4J?
- Live-Entwicklung Web Dynpro for Java
- Live-Entwicklung Web Dynpro for ABAP
- **Diskussion**

