Vorlesung
Software aus Komponenten

3. Komponentenmodelle

Prof. Dr. Hans-Gert Gräbe
Wintersemester 2015/16

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Weitere Konzepte

- Transaktionskontrolle: entweder via JTA oder über EntityTransaction
- Persistenz-Kontexte: Innerhalb eines solchen Kontexts wird die Eindeutigkeit der Zuordnung Entity – Persistenzobjekt garantiert
- Query API mit eigener, an SQL angelehnter EJB Query Language
- Entity Packaging: Im Deployment-Deskriptor können genauere Informationen über EntityManager gespeichert werden.

Zusammenfassung

- Aufgaben des EJB-Container werden an den Kontext delegiert
- Spezielle Eigenschaften des Ausführungskontexts werden durch Annotationen direkt im Quelltext festgelegt statt als Deployment-Informationen
- Persistenz folgt dem Konzept der Java Data Objects JDO
- Standard fixiert in JSR 220

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Java-Basisdienste für Komponenten

Grundlegende Dienste

Java core reflection erlaubt zur Laufzeit

- Inspektion von Klassen und Schnittstellen nach Attributen und Methoden
- Konstruktion neuer Klasseninstanzen und Felder
- Zugriff und Modifikation von Attributen in Verbundobjekten oder Feldern
- Aufruf von Methoden von Objekten und Klassen

- **java.lang.reflect** als eigene Klasse hierfür
  - einige Funktionalität historisch in der (finalen) Klasse **java.lang**.

Java GuI-Klassensammlungen AWT und Swing

- delegierende Ereignisbehandlung
- Datentransfer und Zwischenablage wird unterstützt, drag and drop
- Java 2D rendering, eng damit zusammen Java printing model
- Internationalisierung
- pluggable look and feel, Palette von Standard-Komponenten

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Objektserialisierung

- standardisiertes Kodierungsschema für Serialisierung
- Klasse muss dafür Interface **java.io.Serializable** implementieren
- Objektserialisierung ist sicherheitskritisch
- Mechanismen zu Serialisierung und Deserialisierung ganzer Objekt-Webs
  - nicht zu serialisierende Attribute können als transient markiert werden
    - Bsp: große Cache-Strukturen
  - private Methoden **readObject** und **writeObject** werden statt Default genommen, wenn durch Reflektion gefunden
  - Mehrfachreferenzen auf ein Objekt werden rekonstruiert
- unterstützt einfaches Versionierungsschema:
  - 64-bit-hash-Code (Serial version UID = SUID) wird über die Signatur der Klasse berechnet und kann während **readObject** ausgewertet werden.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Ferne Objekte und RMI

- Auf ferne Objekte kann nie direkt zugegriffen werden, sondern nur über Interface **java.rmi.Remote**. Ressourcenbindung über Namensdienst.
- Remotezugriff kann immer fehlschlagen:
  Exception **java.rmi.RemoteException**
- Parameterübergabe:
  - Referenz, wenn Parameterwert selbst vom Remote-Typ ist
  - Durch Marshalling übergebene Kopie, wenn Parameterwert lokal im aufrufenden Kontext
  - Übergabe nicht serialisierbarer Objekte führt zu Laufzeitausnahme
- Unterstützt verteiltes Garbage Collection
  - durch genaue Buchführung über Remote-Referenzen
  - basiert auf Arbeit [Birrel 1993] über Network Objects
  - eines der bedeutendsten Features von Java RMI
- Konflikt mit Begriff der Objektidentität im Java-Standard
  - Referenzen auf ein fernes Objekt sind Java Referenzen auf das lokale Proxy des fernen Objekts
  - Backcall erzeugt ein lokales Proxy im ObjectHome, neben der eigentlichen Java-Referenz

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Weitere Java-Dienste für Komponenten

JNDI: Java Naming and Directory Service

Aufgabe: Dienste über Namen bzw. Attribute finden

- Interface **Context** macht Namenskontext verfügbar
- Methode **lookup** findet Objekte über ihren Namen
- Interface **DirContext** erweitert **Context** zur Suche über Attributwerte
- Unterstützung von Kontexthierarchien, die rekursiv durchsucht werden.

JMS: Java Messaging Service

Aufgabe: Unterstützung asynchroner datengetriebener Kompositionsmodelle

- Standardisiert Java-Zugriff auf vorhandenes Nachrichtensystem, implementiert keins selbst.

JDBC: Java database connectivity

Aufgabe: Einheitlicher Zugriff auf Datenbanken über entsprechende Treiber

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

JTA: Java Transaction API

JTS: Java Transaction Service

Aufgabe: Unterstützung von Transaktionskonzepten

JCA: J2EE Connector Architecture (seit J2EE 1.3)

- Einheitliches Konzept des Ressourcen-Adapters, über welches externe Ressourcen aus einer EIS (enterprise information structure) in einen J2EE-Applikationsserver eingebunden werden können
- Definition eines entsprechenden **JCA common client interface** (CCI)
- Einsatz innerhalb von Enterprise Application Integration Frameworks

Java und XML: Java unterstützt mit entsprechenden Klassen

- XML-Dokumente (DOM)
- XML-Streaming (SAX)
- XML-Binding (JAXP)
- XML-Messaging (JAXM)
- XML-Processing (JAXP)
- XML-Registries (JAXR)

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Java und CORBA:

- Java wichtigste CORBA-Referenzimplementierung
- Koexistenz in fast allen Applikationsserver-Produkten
- Zugriff auf CORBAservices über Java-spezifische Zugriffs-Schnittstellen sowie weitere Konzepte (POA, Namensdienst) seit Java 1.4
- RMI-over-IIOP als eingeschränkte RMI-Version seit 1999
  - RMI nutzt spezifisches proprietäres Protokoll
  - keine Unterstützung des verteilten Garbage Collection, so dass explizites Lebenszyklus-Management erforderlich ist
    - dafür existieren aber CORBAservices

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

> "... komplette Neudefinition der Art, wie Microsoft in Zukunft Geschäfte machen will ... und wie Software entwickelt werden soll."
>
> **Westphal, 2002**

- Plattform soll bisherige Vorgehensweisen der Windows-Programmierung ersetzen, flexibel auf Betriebssystem- und Basisfunktionen zugreifen und Austausch zwischen Programmen unterstützen.
- Ausgerichtet auf den Einsatz auf verschiedenen Hardware-Plattformen bis hin zu Handys und PDAs. Java-Idee ohne Beschränkung auf Java als Programmiersprache
- Ziele
  - Sicherheit
  - Plattformunabhängigkeit
  - Interoperabilität
  - Homogenität

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Vorgeschichte:

- Rechtsstreit zwischen Sun und Microsoft um Java
  - Microsoft erweitert Java nach eigenen Vorstellungen und Bedürfnissen und gefährdet damit die Java-Kompatibilität
  - Microsoft-Implementierungen J++ und J#
- Weitere Probleme:
  - Auch die für Windowsprogrammierung meist verwendeten Sprachen Visual Basic, C++ und J++ waren untereinander nicht kompatibel
  - String-Datentypen waren sogar nicht binär kompatibel - .NET ist konsequent Unicode basiert
  - kein einheitliches Modell der Speicherverwaltung

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- 1996: erste Arbeiten an .NET
- 2000: .NET-Framework 1.0 Beta
- Oktober 2000 – C# und die CLI werden von MS, HP und Intel zur Standardisierung bei der ECMA eingereicht
  - ECMA – European Computer Manufacturers Association
  - Dezember 2001 – Weitergabe des ersten Standards an die ISO
  - April 2003 – Verabschiedung der ISO-Standards ISO/IEC 23270 (C#) und ISO/IEC 23271 (CLI)
- Januar 2002: .NET Framework 1.0 und Visual Studio .NET
- April 2003 – Auslieferung von .NET Framework 1.1 zusammen mit Windows Server 2003, der eine integrierte .NET-Laufzeitumgebung zur Verfügung stellt.
  - Damit Übergang zur neuen Plattform auf der Ebene von Konzepten für Unternehmensserver
  - Integration in die Produktfamilie geht jedoch nicht so rasch voran wie erwartet

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- Ende 2005: .NET Framework 2.0 zusammen mit Visual Studio 2005, MS SQL Server 2005, MS BizTalk Server 2006
  - Damit weitere Integration in die Produktfamilie, aber viele Anwendungen verwenden noch starken Durchgriff auf assemblerspezifischen Windows-Code jenseits der IL
  - Damit nur eingeschränkte Plattformunabhängigkeit und Interoperabilität
- Ende 2006: .NET 3.0, später integraler Bestandteil von Windows Vista und Windows Server 2008, mit tiefgreifenden auch konzeptionellen Erweiterungen der Architektur
- Ende 2007: Visual Studio 2008 und .NET Framework 3.5
  - Base Class Library (BCL) wird in Framework Class Library (FCL) umbenannt
    - Umfasst fast 12.000 Klassen in 300 Namensräumen
  - Teilweise Freigabe des Quellcodes der Base Class Library unter der restriktiven Microsoft Reference Source License

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- April 2010: .NET 4.0 und Visual Studio 2010
  - Stärkere Ausrichtung auf Mehrkern-Systeme, verteilte Architekturen und Thread-Parallelität
  - Neues Programmiermodell für Multithreading und asynchronen Code
- April 2014: Microsoft kündigt die Gründung einer unabhängigen .NET Foundation an - http://www.dotnetfoundation.org
  - Januar 2015: Ankündigung der .NET Open Source Initiative
  - Stärkere Trennung zwischen .NET Framework und .NET Core (entspricht Unterscheidung von Komponentenmodell als Architekturkonzept und Komponentenplattform)
- .NET 2015 – von Microsoft geprägter Oberbegriff für .NET Framework 4.6 und .NET Core 5.0

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

.NET als Komponentenframework

.NET 3.0 schärft das Profil als Komponentenframework mit mehreren Konzepten, vergleichbar den CORBA Facilities

- *WPF – Windows Presentation Foundation:* Beschreibungssprache XAML zur Darstellung von Objekten auf dem Bildschirm.
- *WCF – Windows Communication Foundation:* Dienstorientierte Kommunikationsplattform für verteilte Anwendungen, in der viele Netzwerk-Funktionen zusammengeführt und standardisiert zur Verfügung gestellt werden.
- *WWF – Windows Workflow Foundation:* Infrastruktur für die einfache Entwicklung von Workflow-Anwendungen, sowohl in geschäftlicher als auch technischer Hinsicht, mit visuellen Modellierungsmöglichkeiten.
- *Windows CardSpace:* Identitätsmanagement-Infrastruktur für verteilte Anwendungen als Basis eines einheitlichen Authentifizierungs- und Autorisierungs-Frameworks

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

ECMA-Standardisierung erlaubt Implementierung des Standards auch auf anderen Plattformen.

Versionen jenseits von Windows:

- Microsoft selbst stellte 2002 mit der Shared Source CLI Versionen für Mac OS und FreeBSD bereit. Diese Aktivitäten wurden später wieder aufgegeben.
- Verschiedene Aktivitäten der Linux-Community, die Konzepte umzusetzen und eine freie .NET-Version zu schaffen.
  - 2009 startet das dotGNU-Projekt, das eine Laufzeitumgebung Portable.NET erstellen will. Kommt über Release-Version 0.1 nicht hinaus und wird Ende 2012 eingestellt.
    - As of December 2012, the DotGNU project has been decommissioned, until and unless a substantial new volunteer effort arises.
- Bleiben hinter der Leistungsfähigkeit der Windows-Versionen zurück.
- Einziges leistungsfähiges „freies“ Projekt ist das Mono-Projekt http://www.mono-project.com/

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Zur Geschichte des Mono-Projekts

- Miguel de Icaza und Nat Friedman gründen 1999 die Firma *Helix Code*, die 2001 in *Ximian* umbenannt wird.
  - Geschäftsmodell: Solutions and Services, basierend auf Mix von freier und kommerzieller Software
  - Beteiligt an Gründung des Gnome Projekts
  - 2002 Start des Mono-Projekts
- 2003 von *Novell* übernommen, das damit sein Linux-Portfolio weiter stärkt.
- 2011 wird Novell im Rahmen des großen Patentdeals von der *Attachmate Group* übernommen, die kein Interesse an der Weiterführung des Mono-Projekts haben.
  - After several months of discussions, the US Department of Justice (DOJ) and the German Federal Competition Office (FCO) have allowed a consortium of Microsoft, Oracle, Apple and EMC to acquire 882 patents from Novell only subject to conditions clearly intended to prevent their use against Free Software players. (FSFE Newsletter, April 2011)

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Zur Geschichte des Mono-Projekts (Fortsetzung)

- 2011 gründen Icaza und Friedman die Firma *Xamarin* http://xamarin.com und bündeln dort die weitere Entwicklung am Mono-Projekt
  - Fokus der Firma liegt auf mobilen Anwendungen.
- Der Mono-Kern, die Laufzeitumgebung, ist unter der LGPL v.2 frei verfügbar, aber Xamarin bietet auch kommerzielle Lizenzen für die Mono-Plattform an
  - If you are planning to use Mono as a bundled part of your commercial product, on embedded hardware, or in any other situation where using the LGPL-licensed Mono is impossible or problematic, Xamarin can sell you a commercially-friendly license that will suit your needs.
  - Many commercial users of Mono acquire a commercial license when they want the flexibility and peace of mind to use Mono without worrying about the terms of the LGPL.
- Neue Etappe der Zusammenarbeit: Ende 2013 gründen Microsoft, Xamarin und andere die *.NET Foundation* als neuer Rechteinhaber und Lizenzgeber des .NET Frameworks.
  - http://www.dotnetfoundation.org/

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

.NET Open Sourcing

- 2008 veröffentlichte Microsoft den Quelltext des Frameworks unter der restriktiven Microsoft Reference License.
- Ende 2013 gründeten Microsoft, Xamarin und andere die *.NET Foundation* als neuer Rechteinhaber und Lizenzgeber des .NET Frameworks. http://www.dotnetfoundation.org/
  - 2007 hatte Microsoft noch behauptet, dass das Mono-Projekt Rechte von Microsoft verletzt
- Ende 2014 wird eine Teilmenge des Reference Source Quellcodes auf GitHub gehosted und unter der MIT-Lizenz veröffentlicht.
  - https://github.com/dotnet
  - Dies geschah auch, damit Lücken zwischen Mono und .NET durch Verwendung desselben Codes geschlossen werden können.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

.NET Open Sourcing

- Gleichzeitig hat Microsoft damit begonnen, die überarbeiteten Komponenten des Framework unter der Bezeichnung *.NET Core* auf GitHub ebenfalls unter MIT-Lizenz zu veröffentlichen.
- Basis für das kommende, modular aufgebaute *.NET Framework 5*.
  - .NET Core ist von Microsoft an die .NET Foundation überstellt worden.
- Durch die Verwendung der MIT-Lizenz gibt es faktisch keine Einschränkungen mehr, wie der Quellcode von .NET Core verwendet werden darf.
  - Mit der Gründung der .NET Foundation und der Übertragung der Rechte und Quellcodes an die Foundation arbeitet Microsoft mit Xamarin aktiv zusammen, um .NET auf unterschiedlichen Plattformen bereitzustellen. Durch die Offenlegung der Quellcodes unter der MIT-Lizenz bzw. Apache 2.0 Lizenz ist der Quellcode des .NET Frameworks nahezu beliebig – auch in Closed-Source-Projekten – verwendbar. Lizenz- und patentrechtliche Auseinandersetzungen sind somit kaum noch möglich und auch nicht mehr zu befürchten. (Wikipedia)