Architekturkonzept

Two to Tango (3T)

Änderungshistorie

Version	Vorgelegt am	Von	Bemerkung
0.1	29.03.2012	Tobias Langlotz	Erstellung
0.2	29.03.2012	Tobias Langlotz	Management Summary erstellt
0.3	07.04.2012	Tobias Langlotz	Management Summary ergänzt und weitere Dokumentenstruktur erstellt.
0.4	16.04.2012	Tobias Langlotz	Vorgaben und Rahmenbedingungen definiert
0.5	19.04.2012	Tobias Langlotz	Grobe Architekturbeschreibung
0.6	22.04.2012	Tobias Langlotz	Verfeinerung der Architektur (4-Schichten)
0.7	28.04.2012	Tobias Langlotz	Auslieferung der Anwendung und Deployment Diagramm erstellt.
0.8	30.04.2012	Tobias Langlotz	Finalisierung des Dokuments
1.0	30.04.2012	Christian Braun	QS und Abnahme

Abbildungsverzeichnis

Inhalt

I Einleitung und Management Summary

Da die Anwendung TwoToTango 3T als Web-Plattform realisiert werden soll, kamen unterschiedliche Entwicklungsansätze in Betracht. Es wurde zwischen einer skriptbasierten PHP-Anwendung und einer programmiersprachenbasierten Java EE Anwendung abgewogen. Die Vorteile einer Java EE Anwendung gegenüber einer PHP-Lösung überwiegen deutlich, da eine Java EE Anwendung unserer Erfahrung nach besser modularisiert und getestet werden kann.
Dieses Dokument befasst sich mit der Architektur der Anwendung TwoToTango und erläutert, mit welchen Technologien die Ziele aus dem Lastenheft umgesetzt werden.
Es gliedert sich in folgende Kapitel:

- *Vorgaben und Rahmenbedingungen*
  Dieses Kapitel erläutert die in der Architektur berücksichtigen Anforderungen.
- *Konzeption der Architektur*
  Das Kapitel formuliert die gewählte Architektur und die Gründe, die zur Auswahl der Architektur führten.
- *Auslieferung der Artefakte*
  In diesem Kapitel wird beschrieben, welche Artefakte einmalig oder zu einem Release ausgeliefert werden und wo diese in der Infrastruktur platziert werden.

II Vorgaben und Rahmenbedingungen

II.a *Vorgaben zur Realisierung und Technologie*

Als relationales Datenbankmanagementsystem ist Oracle in der Version 11g vorgesehen.

Als Programmiersprache wird Java in der Version 6.0 verwendet. Einzelne performancekritische Programmteile können in PL/SQL für Oracle 11g implementiert werden. Als Rahmenwerk steht der komplette Java EE 6 Technologie-Stack zur Verfügung.

Der Zugriff auf die Datenbank erfolgt mittels *Java Database Connectivity (JDBC)*, welche durch die in Java EE 6 verwendete Java Persistence API (JPA) 2.0 gekapselt wird.

II.b *Zielplattform*

Die Anwendung TwoToTango wird für folgende Plattform erstellt:

Betriebssytem	Oracle Linux 6
Datenbankmanagement System	Oracle 11g Enterprise Edition RAC
Middleware	JBoss Application Server 7

Als relationales Datenbank-Management-System soll Oracle in der Version 11g dienen. Dieses ist ausgelegt für eine hohe Anzahl an Transaktionen und parallelen Benutzerzugriffen. Darüber hinaus liegen die Vorteile in der Cluster- und *Backup&Recovery*-Fähigkeit. Somit können weitere Datenbankknoten bei steigender Last hinzugenommen werden.

II.c *Schnittstellen*

Die Anwendung TwoToTango 3T soll über eine Schnittstellen an interne Backoffice Systeme angebunden werden. Zusätzlich soll eine Schnittstelle zu einem Mail-Server eingerichtet werden, um Registrierungs-/Aktivierungs-E-Mails zu verwenden.

II.d *Schnittstellen (spätere Ausbaustufe)*

TwoToTango soll an soziale Netzwerke angebunden werden. Hierzu sind entsprechende Schnittstellen zu spezifizieren.

II.e *Vorgaben zum Betrieb*

Um einen 24/7 Betrieb zu gewährleisten, müssen nächtliche Wartungstasks spezifiziert werden, die einen reibungslosen Betrieb am Tage ermöglichen.

- Um die Ausfallsicherheit zu gewährleisten, müssen sowohl Datenbank als auch Application Server im Verbund als *Cluster* laufen. Dies ist durch entsprechende *Failover-Tests* regelmäßig zu testen.
- Um Fehler im Betriebsablauf zu identifizieren, müssen alle Anwendungskomponenten sämtliche Informationen in eine *Log*-Datei schreiben. Diese befindet sich auf dem Application Server.
- Ein *Monitoring* der Anwendung ist zunächst nicht vorgesehen.

III Konzeption

III.a *Konzeption der Realisierung*

Da es sich bei der Anwendung TwoToTango 3T um eine datenbankgestützte webbasierte Internet-Anwendung handelt, liegt der Ansatz einer Mehrschichtenarchitektur nahe. Daher wurde die Anwendung TwoToTantgo 3T im Rahmen dieses Projekts auf Basis einer 4-Schichten-Architektur konzipiert.

Dieses Konzept sorgt für eine Trennung von Anwendungsdaten, -logik und Benutzungsschnittstelle. Zusätzlich wird die darunter liegende Datenbank separat betrachtet.

In der Anwendungslogik-Schicht wird dieses Konzept ergänzt um eine Komponentenarchitektur, welche als *Java Enterprise Beans* realisiert wird.

III.b *Persistente Datenhaltung*

In der Datenschicht liegen die persistenten Anwendungsdaten in einer relationalen Datenbank. Sie sind nach der dritten Normalform normalisiert und in sich vollständig konsistent. Es gibt keine Redundanzen. Die Datenintegrität wird durch datenbanktypische Mechanismen (*Constraints*) sichergestellt. Ein Historisierung der Daten erfolgt über Historientabellen, welche im ersten Schritt nicht vorgesehen sind. Zu jeder Tabelle `TABELLE` existiert eine Historien-Tabelle `TABELLE_HIST`.

Der Zugriff auf das Schema erfolgt ausschließlich durch die *PersistenceUnit* der Anwendungsdaten und ihrer zugehörigen *DataSource*.

III.c *Anwendungsdaten*

Die Anwendungsdaten sind in Java als *Java Persistent Entities* realsiert, welche durch ein objekt-relationales Mapping auf die einzelnen Tabellen projiziert werden. Als objekt-relationales Mapping Framework wird *Hibernate* verwendet, welches die standardisierte Schnittstelle *JPA 2.0* unterstützt. Die *Hibernate* Bibliothek liegt dem verwendeten *JBoss Application Server 7* bereits in der Version 4.0 bei und muss nicht manuell installiert werden.

Java Persistent Entities sind einer *PersistenceUnit* logisch zugeordnet. Diese ist im Rahmen der Anwendung TwoToTango 3T als `TwoToTangoPU` in der Daten `persistence.xml` definiert. Sie bestimmt die darunter liegende *DataSource*, sowie SQL Dialekt (hier `OracleDialect`), den Transaktionstyp (hier `JTA`) und weitere Eigenschaften.

Die *DataSource* wird im ApplicationServer platziert und definiert die Verbindung zur Datenbank (u.a. Connection URL, Benutzername, Passwort, etc.).

III.d *Anwendungslogik*

Wie bereits beschrieben handelt es sich bei der Anwendung TwoToTango 3T um eine Mischung aus einer klassischen 4-Schichten-Architektur und einer Komponenten-Architektur. In der Anwendungslogik wird dies besonders deutlich, da diese als *Enterprise Java Beans (EJB) 3.1* realisiert sind.

Die einzelnen Komponenten (EJB) sind überwiegend als *Stateless Session Beans* umzusetzen. Die Anwendungslogik befindet sich jeweils in den Methoden und dient als Schnittstelle zur zwischen den Web-Servlets in Richtung Benutzer/Cient und den Daten in der Datenbank. Zusätzlich kapselt diese Schicht die fachlichen Entitäten der Anwendungsdaten-Schicht von der Benutzungsschnittstelle der Web-Schicht ab und stellt anstelle extra *Datentransfer-Objekte* bereit. Eine EJB ist zustandslos (*Stateless*) implementiert.

Alle EJB sind als `Local` gekennzeichnet und können daher ausschließlich aus dem *Servlet-Container* oder innerhalb des *EJBContainer* heraus aufgerufen werden.

Beispiel-Ablauf (Speichern):
1. *Servlet* instanziiert ein *Datentransfer-Objekt*
2. *Servlet* ruft eine Methode des *EJB* auf und übergibt das *Datentransfer-Objekt*
3. Das *EJB* liest die Daten aus dem *Datentransfer-Objekt* und instanziiert relevante *Java Persistent Entities*
4. *EJB* ruft *EntityManager* (PersistenceUnit) auf und übergibt *Java Persistent Entity*
5. Ende der Verarbeitung im *EJBContainer*

Beispiel-Ablauf (Laden einer Liste):
1. *Servlet* ruft Methode des *EJB* auf
2. *EJB* fragt *EntityManager* (*PersistenceUnit*) (ggf. per *Query*) ab
3. *EJB* liest Daten aus der Datenbank über den *EntityManager* ein
4. *EJB* gibt Liste der Daten zurück
5. Ende der Verarbeitung im *EJBContainer*

Beispiel-Ablauf (Detailansicht eines Objekts):
1. *Servlet* ruft Methode des *EJB* auf mit Identitätsmerkmal der Ziel-Entität
2. *EJB* fragt *EntityManager* (*PersistenceUnit*) (ggf. per *Query*) ab
3. *EJB* liest Daten aus der Datenbank über den *EntityManager* ein
4. *EJB* gibt Daten für die Detailansicht zurück
5. Ende der Verarbeitung im *EJBContainer*

III.e *Benutzungsschnittstelle (Web-Schicht)*

Die Benutzungsschnittstelle, bzw. die Web-Schicht ist im *Servlet-Container* des *JBoss Application-Servers* angesiedelt. Entgegen der empfohlenen Frontend-Technologie *JSF* wird das *Google Webtoolkit* in der Version 2.4.0 zur Realisierung der Benutzungsschnittstelle eingesetzt. Diese Entscheidung resultiert aus der Anforderung Q150, dass im späteren Verlauf eine Mobile-Schnittstelle realisiert werden soll. Hierfür ist GWT am Besten geeignet.

III.e.1 Layout Widgets

Zur Realisierung der entworfenen Mockups in GWT wird als Basis ein `DockLayoutPanel` verwendet, welches sich aufteilt in die Bereiche:

- `NORTH` (3T Logo + LoginPanel)
- `WEST` (Navigation links)
- `CENTER` (Seiteninhalt)
- `EAST` (Werbebanner)

III.e.2 Model-View-Presenter

Das vorherrschende Entwurfsmuster für GWT Applikationen ist das so genannte *Model-View-Presenter* (siehe Abbildung).

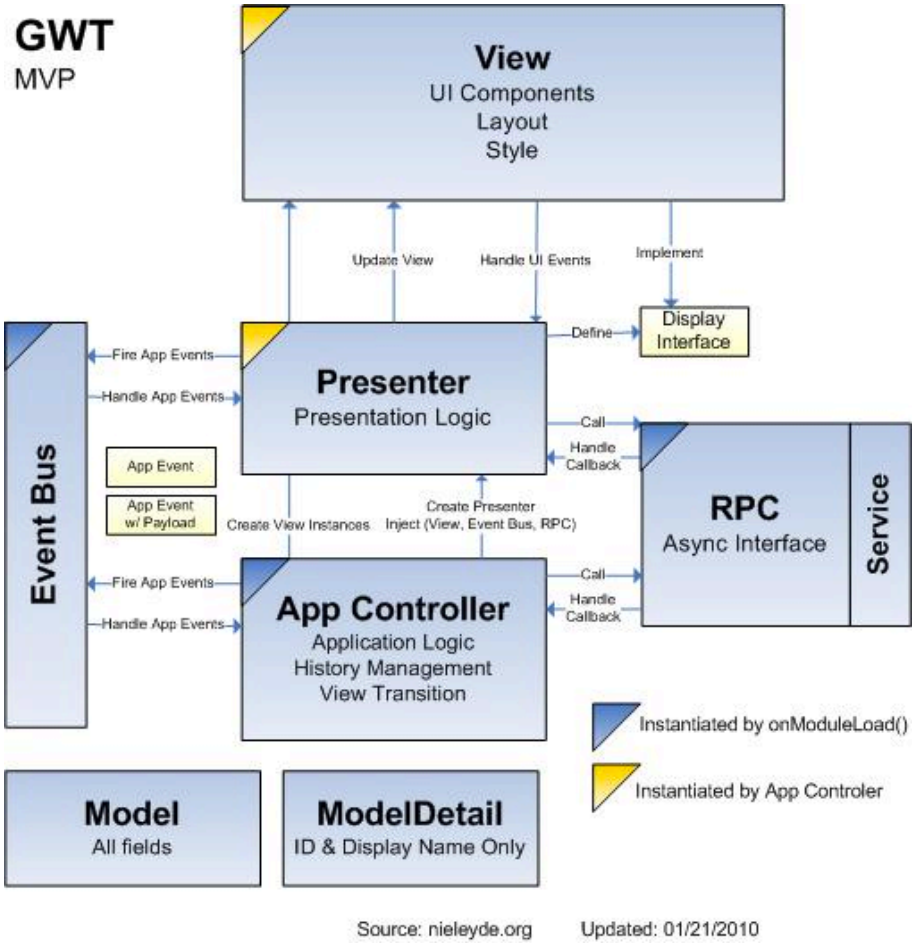

Abbildung 1: Model-View-Presenter

Ein zentraler `EventBus` lauscht auf Events, welche von `UIComponents` geworfen werden und weist `Presenter` an diese zu verarbeiten.

III.e.3 Activities & Places

Die Abbildung im Abschnitt Model-View-Presenter zeigt den App Controller, welcher sich um das *History Management* kümmert. Dies ist in GWT mittels *Activities & Places* realisiert. Eine `Activity` stellt dort den `Presenter` dar. Ein `Place` ist eine serialisierte speicherbare URL (vgl. Lesezeichen/Favoriten im Browser), welche durch GWT wieder rekonstruiert werden kann.

Beispiel:
Ein Internet-Benutzer möchte sich die Tanzschule „Lustiges Tanzbein“ in seinem Browser als Lesezeichen hinterlegen, muss durch ein serialisiertes Token (bspw. die technische ID der Tanzschule) in der URL an den GWT-Code übergeben werden, damit daraus die Seite der Tanzschule wieder rekonstruiert werden kann.

IV Auslieferung der Anwendung

Die 4-Schichten-Architektur der Anwendung TwoToTango 3T spiegelt sich ebenfalls in den ausgelieferten Artefakten wider.

Das folgende UML Deployment Diagramm verdeutlicht dies.

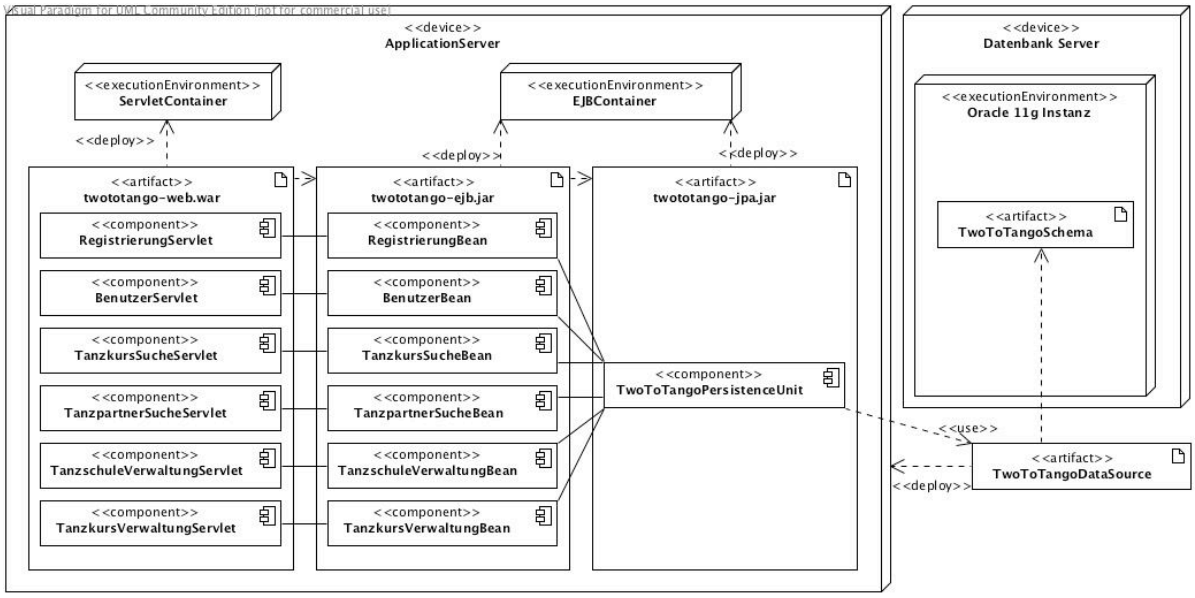

Abbildung 2: UML Deployment Diagramm

Die Auslieferung eines Releases von TwoToTango 3T beinhaltet ein `twototango.ear` Archiv, das auf dem Application Server platziert wird.

Dieses beinhaltet folgende Archive:

- **Web Archive**
  Dieses beinhaltet die serverseitigen Java *Servlets* (siehe Deployment Diagramm) und die clientseitige JavaScript Ausgabe des GWT Kompilers
- **Java Archive (Anwendungslogik)**
  Enthält die *EJB* mit fachlicher Logik und die Datentransfer-Objekte zum Austausch von Daten zwischen Datenbank und Benutzungsschnittstelle.
- **Java Archive (Datenhaltung)**
  Enthält die fachlichen Entitäten als *Java Persistent Entities*.

V Anhänge

V.a *Referenzierte Dokumente*

[Winter 2005] Mario Winter: Methodische objektorientierte Softwareentwicklung; dpunkt Verlag 2005

[Winter 2010] Mario Winter: Fortgeschrittene Softwaretechnologie LE2 Anforderungsermittlung und Systemspezifikation. Lehrbrief im Verbundstudium, IfV NRW, Hagen, 2009