Effiziente Entwicklung persistenter Geschäftsobjekte

Stefan Giegerich
Senior Consultant
Versata GmbH

sgiegerich@versata.de

Agenda

- Einleitung
- Die Herausforderung: *Mehrschicht-Architektur*
- Deklaratives - Regelbasiertes Entwickeln
- Der Versata *Business Logic Designer*
- Zusammenfassung / Diskussion

Die Herausforderung: *Mehrschicht-Architektur*

Warum Herausforderung ?

- Fehlendes *Know-How* (Erfahrung)
  - JSP+JAVA *!=* J2EE
  - MVC *!=* Mehrschicht-Architektur
- Höhere Komplexität / Abstraktion
- Viele verschiedene Technologien
  - Web-, App- und DB- Server (Konzepte, Konfiguration, Tuning, …)
  - Backendsysteme / EAI
  - Netzwerke / Firewalls

Die Herausforderung: *Mehrschicht-Architektur*

Worum geht es hauptsächlich ?

- Aufbau von Mehrschichtarchitekturen bestehend aus ***mindestens drei*** Schichten
- Funktionalität der Client-Schicht beschränkt auf das reine Präsentieren von Daten
- Kapseln der gesamten Geschäftslogik (Prozesse und Transaktionen) in die mittlere(n) Schicht(en)

Die Herausforderung: *Mehrschicht-Architektur*

- Warum N-Schicht Architektur ?
  - Ermöglicht erst Wiederverwendung (Modularisierung/Komponentenarchitektur/Servicearchitektur)
  - Erlaubt die zeitgleiche Erstellung verschiedener Systemteile
  - Ermöglicht ein hohes Maß an Interoperabilität / Integrierbarkeit
  - Flexibilität
  - Ermöglicht die Skalierbarkeit von Systemen
  - Reduzierte Kosten für Wartung und Erweiterung
  - Qualitätssteigerung
  - …

Die Herausforderung: *Mehrschicht-Architektur*

Presentation*Tier*

Presentation *GUI*

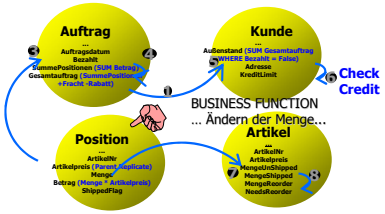
- HTML
- JSP
- Applets
- …

Presentation *Logic*

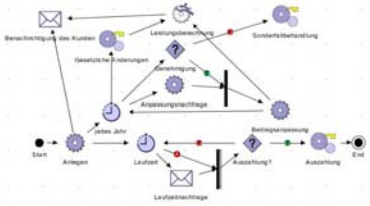
- GUI-Frameworks
  - Struts,…
- GUI-Builder
- "Interfaces"
  - XML
  - WebServices
  - …

Business*Tier*

Business *Logic*
Business Objects and Rules



Business *Processes*



Data*Tier*

Data *Access*

Data "*Storage*"

Backend Integration
- Databases
- EAI
- Applications
- Messaging
- …

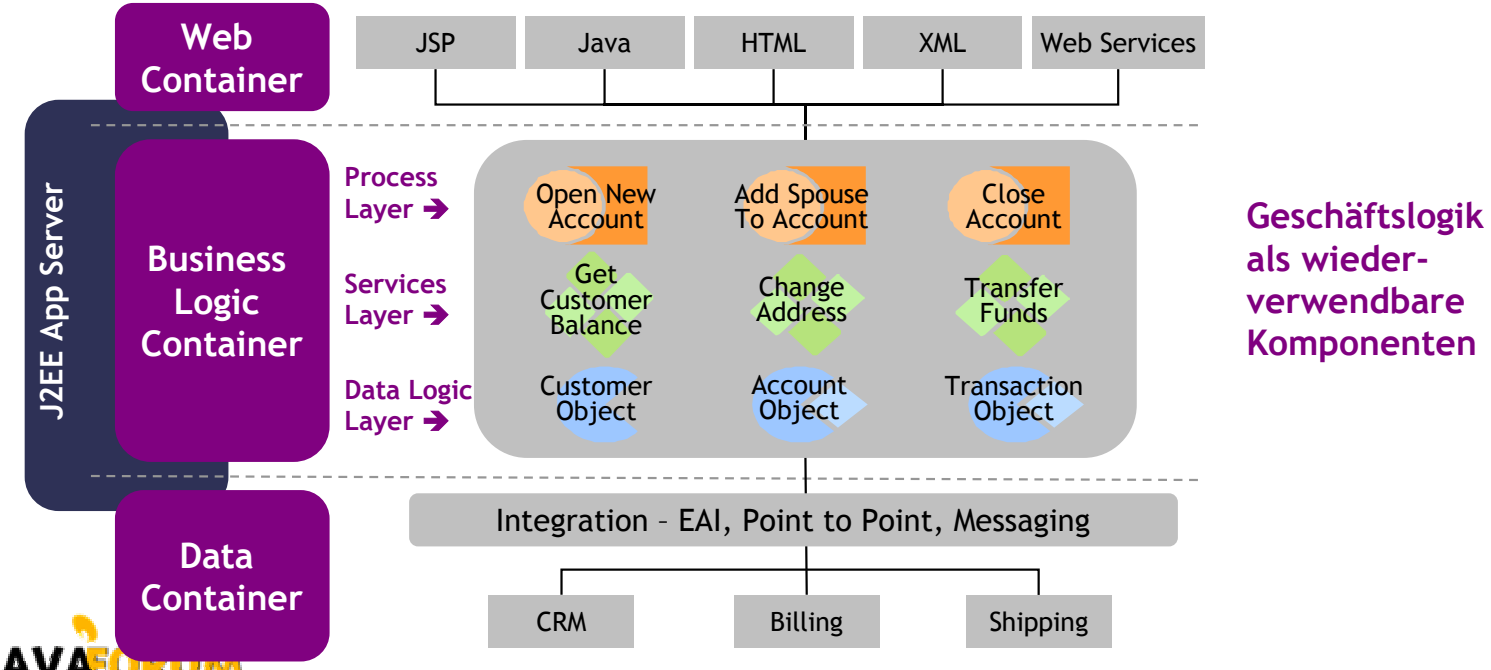
JAVAFORUM stuttgart 2003

Die Herausforderung: *Mehrschicht-Architektur*

- Was ist das *Wichtigste* ?
  - *Saubere* Trennung der logischen Schichten
    - Klare Trennung bei Systemdesign **und** Implementierung
    - Zugriff nur über die jeweiligen Schnittstellen
  - Erstellen der Geschäftslogik-Schicht
    - Oft fehlt diese komplett oder existiert nur als Pseudo-Schicht (Client-Server mit J2EE-Mitteln)
    - Service-Orientierte Implementierung (Fokus: Wiederverwendung)
    - Erfordert die Implementation von „Kerndiensten"
      - Transaktions-Handling / Transaktions-Cache
      - Daten-Mapping / Dynamic Data Access
      - Exception-Handling
      - ...
  - Integration zu bestehenden Systemen nur über die mittlere Schicht

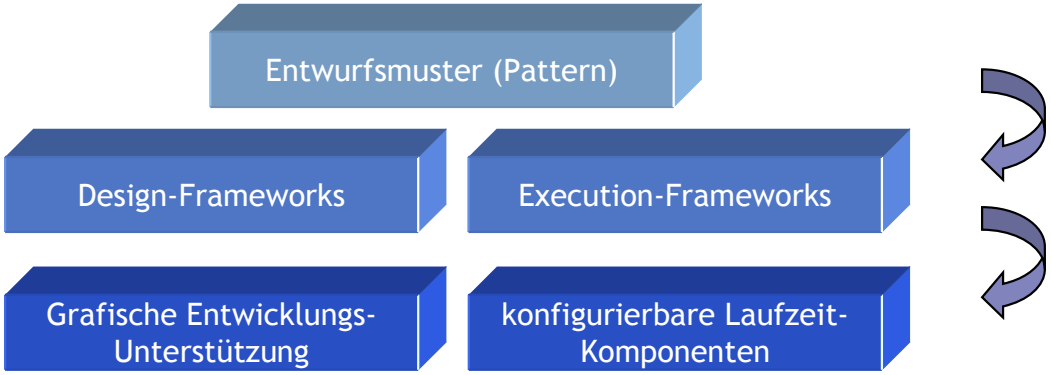
Die Herausforderung: *Mehrschicht-Architektur*

Die eigentliche Herausforderung liegt im Aufbau einer *Service-Orientierten Architektur (SOA)* *(Source: Gartner Group)*



Die Herausforderung: *Mehrschicht-Architektur*

- Was hilft ?
  - Ausbildung / Weiterbildung / Erfahrung
  - Werkzeugunterstützung (IDE's, PlugIn's, ToolSet's, …)
  - Methodik, Standards (MDA, J2EE, Pattern, …)
  - Frameworks und Komponenten

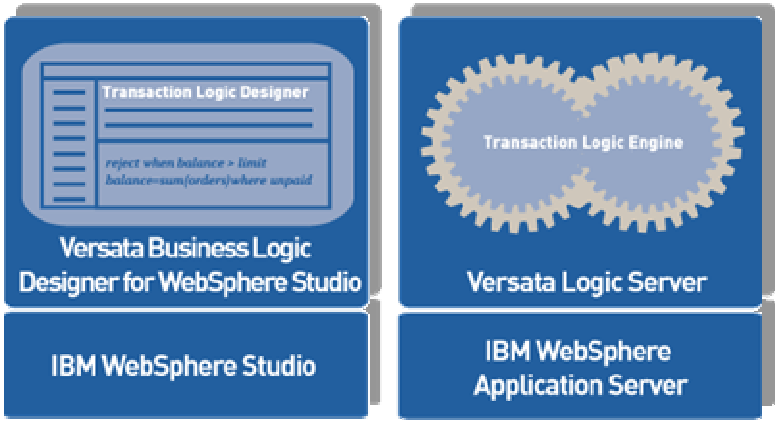


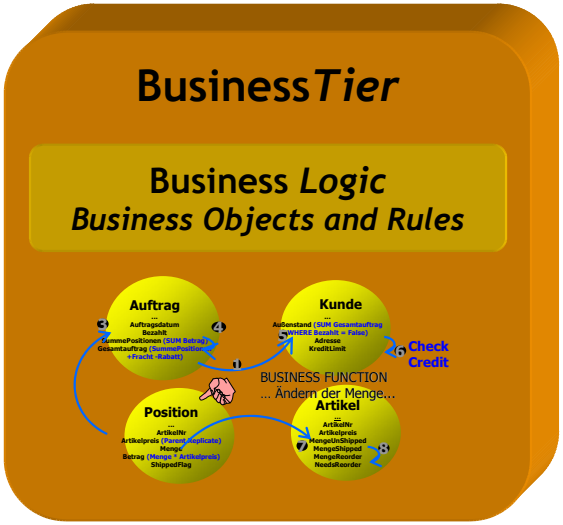
Die Herausforderung: *Mehrschicht-Architektur*

Versata *J2EE Framework Komponenten*
zur Erstellung
Persistenter Geschäftsobjekte

Grafisches, deklaratives Design

Laufzeit-Framework





JAVAFORUM stuttgart 2003

Agenda

- Einleitung
- Die Herausforderung: *Mehrschicht-Architektur*
- Deklaratives - Regelbasiertes Entwickeln
- Der Versata *Business Logic Designer*
- Zusammenfassung / Diskussion

Deklaratives - Regelbasiertes Entwickeln

Die Idee:

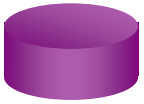
- Beschreiben anstatt Codieren
- Weitgehendes Automatisieren
- Konzentration auf das Wesentliche
  (What not How)
- Framework-Unterstützung in Design, Implementierung und Laufzeit

➢ Zeitersparnis

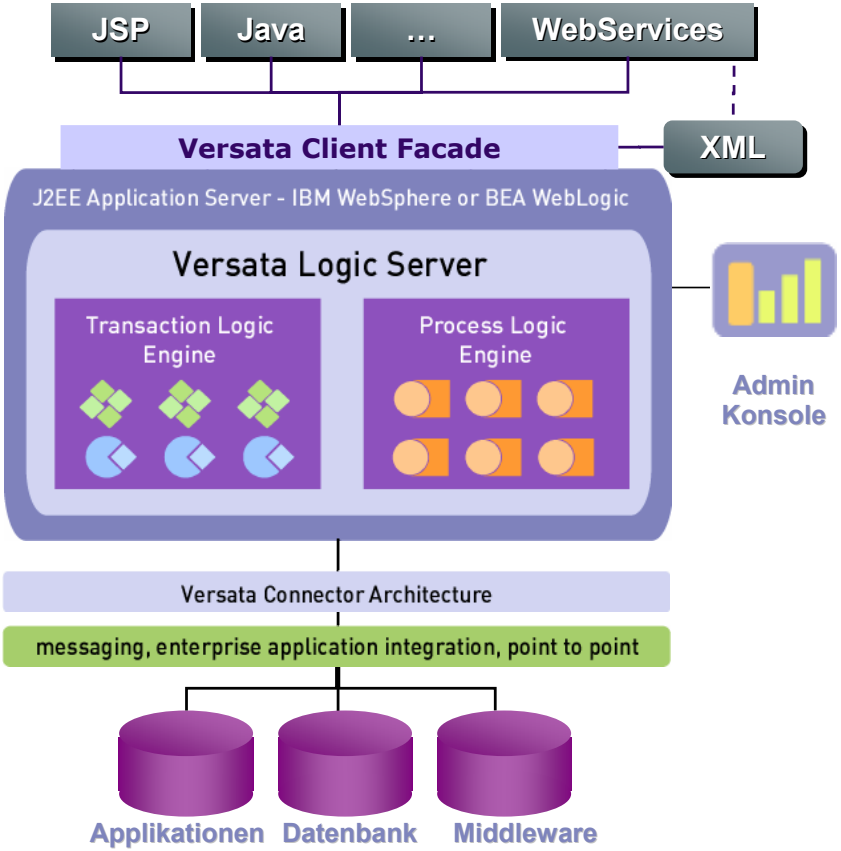
Deklaratives - Regelbasiertes Entwickeln

Anforderungen/Analyse:
Objekt Modelle, Use Cases, UML ...

Spezifikation



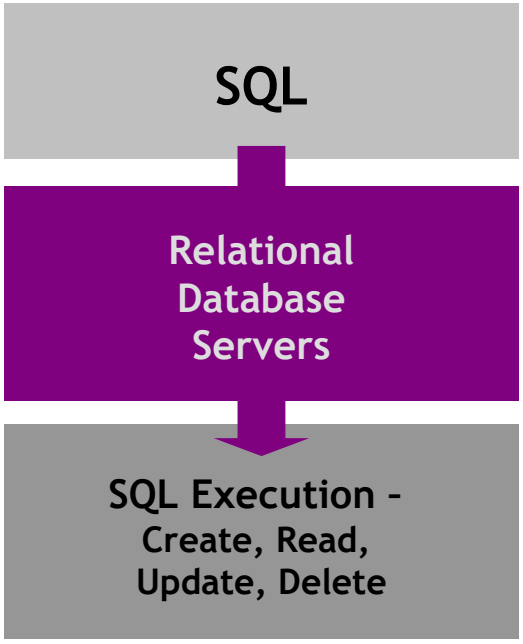
Rules Repository (XML)

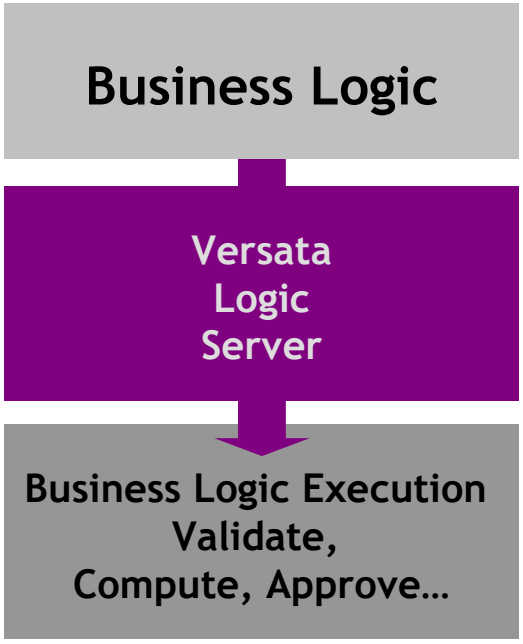


JAVAFORUM stuttgart 2003

Deklaratives - Regelbasiertes Entwickeln

Der Deklarative Ansatz ist nicht neu …

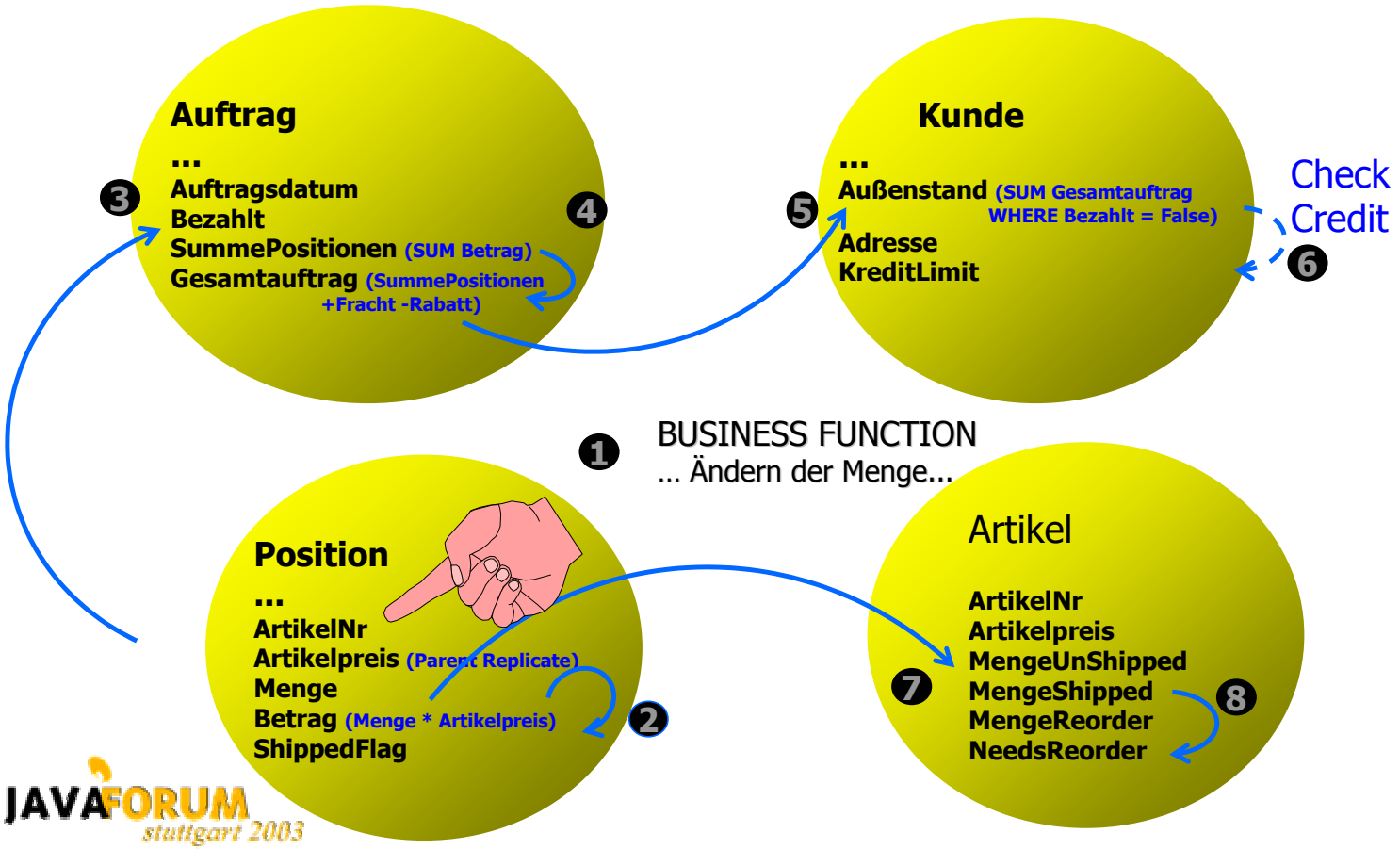




Data Management

Business Logic Management

Deklaratives - Regelbasiertes Entwickeln



JAVAFORUM stuttgart 2003

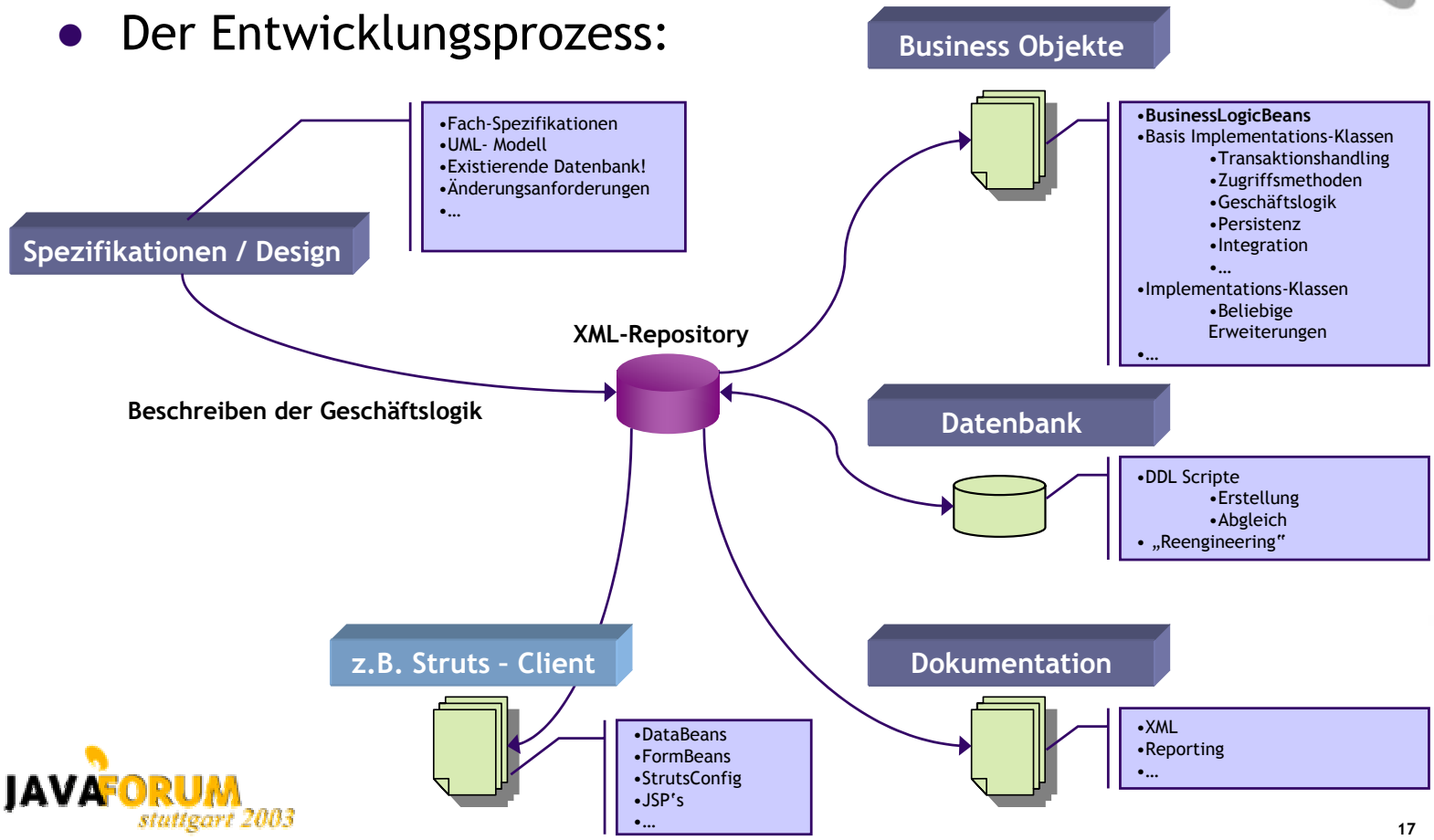
Deklaratives - Regelbasiertes Entwickeln

Deklarative Beschreibung von **Business Rules …**

- **Beziehungen**(Relationships)
  - an Account HAS Holdings
  - "cascade" rules - when updating Account.UserID, update related Holding.UserID
- **Ableitungen**(Derivations - multi-object, chained, unordered)
  - Account.PortfolioValue = sum Holdings.CurrentValue
  - Holding.Price replicates Quote.Price
  - QtyOnHand = QuantityBought * QuantitySold
- **Beschränkungen**(Constraints)
  - NoShortSelling (Reject when Holding.QtyOnHand Balance < 0)
- **Aktionen**(Actions)
  - When Inserting Holding, insertBuyTransaction()
- …

Deklaratives - Regelbasiertes Entwickeln

- Der Entwicklungsprozess:



Deklaratives - Regelbasiertes Entwickeln

- **Automatisierte Erstellung und**
  - Deklarative Beschreibung der Geschäftlogik
  - Codegenerierung (Business Rules Optimizer and Sequencer)
  - Beliebiges Anpassen und Erweitern durch offene Architektur (pur JAVA)
  - ➢ Development: Versata Studio / WSAD PlugIn

- **Management der Geschäftsobjekte** (BusinessLogicBeans)
  - Transaktions-Management (Automatisch / Benutzerdefiniert)
  - Security-Management (User, Rollen, Rechte)
  - Persitenz-Management
  - Regel/Ausnahme-Behandlung
  - Tracing und Monitoring
  - ➢ Runtime: Transaction Logic Engine

Deklaratives - Regelbasiertes Entwickeln

Server-Framework

- Connectivity – Framework (Versata XDA)
- DB Connection Pooling
- Session Management for object access
- Transaction Management (automatic or developer defined)
- Transaction Cache
- Data Validator / Logic Optimizer / Dependency Manager / Logic Sequencer / Object Synchronizer / Logic Integrity Manager
- Optimistic – Locking Features
- Object Caching (for Coded Value Lists)
- Query Objects
- Scrollable / Sortable Object Queries (object fetching)
- …

Deklaratives - Regelbasiertes Entwickeln

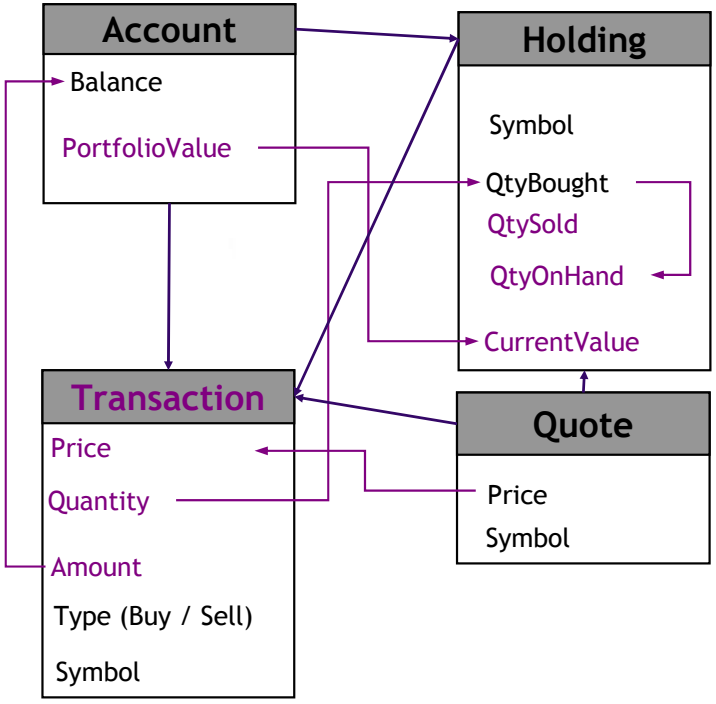
„Client" / Zugriffs - Framework

- Zugriff über:
  - Client API (VFC):
    - Session Facade Pattern
    - Object ResultSet Handling
  - Server API
  - XML
- ToolKits erhältlich für:
  - JSP – Struts
  - JSP
  - WebServices
  - HTML / JavaApplets (Versata Studio)
- J2EE „Wrapper" Klassen
  - Stateless oder Statefull Session Beans
  - Entity Beans

Agenda

- Einleitung
- Die Herausforderung: *Mehrschicht-Architektur*
- Deklaratives - Regelbasiertes Entwickeln
- Der Versata *Business Logic Designer*
- Zusammenfassung / Diskussion

Der Versata *Business Logic Designer* *am Beispiel*



Relationships

- Account is parent of:
  - Transaction
  - Holding
- Holding is parent of:
  - Transaction
  - Holding
  - Transaction
- Quote is parent of:
  - Holding
  - Transaction
- Quantity input bei Transaction
- Transaction.Price replicate Quote.Price
- Transaction.Amount = Quantity * Price
- When Inserting Buy Transaction reduce Account.Balance (:call Account.debit(Amount))
- Insert/Update Parent Object Holding (Relationship)
-

Re-calculate PortfolioValue (Holding, Account)

- Holding.QtyBought
  = Sum(Transaction.Value where type='bought')
- Holding.QtyOnHand
  = QtyBought - QtySold
- Holding.CurrentValue
  = QtyOnHand * getQuote().getPrice()
- Account.PortfolioValue
  = Sum(Holding.CurrentValue)

Der Versata *Business Logic Designer* *am Beispiel*

- *Business Objekte (Logic Beans)*

Web Browser | ACCOUNT Logic Bean | TRANSACTION Logic Bean

TRANSACTION

Alerts and Action Items: No alerts at this time.

Attributes

The following attributes are defined for this TLB:

- transID
- userID
- symbol
- transType
- quantity
- price
- amount

Add... Details Remove

Constraints

The following constraints are defined for this TLB:

- MinQuantity

Add... Details Remove

Associations

The following are the association roles for this TLB:

- Account [TLB: ACCOUNT]
- Holding [TLB: HOLDING]
- Quote [TLB: QUOTE]

Add... Details Remove

Actions

The following actions are defined for this TLB:

- Buy : Debits account balance when stock is bought
- Sell : Credits account balance when stock is sold

Add... Details Remove

Java

Java classes:

- TRANSACTIONBaseImpl : generated implementation file
- TRANSACTIONImpl : customizable implementation file

Open

TLB Superclass: DataObject

Data Access

The following properties are used for Data Access and Runtime Security:

External Object: DB2UDBNT_V72_1\DB2ADMIN\TRANSACTION

XDA Connector:

Lock Mode: AllApplicableAttributes

☐ Deploy Attribute Security

Description

Extended Properties

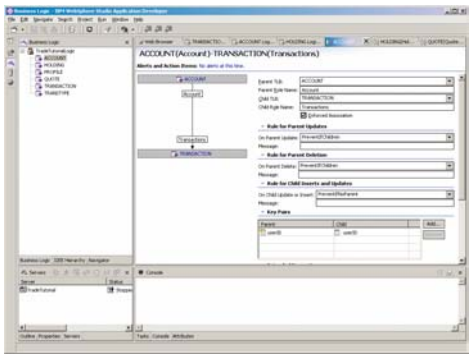
User defined properties:

Overview | Attributes | Constraints | Actions

JAVAFORUM stuttgart 2003

Der Versata *Business Logic Designer* *am Beispiel*

➔ *Relationships*



ACCOUNT(Account)-TRANSACTION(Transactions)

Alerts and Action Items: No alerts at this time.

ACCOUNT

Account

Transactions

TRANSACTION

Parent TLB: ACCOUNT

Parent Role Name: Account

Child TLB: TRANSACTION

Child Role Name: Transactions

☑ Enforced Association

Rule for Parent Updates

On Parent Update: PreventIfChildren

Message:

Rule for Parent Deletion

On Parent Delete: PreventIfChildren

Message:

Rule for Child Inserts and Updates

On Child Update or Insert: PreventIfNoParent

Message:

Key Pairs

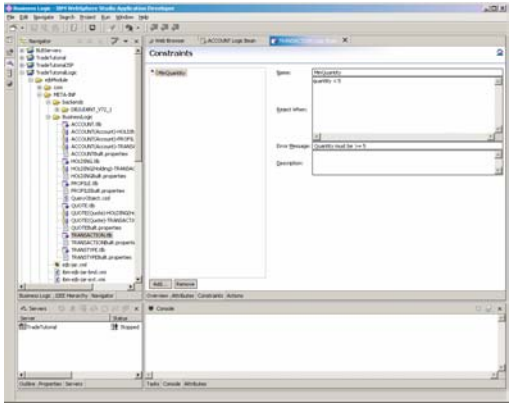
Parent	Child
userID	userID

Add... Remove

JAVAFORUM stuttgart 2003

Der Versata *Business Logic Designer* *am Beispiel*

➔ Check if Quantity is > 5 (*Constraint*)



Name: MinQuantity

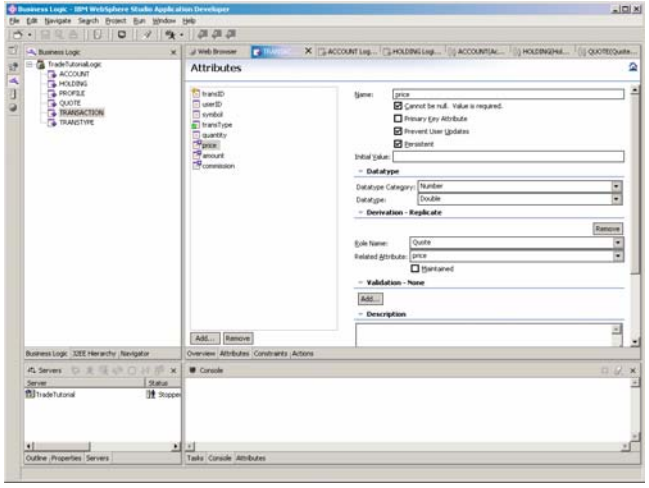
Reject When: quantity < 5

Error Message: Quantity must be >= 5

Description:

Der Versata *Business Logic Designer* *am Beispiel*

➔ Transaction.Price replicate Quote.Price (*Parent Replicate*)



Name: price

☑ Cannot be null. Value is required.

☐ Primary Key Attribute

☑ Prevent User Updates

☑ Persistent

Initial Value:

Datatype

Datatype Category: Number

Datatype: Double

Derivation - Replicate

Remove

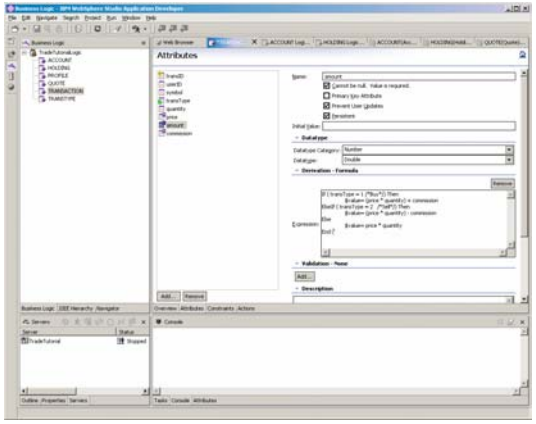
Role Name: Quote

Related Attribute: price

☐ Maintained

Der Versata *Business Logic Designer* *am Beispiel*

➔ Transaction.Amount = Quantity * Price (*Derivation - Formula*)



Name: amount

☑ Cannot be null. Value is required.

☐ Primary Key Attribute

☑ Prevent User Updates

☑ Persistent

Initial Value:

Datatype

Datatype Category: Number

Datatype: Double

Derivation - Formula

Remove

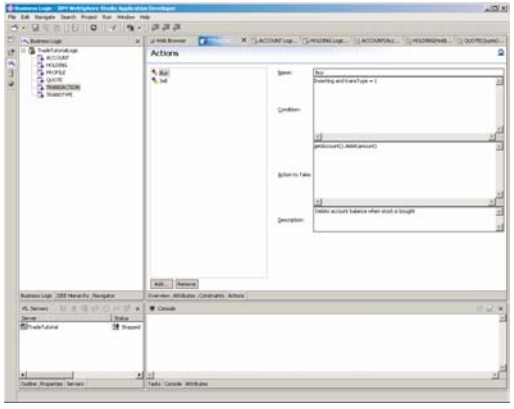
Expression:

```
If ( transType = 1 /*Buy*/) Then
        $value= (price * quantity) + commission
ElseIf ( transType = 2  /*Sell*/) Then
        $value= (price * quantity) - commission
Else
        $value= price * quantity
End if;
```

JAVAFORUM stuttgart 2003

Der Versata *Business Logic Designer* *am Beispiel*

- When Inserting Buy Transaction reduce Account.Balance (*Action*)
  (call Account.debit(Amount))



Name: Buy

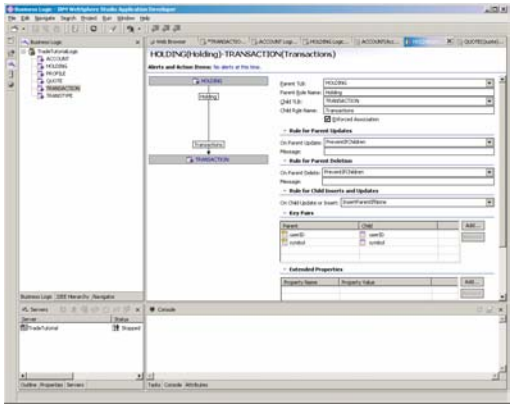
Condition: Inserting and transType = 1

Action to Take: getAccount().debit(amount)

Description: Debits account balance when stock is bought

Der Versata *Business Logic Designer* *am Beispiel*

➔ Insert/Update Parent Object Holding (*Relationship*)



Parent TLB: HOLDING
Parent Role Name: Holding
Child TLB: TRANSACTION
Child Role Name: Transactions
☑ Enforced Association

Rule for Parent Updates

On Parent Update: PreventIfChildren
Message:

Rule for Parent Deletion

On Parent Delete: PreventIfChildren
Message:

Rule for Child Inserts and Updates

On Child Update or Insert: InsertParentIfNone

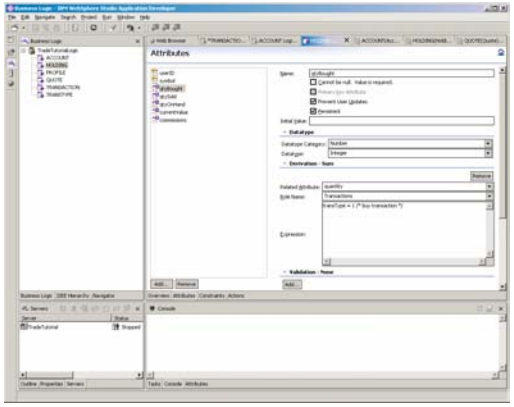
Key Pairs

Parent	Child
userID	userID
symbol	symbol

Add... Remove

Der Versata *Business Logic Designer* *am Beispiel*

- Holding.QtyBought = Sum(Transaction.Value where type='bought') (*Derivation - Sum*)



Name: qtyBought

☐ Cannot be null. Value is required.

☐ Primary Key Attribute

☑ Prevent User Updates

☑ Persistent

Initial Value:

Datatype

Datatype Category: Number

Datatype: Integer

Derivation - Sum

Remove

Related Attribute: quantity

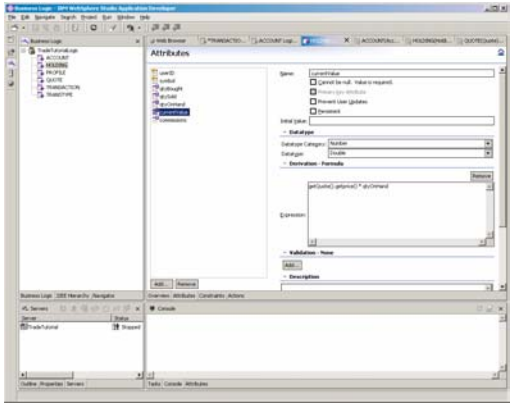
Role Name: Transactions

Expression:

```
transType = 1 /* buy transaction */
```

Der Versata *Business Logic Designer* *am Beispiel*

- Holding.QtyOnHand = QtyBought - QtySold
  (*Derivation - Formula*)



Name: qtyOnHand

☐ Cannot be null. Value is required.

☐ Primary Key Attribute

☑ Prevent User Updates

☑ Persistent

Initial Value:

Datatype

Datatype Category: Number

Datatype: Integer

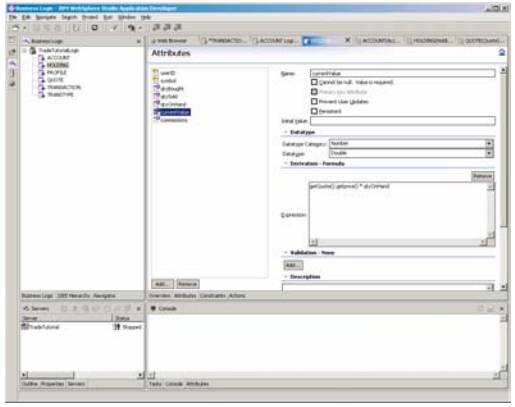
Derivation - Formula

Remove

Expression: qtyBought - qtySold

Der Versata *Business Logic Designer* *am Beispiel*

- Holding.CurrentValue = QtyOnHand * getQuote().getPrice()
  (*Derivation - Formula*)



Name: currentValue

☐ Cannot be null. Value is required.

☐ Primary Key Attribute

☐ Prevent User Updates

☐ Persistent

Initial Value:

Datatype

Datatype Category: Number

Datatype: Double

Derivation - Formula

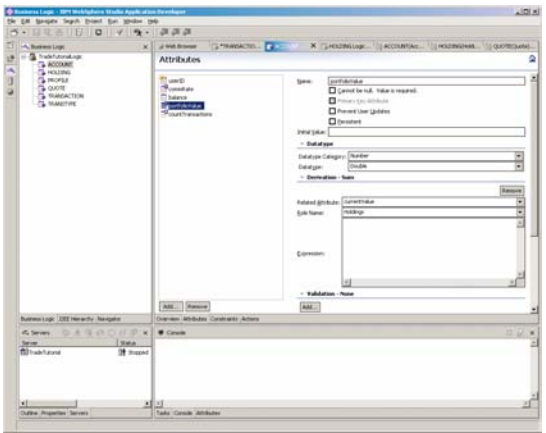
Remove

Expression: getQuote().getPrice() * qtyOnHand

JAVAFORUM stuttgart 2003

Der Versata *Business Logic Designer* *am Beispiel*

- Account.PortfolioValue = Sum(Holding.CurrentValue)
  (*Derivation - Sum*)



Name: portfolioValue

☐ Cannot be null. Value is required.

☐ Primary Key Attribute

☐ Prevent User Updates

☐ Persistent

Initial Value:

Datatype

Datatype Category: Number

Datatype: Double

Derivation - Sum

Remove

Related Attribute: currentValue

Role Name: Holdings

Expression:

Der Versata *Business Logic Designer* *am Beispiel*

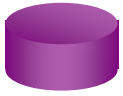
Ist Ihnen etwas aufgefallen ?

Geschäftsregeln für sich sind **transaktionsunabhängig**

- allerdings -

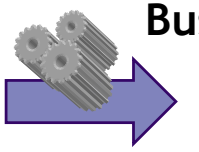
die Implementierung dieser Regeln **sehrwohl**

Regel-Repository

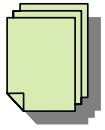


Abhängigkeits-Graph

Data Validator
Logic Optimizer
Dependency Manager
Logic Sequencer
Object Synchronizer
Logic Integrity Manager



Business Objekte



Der Versata *Business Logic Designer* *am Beispiel*

- Base Implementation Class

Web Browser | TRANSACTION Logic Bean | ACCOUNT Logic Bean | ACCOUNTBaseImpl.java

```
*        - association getters / setters (eg, an Order.getCustomer() )
*        - the save() method to enforce the business logic and persist the
*
*   When the Transaction Logic Bean, one of its Associations or a related
*   Logic Bean changes, this file will be deleted and rebuilt.  Any chang
*   to this file will be lost.  To customize the logic, add code in the A
*   Since it is built from the TLB, it is not strictly necessary to sourc
*/

//javaComponentBaseImportStmtsArchetype.tpl

package TradeTutorialLogic_blb;
import java.util.Enumeration;
import java.util.Vector;
import versata.common.*;
import versata.common.vstrace.*;
import versata.vls.*;
import versata.vls.cache.*;
import java.util.*;
import java.math.*;

//javaComponentBaseArchetype.tpl
public abstract class ACCOUNTBaseImpl extends  DataObject

{
    public ACCOUNTBaseImpl () {
        super();
        addListeners();
    }


    private final static String deployedFromRepository = "TradeTutorial
    /**
    Cache support
    */
    public static int cacheSize = 0;
    public static long expires = -1;
```

JAVAFORUM stuttgart 2003

Der Versata *Business Logic Designer* *am Beispiel*

- Implementation Class

Web Browser | TRANSACTION Logic Bean | ACCOUNT Logic Bean | ACCOUNTImpl.java

```
    public void addListeners() {

        /**
        *   add listeners for each event registered
        *
        *   For example,
        *   addTransactionEventListener(this);
        *      For a complete list of events, see the TransactionEventLis
        */
    }

    /* Static functions */
    public static ACCOUNTImpl getNewObject(
        Session session,
        boolean makeDefaults) {
        return new ACCOUNTImpl(session, makeDefaults);
    }

    /**  Event Handlers go here
    *   Note that events are methods in the object, so
    *   they can reference the public methods of TLB as shown below
    *
    *   public void beforeCommit(Session session, Response response) {
    *       String deptName =  getSalesRep().getDepartment.getName(); //
    *   }
    */
    public void debit(double amount) {
        setbalance(getOldbalance() - amount);
        save();
    }
    public void credit(double amount) {
        setbalance(getOldbalance() + amount);
        save();
    }
```

Der Versata *Business Logic Designer* *am Beispiel*

… kurze Live Präsentation …

Agenda

- Einleitung
- Die Herausforderung: *Mehrschicht-Architektur*
- Deklaratives - Regelbasiertes Entwickeln
- Der Versata *Business Logic Designer*
- Zusammenfassung / Diskussion

Was bietet mir das Business Logic Designer PlugIn?

- Mehrschicht Architektur !
- WSAD Integration
  - Test Server Integration
  - Debugging
  - Code Editor
  - Team Development
  - Integration mit anderen J2EE Komponenten (JMS, Web Services, etc.)
- *Business Logic Development*
  - Deklarative Geschäftslogik (GUI zum Editieren von Geschäftsregeln)
  - Entwicklungs- und Laufzeit Framework
  - J2EE data access patterns (z.B. für JSP, XML, …)

Warum machen wir das ?

Für Sie …

- ➔ Schnellere Entwicklungszyklen (Time to market)
- ➔ Minimierung der Aufwände für Änderung und Wartung
- ➔ J2EE Entwicklung mit limitierten Resourcen
- ➔ Steigerung der Rentabilität von IT (ROI)

Und technisch …

- ➔ J2EE Skalierbarkeit
- ➔ Wiederverwendbarkeit
- ➔ Transaktions- / Persistenz- Management
- ➔ Regelbasiertes Entwickeln

Distribution

- Free Download

  über http://www.versata.com

  - Spezieller *Support* für die Evaluierung
  - Beinhaltet bereits den JSP-Adapter
  - Struts-ToolKit über *support@versata.de*

Vielen Dank für Ihre Aufmerksamkeit

Weitere Info's auch an unserem Ausstellungsstand

JAVAFORUM stuttgart 2003