www.ordix.de € 3,20

1|2017

ORDIX® news

einfach. besser. informiert.

Was ist Data Mining?

SIND SIE STARTKLAR FÜR IHRE ZUKUNFT?

WIR GEBEN IHRER ENTWICKLUNG RÜCKENWIND.

NEUGIERIG?

Als neuem Mitarbeiter bieten wir Ihnen ein speziell auf Sie zugeschnittenes Start- und Entwicklungsprogramm. Die Themen Weiterbildung und Weiterentwicklung sind ein wesentlicher Teil unserer Unternehmensphilosophie.

STUDENTEN

Wir bieten in Kooperation mit unseren Partner-Hochschulen duale Studiengänge für die Standorte Paderborn, Wiesbaden und Köln an.

Zum Beispiel:

Bachelor of Science
in Wirtschaftsinformatik

Im regelmäßigen Wechsel von Theorie und Praxis steht die Konzeption, das Design und die Entwicklung von modernen Anwendungen im Vordergrund.

YOUNG PROFESSIONALS

Wir bieten Ihnen die Möglichkeit, sich in unserem Unternehmen zu entwickeln und ermöglichen Ihnen den optimalen Karrierestart.

Zum Beispiel:

Junior Software-Entwickler – Java

An unseren ORDIX Standorten arbeiten Sie mit modernen und innovativen Technologien und unterstützen unsere Experten in der Entwicklung von komplexen Applikationen.

PROFESSIONALS

Sie suchen eine neue berufliche Herausforderung? Auf Sie warten spannende und anspruchsvolle IT-Projekte.

Zum Beispiel:

IT-Projektmanager

Die Planung, Organisation und Steuerung interessanter IT-Projekte bei unseren renommierten Kunden stehen im Fokus Ihres Aufgabenfeldes.

Weitere Informationen und Jobangebote finden Sie auf unseren Karriereseiten im Internet.

kununu TOP COMPANY www.kununu.com VON MITARBEITERN EMPFOHLEN!

www.ordix.de/karriere
www.xing.com/companies/ordixag/jobs

Paderborn, Mai 2017

Versprochen ist versprochen

Jetzt habe ich extra die Wahlen in Nordrhein-Westfalen (NRW) abgewartet, damit ich nichts politisches vorher schreibe und somit vielleicht irgendjemanden (un-)bewusst beeinflusse. Nun sind #NRWir völlig abgestürzt und werden von der FDP Ein-Mann-Show, die nach drei Monaten Düsseldorf dann auch nur noch aus Berlin mitregieren will, bestimmt.

Besser ich sage dazu nichts mehr, aber ich habe unserem Redakteur ja versprochen, dass ich heute liefere. Also liefere ich! Versuchen wir es mal mit Weltpolitik: Trump? Oh nein lieber doch nicht. Das beleidigte Kleinkind stampft trotzig mit dem Twitter-Fuß auf und sagt, als Präsident darf ich jederzeit Geheimnisse verraten und ich kann doch nicht alle, die diese Hexenjagd auf mich machen, entlassen. Der Arme, vermutlich würde er sie ja lieber wie Erdogan verhaften und einsperren. Der von Merkel ernannte Retter der Flüchtlinge läuft ja leider mittlerweile auch noch per Volksabstimmung legitimiert innen- wie außenpolitisch Amok. Und der größte Schock aus Deutschland erfolgte die Legitimation mit sehr großer Mehrheit.

Ich habe mich für die deutschen Türken fremdgeschämt. Aber ich muss aufpassen, was ich sage. Politische Korrektheit nennt man das. Das ist ganz wichtig. Genauso wichtig, wie Entbürokratisierung und schlankerer Staat, schreiben sich immer alle vor der Wahl auf die Flagge. Danach kommt aber genau das Gegenteil heraus. Oh der WDR 2 hat das Thema Entbürokratisierung auch gerade im Programm …

Bekam ich früher (so Mitte bis Ende der Neunziger) zwei bis drei Einladungen pro Jahr zu Weiterbildungen für Geschäftsführer oder Vorstände zu rechtlichen und vor allem steuerrechtlichen Veranstaltungen, liege ich heute bei ca. vier pro Monat (!). Das ist jetzt nicht einem intensiverem Marketing der Schulungsanbieter geschuldet, sondern ausschließlich der Flut an unübersichtlichen neuen Gesetzen, Gesetzesvorhaben, Erlassen und Interpretationen zu Gesetzen, die eigentlich schon von Anfang an Unfug waren.

Die, die aber Unrecht begehen (ich sage z.B. Diesel-Gate), kommen mit ministerieller Erlaubnis (Dobrindt) oder besser Blindheit aus der Klemme. Oh aufpassen, vermutlich wieder politisch nicht korrekt. Vielleicht sollte ich mir mal einen Twitter Account zulegen und wie Trump die Welt mit allem Mist, den keiner hören will, zu ballern. Oder genauso schön und sinnlos auf Instagram und Facebook posten.

Nein das ist nicht mein Ding. Mir fällt es ja schon schwer in diesen AfD- (wobei die hat sich vielleicht doch bald selbst erledigt?!?), Trump-, Wilders-, Le Pen-Zeiten hier ein politisch korrektes Editorial zu schreiben. Es lebe Macron und wir brauchen eigentlich auch mal Erneuerung da oben in Berlin. Ooops aufpassen, da kommt dann die Ein-Mann-Partei, der regiert dann in Düsseldorf und in Berlin. Naja viel schlimmer als jetzt kann es auch nicht werden. *)

Das ist jetzt alles Big Data und das gibt es auch in dieser News. Wir bei ORDIX haben schon sehr frühzeitig in diese Technologien investiert und eine Reihe von Mitarbeitern ausgebildet, die Ihnen gleich drei Artikel dazu geschrieben haben.

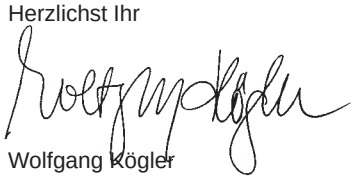
Natürlich gibt es auch in dieser Ausgabe, neben Big Data, etwas zu klassischen Datenbanksystemen (Oracle). Aus dem System Bereich haben wir dieses Mal nur einen Artikel und das ist wirklich nach so vielen Jahren ein Novum: Kein Linux, kein Solaris sondern unser Autor präsentiert Microsoft mit dem System Center Configuration Manager.

Last but not least auch ein breites Spektrum aus dem Entwicklungsumfeld, über Entwurfsmuster, ein JavaScript Framework bis hin zum immer wichtiger werdenden Thema Continuous Integration.

Gepaart mit unserem sozialen Engagement und unserem Angebot an Weiterbildungen in der Heftmitte ist diese ORDIX News mindestens so interessant wie die Ein-Mann-Show im Politikzirkus in Düsseldorf und Berlin.

So Herr Pothmann, versprochen ist versprochen, hiermit habe ich mein Editorial abgeliefert. Mal sehen was die neue Regierung in Düsseldorf abliefern wird.

Herzlichst Ihr



Wolfgang Kögler

*) Einer meiner Aufsichtsräte und ich haben ein unterschiedliches Empfinden in Bezug auf den FDP (DEN Freien Demokratischen Politiker), obwohl wir dringend, nicht nur in Deutschland und Europa, eine mehr und mehr liberale und soziale Politik benötigen. Denn auch maximale Sicherheit bekommt man durch Vernunft und Freiheiten und nicht durch Überwachen, Einsperren und Beobachten. Das gilt im realen Leben wie auch in der IT. Also gehen Sie im September wählen, es ist (hoffentlich) alles möglich.

Was ist Data Mining?

Redis: Key Value einmal anders

Big Data

Web und Application-Server

Oracle

Microsoft

Magier im Verborgenen

Data Mining – Klassifikation

Entwicklung

Aktuell

Impressum

Herausgeber: ORDIX AG Aktiengesellschaft für Softwareentwicklung, Beratung, Schulung und Systemintegration, Paderborn

Redaktion/Layout: Jens Pothmann, Isabell Rosenblatt

V.i.S.d.P.: Christoph Lafeld, Wolfgang Kögler

Anschrift der Redaktion: ORDIX AG | Karl-Schurz-Straße 19a | 33100 Paderborn
Tel.: 0 52 51 10 63 -0 | Fax: 0180 1673490

Auflage: 7.000 Exemplare

Druck: Druckerei Bösmann, Detmold

Bildnachweis:
© pixabay | dimitrisvetsikas1969 | Working Man Sculptor ...
© freepik.com | Onlyyouqj | Marble platform in front of ...
© freepik.com | @stockvault | Cracked surface
© freepik.com | @kjpargeter | Hydraulic hammer design
© istockphoto.com | cybrain | Globale Datenspeicherung
© deviantart.com | xxmysterystockxx | Friesian Stallion stock 15
© istockphoto.com | egal | Stone road im magischen Wald
© deviantart.com | skydancer-stock | Wizard Afternoon 20
© wikimedia.org | Carsten Jünger | Titanic at Minimundus
© freepik.com | creativart | Shadow drawing cool cheering...
© freepik.com | d3images | Character playing with a float
© istockphoto.com | your_photo | Think different

Autoren: Dr. Hubert Austermeier, Thilo Fleischhauer, Sebastian Grimm, Frank Heimerzheim, Sebastian Herd, Dr. Stefan Koch, Wolfgang Kögler, Phillip Kürsten, Holger Wegert, Klaus Reimers, Isabell Rosenblatt

Data Mining in der Praxis (Teil I)

Was ist Data Mining?

Daten sind das Unternehmenskapital der Zukunft. Wertvoll wird dieses Kapital allerdings erst, wenn es sinnvoll eingesetzt wird. Data Mining ist eine Ansammlung von Prozessen und Methoden, die dazu dienen, Erkenntnisse aus Daten zu ziehen und diese zur Wertschöpfung einzusetzen. Diese neue Artikelserie zum Thema Data Mining gibt mit dem ersten Artikel einen Überblick über das Thema und wird mit anwendungsnahen Beschreibungen einzelner Methoden des Data Mining fortgesetzt.

Wozu dient Data Mining?

Für den, aus dem englischen Sprachraum stammenden, Begriff Data Mining existiert keine etablierte deutsche Übersetzung. Der Begriff „Mustererkennung in Daten“ kommt dem mit Data Mining verfolgten unternehmerischen Ziel am nächsten.

Genau darum geht es beim Data Mining: In den immer weiter wachsenden Bergen von Daten in Unternehmen sollen Zusammenhänge entdeckt und aufbereitet werden, so dass ein unternehmerischer Wert aus diesen Informationen generiert werden kann.

War die Fähigkeit hierzu vor einigen Jahren noch ein Merkmal, welches einem Unternehmen einen Wissensvorsprung vor den Mitbewerbern verschafft hat, ist der Einsatz von Techniken aus dem Bereich Data Mining inzwischen eine unternehmerische Notwendigkeit, um keine Wettbewerbsnachteile befürchten zu müssen. Wissen ist zum Kapital geworden. Wertvoll ist dieses Kapital allerdings nur durch die Fähigkeit, es auch gewinnbringend einzusetzen.

Auf Datenschutzaspekte, sonstige gesetzliche Rahmenbedingungen und ethische Fragestellungen wird in diesem Artikel bewusst nicht eingegangen.

In Europa, speziell in Deutschland, sind die Standards und Erwartungen der Kunden bezüglich dieser Aspekte deutlich stärker ausgeprägt als z.B. in den USA. So wichtig diese Fragen auch sein mögen, für das Verständnis, der im Zusammenhang mit Data Mining stehenden Prozesse und Verfahren, sind sie nicht direkt relevant.

Phasen eines Data-Mining-Prozesses

Ein typischer Ablauf zur Erkennung von Mustern in Daten umfasst fünf Schritte:

- **Selektion der untersuchten Daten**
  Die zu untersuchenden Daten müssen klar definiert werden. Insbesondere bei diesem Schritt gilt es, Datenschutzaspekte zu beachten.
- **Datenbereinigung**
  Unvollständige Datensätze müssen ergänzt oder entfernt werden.
- **Transformation**
  Aufbereitung der Daten für den eigentlichen Analyseschritt, z.B. durch Diskretisierung von Werten.
- **Data Mining**
  Dies ist der eigentliche Analyseschritt.
- **Interpretation und Evaluation**
  Aufbereitung der gefundenen Erkenntnisse und Interpretation durch Experten.

Die Selektion der zu untersuchenden Daten und die Interpretation der Ergebnisse liegen in der Verantwortung der für die Daten verantwortlichen Fachabteilung. Die Schritte der Datenbereinigung und Transformation liegen in vielen Unternehmen in der Verantwortung eines bereits eingesetzten Data Warehouse. Für den Bereich der eigentlichen Analyse wird spezielles Wissen benötigt, das nicht zwingend im Unternehmen vorhanden sein muss. Neben fundierten mathematischen und insbesondere statistischen Kenntnissen, sollten die eingesetzten Fachkräfte auch solide Erfahrung als Programmierer aufweisen.

Aufgaben des Data Mining

Die möglichen Fragestellungen, die mithilfe von Data Mining beantwortet werden sollen, lassen sich grob in fünf Bereiche aufteilen:

- Klassifikation
- Regressionsanalyse
- Assoziationsanalyse
- Ausreißererkennung
- Clusteranalyse

Über diese fünf Bereiche hinaus ist es eine grundsätzliche Aufgabe von Data Mining, aus Daten Zusammenfassungen (Aggregationen) zu erstellen. Die häufig sehr großen Datenmengen werden bezüglich bestimmter Fragestellungen in eine überschaubare Form aufbereitet. Dies kann durch die Berechnung von Kennzahlen, wie zum Beispiel dem Durchschnittsumsatz eines Monats, oder auch durch Visualisierungen, wie zum Beispiel einem Histogramm der Umsätze, erfolgen.

Klassifikation

Mithilfe von vorhandenen Daten, oft auch als historische Daten bezeichnet, wird ein Modell bezüglich einer speziellen Fragestellung entwickelt. Eine solche Fragestellung kann zum Beispiel sein, ob eine Mail als Spam eingestuft werden soll oder nicht. Mithilfe des Modells kann nun für eine neue Mail anhand des Inhalts automatisch ermittelt werden, welcher Kategorie sie zugeordnet werden soll.

In dieser Ausgabe der ORDIX® news befindet sich ein weiterer Artikel zum Thema Klassifikation inklusive Beispielcode zur Umsetzung des präsentierten Beispiels. In den folgenden Ausgaben wird in der Reihe Data Mining insbesondere auf die Regressions- und Clusteranalyse eingegangen.

Regressionsanalyse

Auf Basis von vorhandenen Daten, wie z.B. der Umsatz eines Kunden und seinem Wohnort, wird eine Kennzahl ermittelt. Diese Kennzahl kann zum Beispiel der mit diesem Kunden in Zukunft zu erwartende Umsatz sein. In der Regel sind derartige Beziehungen zwischen einer Anzahl von numerischen Eingangsgrößen und einer numerischen Ausgangsgröße nicht einfach zu ermitteln. Die Regressionsanalyse erstellt aus historischen Daten ein Modell, das genau diese Zusammenhänge beschreibt.

Assoziationsanalyse

In vorhandenen Daten werden Zusammenhänge zwischen verschiedenen Datensätzen untersucht und hieraus Regeln abgeleitet. Ein bekanntes Beispiel ist die Analyse der Kassenbons eines Supermarktes, laut der „Windeln“ und „Bier“ häufig an einem Samstag zusammen verkauft wurden. Es liegt also nahe, an Wochenenden Paletten mit Bier neben Paletten mit Windeln aufzustellen, um die Umsätze zu steigern. Ein solcher Zusammenhang kann, auch für deutlich komplexere Situationen, anhand der Assoziationsanalyse ermittelt werden.

Ausreißererkennung

Mithilfe von Verfahren zur Analyse von Ausreißern kann aufgrund historischer Daten für jeden neuen Datensatz eine Wahrscheinlichkeit angegeben werden, mit der er ein Ausreißer ist. Ein Ausreißer ist ein Datensatz, der signifikant von anderen gemessenen Daten abweicht. Ein vom Kunden angegebenes Alter von 200 Jahren ist offensichtlich unmöglich und somit falsch. Diese Information kann dazu dienen, solche Datensätze automatisch zu löschen oder zur manuellen Prüfung zu sammeln. Neben dem Ziel der Datenbereinigung können gefundene Ausreißer bei der Betrugsfallerkennung zum Beispiel aufgrund ungewöhnlicher Kontenbewegungen wertvolle Informationen liefern.

Clusteranalyse

Bei der Klassifikation sind Kategorien vorgegeben. Bei der Clusteranalyse ist es Aufgabe des Verfahrens, neue

Seminare zum Thema

[1] Big Data: Informationen neu gelebt | Apache Hadoop Grundlagen
https://seminare.ordix.de/seminare/big-data-und-data-warehouse/big-data

Bildnachweis

Frank Heimerzheim
(info@ordix.de)

Kategorien zu ermitteln. So können zum Beispiel durch die Analyse von Warenkörben einer Vielzahl von Onlinekunden automatisiert Gruppen von Kunden mit speziellen Interessen ermittelt werden. Diese wiederum lassen sich gezielt mit Werbemaßnahmen ansprechen. Besonders interessant sind hierbei Zusammenhänge, die auch eine Fachabteilung bislang nicht vermutet hat und die gänzlich neue unternehmerische Sichtweisen aufzeigen können.

Fazit

Dieser Artikel gibt einen ersten Überblick über die verschiedenen Aufgaben und damit verbundenen Prozesse, die den Begriff Data Mining ausmachen. In jedem Fall werden vorhandene Daten analysiert und aus gefundenen Zusammenhängen Modelle entwickelt, die in der Lage sind, spezielle Fragestellungen zu beantworten.

Auch wenn die meisten eingesetzten Prozesse in vielen Unternehmen bereits vorhanden oder einfach zu etablieren sind, erfordert die Kernkompetenz der analytischen Untersuchung der Daten spezielles Wissen. In weiteren Artikeln in der ORDIX® news wird mit anwendungsnahen Codebeispielen auf Verfahren des Data Mining eingegangen. Gerne stehen Ihnen unsere Experten, sowohl zur Beratung als auch zur Umsetzung zur Verfügung.

Data-Access-Objekte (DAO)

BEST-P: Find-By-Example

Der Zugriff auf die Datenbank ist für Enterprise-Anwendungen von zentraler Bedeutung. Neben der fachlichen Konsistenz spielen Aspekte wie Performance und Security eine wesentliche Rolle. Umfangreiche Anwendungen behandeln solche Aspekte in einer eigenen Persistenz-Schicht, die aus Data-Access-Objekten (DAOs) gebildet wird. In Java können diese Klassen mit JPA und dem Design-Pattern Find-By-Example generisch erstellt werden: Eine Klasse enthält die Logik für alle DAOs.

Ein Credo für DAOs

DAO ist eine Abkürzung für das Design-Pattern Data Access Object [1]. Damit wird der Zugriff auf ein Informationssystem – meistens eine Datenbank – gekapselt. Auch wenn das Design-Pattern nicht unumstritten ist [2], gibt es für dessen Nutzung gute Gründe.

- **Single-Responsibility-Prinzip:**
  DAOs konzentrieren sich auf die Interaktion mit der Datenbank. Ihre Aufgabe ist es, Daten aus Objekten in die Datenbank zu überführen und umgekehrt.
- **Don't-Repeat-Yourself:**
  Zugriff-Operationen werden einmalig in der DAO definiert. Auf die `DAO`-Methoden wird im Code vielfach zugegriffen.
- **Testbarkeit:**
  Zugriffe auf die Datenbank lassen sich unabhängig von fachlichen Services testen.
- **Zugriffsbeschränkungen:**
  In der DAO lässt sich beispielsweise verhindern, dass mal eben 100 Millionen Datensätze eingelesen werden.

Und dann tauchen in Projekten nicht selten auch Aspekte auf, die ohne DAOs nur schwer zu bewältigen sind.

- **Performance:**
  Die Ausführungsgeschwindigkeit kann in dem DAO gemessen und verbessert werden.
- **Security:**
  Der Zugriff auf Daten lässt sich an dieser zentralen Stelle kontrollieren.
- **Mandantenfähigkeit:**
  In Abhängigkeit von ihren Berechtigungen bekommen Anwender nur einen Teil der Daten zu sehen.
- **Archivierung von Alt-Daten:**
  Jede Änderung von Daten soll persistent nachvollziehbar sein.
- **Fachliche Konsistenz-Prüfung:**
  Die Datenbank stellt auch eine Eingangs-Schnittstelle zur Anwendung dar. Manchmal muss die Qualität der eingelesenen Daten geprüft werden.

Sicher lassen sich alternative Design-Patterns finden oder konstruieren, um die oben genannten Aspekte zu berücksichtigen. Doch spricht die normative Kraft des Faktischen für das DAO: Jeder Java-Enterprise-Entwickler kennt es.

Nachteile von DAOs

Der Zugriff auf die Datenbank ist Dank der Java Persistence API (JPA) alles andere als spannend: Die Implementierung der CRUD-Operationen sieht für alle DAOs nahezu gleich aus.

Um diesen Code nicht in jede DAO-Klasse schreiben zu müssen, bietet sich eine generische DAO-Klasse an, die als `GenericDaoImpl` in Abbildung 1 definiert ist. Diese Klasse ist in dem Beispiel für den Einsatz in einer Java EE-Umgebung gedacht – der `EntityManager` wird in Folge der Annotation `@PersistenceContext` injiziert. Für die CRUD-Operationen stehen die Methoden `save`, `getById`, `update` und `delete` zur Verfügung. Damit ist der Grundbaustein für ein DAO gelegt.

Find-By-Example

Der Großteil der Arbeit besteht in den DAO-Klassen darin, die gewünschten `find`-Methoden zu implementieren. Beispielsweise können Personen nach ihrem Nachnamen gesucht werden und Kunden möchte man anhand des Umsatzes suchen. Darüber hinaus gibt es auch noch Variationen dieser Suche. Neben dem Nachnamen wird auch der Vorname oder das Geburtsdatum angegeben.

Durch derartige Anforderungen entsteht im Laufe der Entwicklung eine Vielzahl von `find`-Methoden. Eine gut bekannte Strategie zur Selektion von Daten nach unterschiedlichen Suchbegriffen ist das Design-Pattern

Find-By-Example (siehe Abbildung 2). Die Suche wird in einem Example-Objekt in der Weise codiert, dass die Suchkriterien als Attribut angegeben werden. Mit **`Example.person="Koch"`** sucht der **`CriteriaBuilder`** in der Personen-Tabelle nach den entsprechenden Datensätzen. Enthält das **`Example`**-Objekt mehr als ein Such-Attribut, so werden alle Attribute mit einem logischen "Und" verknüpft bei der Selektion berücksichtigt. Als Ergebnis der Suche werden die gefundenen Entity-Objekte zurückgeliefert.

```
public abstract class GenericDaoImpl<E, P extends
Serializable> implements Dao<E, P> {
    @PersistenceContext(unitName = "exampleds")
    protected EntityManager entityManager;
    private final Class<E> type;

    public GenericDaoImpl(Class<E> clazz) {
        this.type = clazz;
    }
    protected void detach() {
        entityManager.flush();
        entityManager.clear();
    }

    @Override
    public E save(E object) {
        entityManager.persist(object);
        detach();
        return object;

    }
    @Override
    public void delete(E object) {
        E foundObject = entityManager.merge(object);
        entityManager.remove(foundObject);
        detach();
    }
    @Override
    public E getById(P id) throws NoResultDaoException {
        E result = entityManager.find(type, id);
        detach();
        if (result == null) {
            throw new NoResultDaoException();
        }
        return result;
    }
    @Override
    public E update(E object) {
        E returnValue = entityManager.merge(object);
        detach();
        return returnValue;

    }
```

Abb. 1: GenericDaoImpl: Generische Dao-Klasse

Abb. 2: Prinzip des Find-By-Examples

Das Design-Pattern Find-By-Example ist optimal geeignet, wenn in der GUI ein Suchdialog angeboten wird. Der Anwender entscheidet, welche Attribute für die Suche herangezogen werden. Auch in Services sind viele Selektionen mit Find-By-Example abzubilden. Die Anzahl der zu implementierenden **`find`**-Methoden in der DAO lässt sich dadurch drastisch reduzieren.

Die Idee für dieses Design-Pattern, das auch als Query-By-Example bezeichnet wird, ist bereits zu Beginn der ORM-Frameworks aufgegriffen und beispielsweise in Hibernate oder EclipseLink implementiert worden. Einen Überblick über die Möglichkeiten und Beschränkungen wird in [3] gegeben. JPA stellt keine standardisierte Schnittstelle für diese Art der Suche zur Verfügung.

Implementierung mit der Criteria API

Die bestehenden Lösungen von Hibernate und EclipseLink lassen noch folgende Wünsche offen.

- Suche in assoziierten Objekten
- Verwendung von Relationen (>, <, >=, <=) bei numerischen oder Datumsangaben
- Angabe von Intervallen bei numerischen oder Datumsangaben
- Suche in Zeichenketten mit Like

In unserem Projekt haben wir uns für eine eigene Implementierung von Find-By-Example auf Basis der Criteria-API entschieden. Maßgebend war die Unabhängigkeit vom JPA-Provider und vor allem die Erweiterbarkeit: Es ist absehbar, dass weitere Anforderungen, wie die Suche nach **`NULL`** oder **`NOT NULL`**, die Negierung der Bedingung oder die Angabe einer Wertemenge dazukommen werden.

In der eigenen Lösung ist das Example-Objekt ein Pojo, dessen Attribut-Namen mit dem der Entity übereinstimmen. Anstelle der primitiven Datentypen sieht die Example-Klasse den Typ **`String`** vor, um Suchbegriffe angeben zu können. Anstelle der Assoziationen werden auch für die assoziierten Objekte Example-Objekte angegeben.

In Abbildung 3 ist ein JUnit-Test dargestellt, in dem das Find-By-Example getestet wird. Ausgangspunkt ist ein Personal-Datensatz, der über eine Datei geladen wurde. Dieser Zusammenhang wird durch die Assoziation des **`Personal-`** und **`Upload-`**Objekts abgebildet. In dem Test wird dazu ein Upload-Objekt mit dem Dateinamen „meine-Datei“, dem Personal-Objekt übergeben und mit diesem abgespeichert.

In der Suche sollen alle Personal-Datensätze gefunden werden, die über eine Datei mit dem Muster „meine%“ geladen wurden. Das Muster wird mit dem Attribut **`Dateiname`** einem **`UploadExample-`**Objekt übergeben.

Danach wird das **UploadExample**-Objekt einem **PersonalExample**-Objekt übergeben. Dadurch ist die Suche spezifiziert. Der Test findet den zuvor gespeicherten Datensatz – anschließend wird der Gegentest gemacht, indem für den Dateinamen „falscher Dateiname“ als Suchbegriff angegeben wird.

Das Prinzip für die Eigenimplementierung lässt sich im Listing der Abbildung 4 nachvollziehen. Die Attribute der Entity und die der Example-Klassen werden in Key-Value-Paaren zerlegt. Das Ergebnis sind **targetTypes** und **exampleProperties.entrySet**. Für jede Property des Example-Objekts wird dann eine Criteria in der For-Each-Schleife von **addCriteria** erstellt. Hierbei ist zunächst der Datentyp des Example-Attributs bedeutsam. Im Regelfall ist der Typ ein String. Aufgrund des Inhalts und des Datentyps des Entity-Attributs wird eine passende Criteria erstellt.

Ist der Attribut-Typ der Example-Klasse ein Enum, wird einfach eine Criteria für den Vergleich der Enums erstellt. In allen anderen Fällen muss es sich um ein assoziiertes Objekt handeln. Für diesen Fall wird die Method **addCriteria** rekursiv aufgerufen. Die vollständige Implementierung können Sie in unserem Blog [4] herunterladen.

Fazit

Das Design-Pattern Find-By-Example hat sich in unseren Java-Enterprise-Anwendungen bewährt. Durch diesen generischen Ansatz lässt sich nicht selten ein DAO vollständig generisch erzeugen. Die Eigenimplementierung finden Sie in Form eines Maven-Projekts in unseren Blogbeiträgen [4]. Für weitere Fragen rund um das Thema Find-By-Example kontaktieren Sie unsere Experten unter 0 52 51 / 10 63 -0.

Dr. Stefan Koch
(info@ordix.de)

Links/Quellen

[1] Core J2EE Patterns - Data Access Object, Oracle:
http://www.oracle.com/technetwork/java/dataaccessobject-138824.html

[2] The DAO Anti-Patterns, RRees:
https://rrees.me/2009/07/11/the-dao-anti-patterns/

[3] Hibernate Query By Example, Donat Szilagyi:,
https://dzone.com/articles/hibernate-query-example-qbe

[4] ORDIX Blog:
https://blog.ordix.de/technologien/best-p-find-by-example.html

```
@Test
public void findPersonalAnhandUpload()
   throws NoResultDaoException {
   Personal personal = createPersonalWithUpload(
      "Koch", "Stefan", "meineDatei");
   personal = save(personal);

   UploadExample upExample = new UploadExample();
   upExample.setDateiname("meine%");
   PersonalExample pExample = new PersonalExample();
   pExample.setUpload(upExample);
   List<Personal> ergebnisListe =
      findByExample(pExample);
   assertEquals(1, ergebnisListe.size());
}
@Test(expected=NoResultDaoException.class)
public void dontFindPersonalAnhandUpload()
   throws NoResultDaoException {
   Personal personal = createPersonalWithUpload(
      "Koch", "Stefan", "meineDatei");
   personal = save(personal);

   UploadExample upExample = new UploadExample();
   upExample.setDateiname("falscher Dateiname");
   PersonalExample pExample = new PersonalExample();
   pExample.setUpload(upExample);
   findByExample(pExample);
   fail("Exception wird erwartet");
}
```

Abb. 3: PersonalDaoImplTest – Suche nach Personal mit Find-By-Example

```
protected void addCriteria(Path<E> property
   , Object value, Map<String, Class> targetTypes) {
   Map<String, Object> exmampleProperties =
      propertyMap(value);
   for (Entry<String, Object> entry :
   exmampleProperties.entrySet()) {

   String propertyName = entry.getKey();
   Object searchObject = entry.getValue();
   Class targetClass = targetTypes.
get(propertyName);
   if (targetClass == null) {
      throw new IllegalStateException();
   }
   if (searchObject == null) {
      continue;
   }

   if (searchObject instanceof String) {
      if (targetTypes.get(propertyName) != null) {
      addCriteria4String(property.get(propertyName),
         (String) searchObject, targetClass);
      }
   } else if (searchObject.getClass().isEnum()) {
      addCriteria4Enum(property.get(propertyName),
         searchObject, targetClass);
   } else {
      addCriteria(property.get(propertyName),
      searchObject, getTargetTypes(targetClass));
   }
   }
}
```

Abb. 4: CriteriaBuilder-Auszug

Big Data – Informationen neu gelebt (Teil VI):

Redis: Key Value einmal anders

Im Kontext von NoSQL-Datenbanken denken viele mit Apache Cassandra, MongoDB und Neo4j zunächst an die klassischen Vertreter der Wide Column Stores, Document Stores oder der Graphdatenbanken. Key Value Stores werden dagegen, trotz ihrer besonderen Eigenschaften, nur wenig berücksichtigt. In diesem Artikel wird daher der Fokus auf diese Kategorie von NoSQL-Datenbanken gelegt, zudem wird mit Redis ein vielseitiger Vertreter vorgestellt und dabei dessen Stärken sowie Schwächen beleuchtet.

Eingruppierung

Bei Redis (Remote Dictionary Server) handelt es sich um eine In-Memory-NoSQL-Datenbank, welche unter der Open-Source-Lizenz BSD entwickelt wird. Grundsätzlich lässt sich die Datenbank den sogenannten Key Value Stores zuordnen. Dies bedeutet, dass ein Datensatz grundsätzlich auf Basis eines Keys eindeutig identifiziert wird und einen beliebigen Value besitzt.

Im Gegensatz zu klassischen KeyValue Stores zeichnet sich Redis durch die zusätzliche Unterstützung erweiterter Datenstrukturen aus. Aufgrund dieser Besonderheit bezeichnet sich das Projekt auch selbst viel mehr als sogenannter Data Structure Store. Ursprünglich wurde die Datenbank von Salvatore Sanfilippo im Jahre 2009 entwickelt, der seitdem durch Unternehmen wie VMware, Pivotal und Redis Labs aktiv dabei unterstützt wird. Seit jeher liegt der Fokus auf der Durchführung schneller Schreib- bzw. Leseoperationen und der platzsparenden Speicherung einfacher Datenstrukturen. Dies macht Redis vor allem für jene Anwendungsfälle interessant, in denen der Performanz eine übergeordnete Priorität zugeordnet wird. Hierzu zählen beispielhaft die temporäre Speicherung von Warenkorbinformationen eines Webshops, die Realisierung eines Customer Support Chats oder das Caching häufig abgefragter Informationen, wie die einer Benutzerkonfiguration.

Datenmodellierung

Wie eingangs bereits erwähnt, handelt es sich im Grunde genommen auch bei Redis um einen einfachen Key Value Store. Als Datentyp, wird sowohl für den Key als auch

für den Value, ein binary safe String verwendet. Dadurch kann der Inhalt entweder aus einer einfachen Zeichenkette (z.B. „Hallo Welt“) oder einem binären Objekt (beispielsweise eine Bilddatei) bestehen. Die Größe des Strings ist sowohl für den Key als auch für den Value gleichermaßen auf 512 MB begrenzt und bietet somit auch ausreichend Speicherplatz für größere Inhalte. Zusätzlich zu den Strings erlaubt Redis auch die Verwendung einfacher Datenstrukturen innerhalb eines Values (siehe Abbildung 1), wodurch die Einsatzmöglichkeiten gegenüber klassischen Key Value Stores erweitert werden.

Mittels einer Liste kann eine beliebige Menge von Strings zu einem Key gespeichert werden. Bedingt durch die Implementierung, auf Basis einer verketteten Liste, bleibt der Aufwand beim Hinzufügen eines Strings am Anfang oder Ende der Liste konstant. Entsprechend eignet sich eine Liste gerade für Anwendungsszenarien, in denen eine hohe Anzahl an Strings sehr schnell hinzugefügt bzw. entfernt werden muss (zum Beispiel für die Kommentarfunktionalität eines Blogs). Weniger geeignet ist diese Datenstruktur für Anwendungsfälle, in denen ein freier Zugriff auf Elemente innerhalb der Liste erforderlich ist.

Ein Set dient der Speicherung eindeutiger Elemente innerhalb einer Menge von Strings. Gegenüber der Liste verfügt diese Datenstruktur über keine Möglichkeit des Zugriffes mittels eines Positionsindizes. Stattdessen wird über den String selbst zugegriffen. Beim Sorted Set wird jeder String um einen zusätzlichen Fließkommawert ergänzt. Dieser sogenannte Score ermöglicht eine sortierte Speicherung und in der Konsequenz die Durchführung effizienter Bereichsabfragen.

Die Datenstruktur Hash entspricht einer klassischen Hashtabelle und erlaubt die Ablage von separaten Key-Value-Paaren zu einem Key. Bedingt durch das zugrundeliegende Hashing eignet sich diese Datenstruktur vor allem in Anwendungsfällen, in denen Punktabfragen sehr schnell und zudem unabhängig von der darin enthaltenen Datenmenge erfolgen sollen.

Die Bitmap erlaubt einen Bit-weisen Zugriff auf Basis eines Positionsindizes und somit die platzsparende Speicherung von Informationen. Dies kann zum Beispiel dazu verwendet werden, Einstellungen von Nutzerprofilen (Newsletter ja/nein, Funktion X ja/nein, …) effizient zu speichern. Bedingt durch die Implementierung auf Basis des Strings, ergibt die maximale Größe von 512 MB eine Menge von bis zu 2^{32} Bits innerhalb einer Bitmap.

Mit HyperLogLog verfügt Redis zudem über einen Datentypen, der dem effizienten Zählen eindeutiger Strings dient. Das Besondere hierbei ist, dass der Speicherbedarf unabhängig von der Menge zu zählender Strings ist und lediglich 12 KB beträgt. Möglich ist dies aufgrund der Implementierung des gleichnamigen Approximationsalgorithmus, welcher nicht die zu zählenden Strings selbst speichert, sondern auf einer probabilistischen Datenstruktur basiert. Sinnvoll ist die Verwendung von HyperLogLogs, sofern eine Abfrage der gezählten Strings selbst nicht erforderlich ist und eine Abweichung bei der Zählung von bis zu 0,81 % toleriert werden kann. Zu den möglichen Anwendungsszenarien zählt unter anderem das effiziente Zählen von Webseitenaufrufen durch eindeutig identifizierte Besucher.

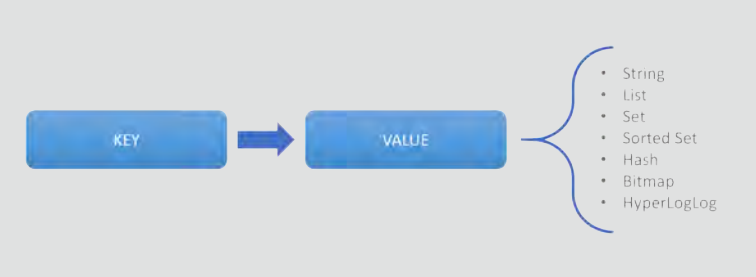


Abb. 1: Redis-Datenstruktur

Datenzugriff

Statt einer eigenen Abfragesprache verfügt Redis über eine komplexe API, welche für die unterschiedlichen Datenstrukturen separate Funktionsaufrufe bereitstellt. Die Implementierung spezialisierter Methoden erlaubt dem Entwickler stets jene Methode zu verwenden, welche im jeweiligen Anwendungskontext eine schnellstmögliche Ausführung erlaubt. Für den Zugriff auf die Datenbank werden Bibliotheken zu mehr als 40 Programmiersprachen angeboten. Abgedeckt werden hierbei klassische Sprachen wie Java, Python, C++, C#, Scala, Perl und PHP, aber auch Exoten wie Matlab oder R.

Der Verzicht auf eine komplexe Abfragesprache wie SQL führt dazu, dass ein Teil der gewohnten Funktionalität (Selektion, Projektion, Gruppierung, Sortierung, analytische Funktionen, …) durch den Entwickler auf Anwendungsseite entweder selbst implementiert werden muss oder durch die geeignete (ggf. redundante) Datenspeicherung in unterschiedlichen Datenstrukturen erreicht werden kann. Letztlich führt somit der bei Redis gewonnene Gewinn an Performanz auf der anderen Seite häufig zu einem zusätzlichen Implementierungsaufwand und/oder einer erhöhten Anwendungskomplexität. Um diesem Aspekt teilweise entgegenzuwirken, werden für einen Teil der Datenstrukturen charakteristische Zugriffsmethoden angeboten. So besteht unter anderem die Möglichkeit, Mengenoperationen, über mehrere (Sorted) Sets durchzuführen und somit entsprechende Schnitt-, Vereinigungs- oder Differenzmengen abzufragen. Darüber hinaus wurde zusätzlich ein Publish/Subscribe-Messaging-System integriert, welches bei Bedarf einen performanten Austausch von binary safe Strings zwischen den an der Datenbank angemeldeten Clients erlaubt. Durch diesen erweiterten Funktionsumfang kann zum Beispiel der oben erwähnte Anwendungsfall des Customer Support Chats elegant realisiert werden.

Architektur

Redis bedient sich einer ausgesprochen leichtgewichtigen Architektur und setzt neben einer Single-Thread-Verarbeitung auf eine In-Memory-Datenhaltung. Insbesondere der zweite Aspekt ermöglicht eine größtmögliche Performanz beim Datenzugriff aufgrund fehlender Zugriffe auf vergleichsweise langsame Sekundärspeicher, wie HDDs oder SSDs. Andererseits limitiert diese Designentscheidung die Größe der Datenbank auf den zur Verfügung stehenden Hauptspeicher.

Zur optimalen Ausnutzung des Hauptspeichers wurde bei der Entwicklung darauf geachtet, dass möglichst wenig Speicher durch das Datenbankmanagementsystem selbst benötigt wird. Entsprechend erfordert eine leere Redis-Instanz nur ca. 1 MB des Hauptspeichers.

Für den Fall, dass die erforderliche Datenmenge den verfügbaren Hauptspeicher übersteigt, bietet die Redis-Cluster-Funktionalität eine Möglichkeit zur Verteilung der Datensätze auf mehrere Server. Jeder Server hält dabei exklusiv einen Teil der Key-Value-Paare, wobei die Verteilung über sogenannte Hash Slots erfolgt. Insgesamt existieren standardmäßig 16.384 Hash Slots, wobei jeder Server für einen Teil der Hash Slots verantwortlich ist. Um zu ermitteln, welches Key-Value-Paar zu welchem Hash Slot (und somit zu welchem Server) gehört, wird mittels einer Hash-Funktion der jeweilige Key auf einen numerischen Wert abgebildet, und anschließend mittels der Moduloperation auf den zugrundeliegenden Bereich von Hash Slots reduziert (siehe Abbildung 2). In der Folge ermöglicht dieses Verfahren eine effiziente Lastverteilung und kompensiert somit die sequentielle Verarbeitung, welche der Single-Threaded-Architektur geschuldet ist.

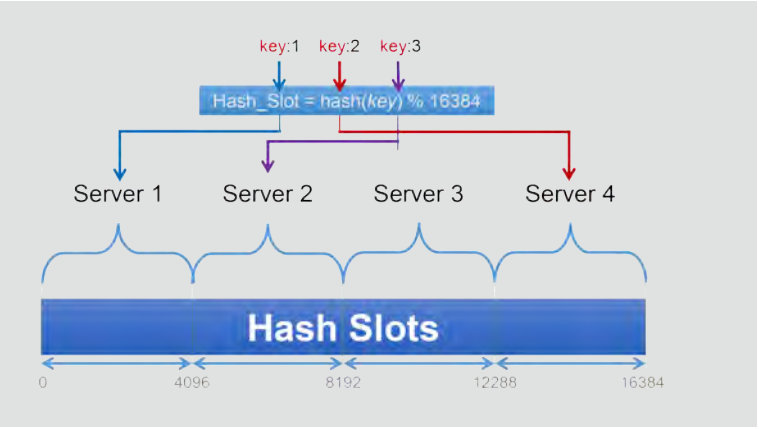


Abb. 2: Redis-Cluster-Datenverteilung

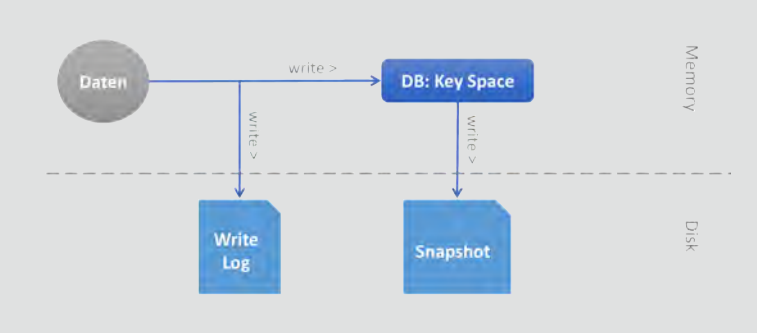


Abb. 3: Dauerhafte Datenspeicherung

Trotz der reinen In-Memory-Datenhaltung bietet Redis auch zwei Möglichkeiten zur dauerhaften Datenspeicherung (siehe Abbildung 3). Zum einen existiert die Möglichkeit, einen Point-in-Time-Snapshot bei Bedarf oder automatisiert in zyklischen Abständen erstellen zu lassen. Hierbei wird der laufende Redis-Prozess innerhalb des Hauptspeichers kopiert, sodass dieser Prozess im Hintergrund einen konsistenten Snapshot auf den Sekundärspeicher schreiben kann, während der ursprüngliche Server-Prozess weiterhin unbeeinträchtigt (für lesende und schreibende Datenzugriffe) zur Verfügung steht. Bedingt durch das zugrundeliegende Copy-On-Write-Verfahren, werden initial keine Daten beim Kopieren des Redis-Prozesses dupliziert. Dies erfolgt erst, sobald Daten durch eingehende Anfragen geändert werden. In der Konsequenz wird somit sichergestellt, dass der erforderliche Hauptspeicher lediglich auf die doppelte Größe anwachsen kann, sofern während der Snapshot-Erstellung sämtliche Daten im Original verändert werden.

Zusätzlich zu dem Snapshot-Mechanismus verfügt Redis auch über eine dauerhafte Protokollierung von schreibenden Operationen, wodurch sich die Daten im Falle eines Absturzes oder eines geplanten Neustarts, ebenfalls wiederherstellen lassen. Je nach Anforderung kann diese Protokollierung so konfiguriert werden, dass jede Schreiboperation unmittelbar dauerhaft gesichert wird oder die Speicherung zugunsten der Performanz zyklisch erfolgt. Beide Verfahren zur dauerhaften Datenspeicherung sind optional und können bei Bedarf einzeln oder in Kombination verwendet werden. In Anwendungsfällen, in welchen Redis lediglich als temporärer Cache verwendet wird, können beide Varianten auch zugunsten der Performanz vollständig deaktiviert werden.

Um die Verfügbarkeit eines Redis-Clusters zu erhöhen, können die Daten eines Servers zu einem separaten Server repliziert werden. Bedingt durch die zugrundeliegende Master-Slave-Architektur (siehe Abbildung 4) können schreibende Operationen jedoch nur über die Master Server erfolgen. Slave Server können im Fehlerfall den entsprechenden Master ersetzen und zusätzlich zur Skalierung lesender Abfragen verwendet werden. Bei einer größeren Anzahl von Slave Servern empfiehlt es sich, die Replikation kaskadierend zu organisieren (siehe Abbildung 5). Dies hat den Vorteil, dass ein Master Server nicht mehr zu all seinen Slave Servern die Daten direkt replizieren muss, wodurch dieser in der Folge weniger stark belastet wird. Es gilt jedoch zu beachten, dass die Replikation generell asynchron erfolgt und ein konsistenter Zugriff nur über den Master Server garantiert wird.

Security

Eine Schwachstelle im wörtlichen Sinne stellt das Fehlen grundlegender Sicherheitsfunktionalitäten dar. So verfügt Redis über keine sichere Benutzerauthentifizierung, Autorisierung, Verschlüsselung (weder für Data-at-Rest noch für Data-in-Motion), Auditierung oder Validierung von Benutzereingaben.

Vielmehr begnügt es sich mit einer einfachen Passwortabfrage ohne Benutzerkennung und einer möglichen Einschränkung des zulässigen Netzwerk-Interfaces. Letztlich setzt es eine vertrauenswürdige Umgebungslandschaft voraus und überträgt somit die Verantwortung für die Sicherheit an die nutzende Anwendung und an eine restriktive Firewall.

Fazit

Mit Redis existiert eine Datenbank, welche die Anforderungen nach Performanz und Skalierbarkeit konsequent bedient. Für Anwendungsfälle mit entsprechender Priorisierung und einfachen Datenmodellen kann Redis unter den NoSQL-Datenbanken eine geeignete Alternative sein. Voraussetzung hierfür ist jedoch, dass die fehlenden Sicherheitsfunktionalitäten durch eine entsprechend sichere Umgebung kompensiert werden oder schlichtweg nicht erforderlich sind.

Die einfache Architektur und die solide Dokumentation erlauben einen einfachen Einstieg und schnelle erste Erfolge. Herausfordernd wird dagegen im weiteren Projektverlauf die geeignete Datenmodellierung, welche einerseits den Performanz-Anforderungen gerecht wird und andererseits zusätzliche Komplexität und unnötige Redundanzen vermeidet.

Holger Wegert
(info@ordix.de)

Abb. 4: Master-Slave-Replikation

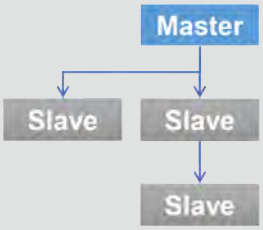


Abb. 5: Kaskadierende Replikation

Glossar

BSD Lizenz
Bei der „Berkeley Software Distribution Lizenz“ handelt es sich um eine Open Source Lizenz, welche es erlaubt, die darunter lizensierte Software frei zu kopieren, zu verändern und zu verbreiten, ohne dass das daraus resultierende Ergebnis selbst einer Open Source Lizenz unterliegen muss.

NoSQL
NoSQL steht für „Not only SQL“ und bezeichnet eine Menge nicht relationaler Datenbanken, welche standardmäßig auf alternative Zugriffsmechanismen zu SQL setzen.

SQL
Die „Standard Query Language“ ist eine Abfragesprache für relationale Datenbanken.

Links/Quellen

[1] Downloadseite Redis
https://github.com/antirez/redis

[Q1] Produktseite der Redis-Datenbank
https://redis.io/

Bildnachweise

Oracle Compression

Weniger ist mehr

In einigen Projekten bin ich in den letzten Monaten immer wieder von Kunden mit Fragen zum Thema Komprimierung angesprochen worden. Es herrscht offensichtlich eine große Unsicherheit, welche Komponenten zusätzlich lizenziert werden müssen und welche in der SE2 oder EE enthalten sind. Außerdem ist für viele Kunden interessant, ob die Verwendung der Komprimierung merkbare Einsparungen im Hinblick auf Performance und Plattenplatz mit sich bringt. In diesem Artikel geben wir eine Lizenzübersicht, stellen die Möglichkeiten kurz dar und berichten von Lasttests.

Lizenzsituation

Wer die SE2 lizenziert hat, kann Backups in Oracle 12c unter bestimmten Umständen komprimieren. Wer die EE hat, kann zusätzlich Tabellen beim Laden, sowie Indizes und IOTs komprimieren. Für alle anderen Methoden muss die Advanced-Compression-Option zusätzlich zur EE lizenziert werden (siehe Abbildung 1).

Überblick

Mit der Datenbankversion 9i (Release 2) hat Oracle die Komprimierung von Tabellen für **`Bulk Loads`** eingeführt. Hierdurch kann der Speicherbedarf für Datensätze bei einem **`DIRECT LOAD`** oder bei dem Befehl **`CREATE TABLE ... AS SELECT ...`** reduziert werden.

Diese Art der Datenkomprimierung eignet sich in hervorragender Weise für Data-Warehouse-Umgebungen, in denen der Großteil der Daten durch Batchjobs in die Datenbank gebracht wird.

Funktionalität	seit Version	SE2	EE	Compression Option	In-Memory Option
Basic Table Compression	9i		X		
Advanced Table Compression	11g		X	X	
Basic Index / IOT Compression	9i		X		
Advanced Index Compression	12c		X	X	
LOB Compression / Deduplication	11g		X	X	
rman Basic Compression	10g	X			
rman Advanced Compression	11g		X	X	
Data Pump Advanced Compression	11g		X	X	
Network Compression	11g/12c		X	X	
Information Lifecycle Management	12c		X	X	
In-Memory Compression	12c		X		X

Abb. 1: Lizenzübersicht

In der Datenbankversion 11g wurde das neue Feature **`OLTP Table Compression`** eingeführt. Die Möglichkeit, Daten zu komprimieren wurde hierbei auf alle DML-Operationen, wie **`INSERT`**, **`UPDATE`** und **`DELETE`**, ausgeweitet. Hierbei handelt es sich nicht um eine schlichte Erweiterung, sondern der verwendete Algorithmus ist komplett überarbeitet worden, um den Overhead bei Schreiboperationen zu reduzieren. Dadurch hat sich der Einsatzbereich auf die OLTP-Tabellen erweitert. IOTs können seit Oracle 9i, Indizes bereits seit Oracle 8i komprimiert werden.

Basic Table Compression – Compression For Direct Load Operations

Mit der Oracle 9iR2 wurde die „Compression for Direct Load Operations“ eingeführt. Die Datenkompression erfolgt auf Blockebene. Beim Einfügen in die Tabelle wird nach mehrfach auftretenden Datenwerten innerhalb eines Blocks gesucht. Für diese wird in jedem Block ein eigener Bereich reserviert. In diesem Bereich liegt die so genannte Symboltabelle.

Mehrfach auftretende Datenwerte werden ausschließlich in der Symboltabelle gespeichert. Im restlichen Platz innerhalb des Blocks werden die eigentlichen Datensätze gespeichert. Diese enthalten für die komprimierten Datenwerte Zeiger auf die Symboltabelle.

Die Platzersparnis ergibt sich aus der Verwendung der Zeiger, die weniger Speicherplatz als die Datenwerte erfordern. Um eine komprimierte Tabelle zu erstellen, wird beim Anlegen die Option **`COMPRESS`** angegeben.

Nachträgliche Änderungen der Daten einer komprimierten Tabelle führen zunächst zu einer Dekomprimierung. Nach der Änderung werden die Datensätze dann in unkomprimierter Form wieder eingefügt. Dadurch geht die angestrebte Platzersparnis also verloren.

Nur bei Verwendung einer der folgenden Blockoperationen (**BULK LOAD** oder **BULK INSERT**) erfolgt eine Komprimierung:

- **SQL*Loader** mit der Option "**direct = true**"
- **CREATE TABLE . . . AS SELECT . . . "**
- **INSERT /*+ APPEND */ ...**
  INSERT /*+ PARALLEL(tab_name, n) */ ...

Die Komprimierungsrate ist stark von der Weise abhängig, in der die Daten geladen werden. Es sollte dafür gesorgt werden, dass die Daten in sortierter Reihenfolge in die Tabelle gelangen, um die Blöcke direkt optimal zu befüllen.

Table Compression – Compression For All Operations

Die Kompressionsmethode steht ab Oracle 11g zur Verfügung (bei Lizenzierung der Advanced-Compression-Option).

Die grundlegende Methode der Komprimierung wurde hierbei nicht verändert. Die Einsparungsquote ist somit auch hier stark von der Gleichartigkeit der Datensätze innerhalb eines Datenblocks abhängig.

Um den durch die Komprimierung der Daten entstehenden Overhead zu reduzieren, werden die Datensätze nicht einzeln, sondern blockweise in einer Art Batch-Job komprimiert.

Ein leerer Datenbankblock wird zunächst mit unkomprimierten Daten gefüllt, bis die Freispeichergrenze PCTFREE erreicht wird. Wird diese Grenze überschritten, wird der gesamte Inhalt des Datenbankblocks nach der oben genannten Kompressionsmethode komprimiert. Aus einem unkomprimierten Block wird in diesem Schritt ein komprimierter Block. Dieser Datenbankblock steht weiterhin für die Aufnahme von Datensätzen durch **INSERT**-Statements bereit. Durch die weitere Aufnahme von Datensätzen wird aus dem unkomprimierten Block ein teilweise komprimierter Block. Durch erneutes Erreichen der Freispeichergrenze wird der teilweise komprimierte Datenbankblock nochmals komprimiert. Es entsteht wiederum ein vollständig komprimierter Datenblock. Diese Prozedur wird fortgesetzt, bis der Block gefüllt ist.

Aufgrund dieser Vorgehensweise lösen nur wenige SQL-Statements die eigentliche Komprimierung der Daten aus, wodurch die Performance nur minimal beeinträchtigt wird.

Index Compression

Doppelte Indexwerte werden bei einem komprimierten Index nur einmalig gespeichert. Dadurch kann der Index-Baum kleiner werden, die Anzahl der Leaf-Blöcke kann minimiert werden. Natürlich ist das Ziel der Komprimierung auch eine Minimierung der Plattenzugriffe. Eine Index-Komprimierung kann sehr sinnvoll sein, wenn die indizierte Spalte wenig unterschiedliche Werte enthält oder wenn der Index aus mehreren Spalten zusammengesetzt ist.

Aktion	Details	unkomprimiert	komprimiert
Größe der Tabelle		2910 MB	749 MB
Lesen mit count(*)	Laufzeit	67 sec	30 sec
	Gelesene Blöcke	3719545	952518
Massen Update (200.000 rows)	Laufzeit	13:11 min	3:05 min
	Gelesene Blöcke	7382951	952543
Massen Insert (1 Mio rows)	Laufzeit	2:28 min	5:16 min
Massen Delete (200.000 rows)	Laufzeit	4:48 min	2:09 min
	Gelesene Blöcke	75875428	57096136

Abb. 2: Lasttest – Tabelle ohne LOB

Indexname	Größe aktuell	Größe nach Reorganisation		Gewinn
		unkomprimiert	komprimiert	In Prozent
I1	3342 MB	3392 MB	2318 MB	31,66 %
I2	608 MB	609 MB	447 MB	26,60 %
I3	392 MB	392 MB	96 MB	75,51 %
I4	392 MB	385 MB	264 MB	32,65 %
I5	296 MB	298 MB	96 MB	67,79 %
I6	192 MB	192 MB	136 MB	29,17 %
I7	320 MB	184 MB	54 MB	70,65 %
I8	262 MB	252 MB	94 MB	62,70 %
I9	205 MB	197 MB	141 MB	28,43 %
I10	183 MB	175 MB	121 MB	30,86 %
I11	152 MB	152 MB	96 MB	36,84 %
I12	129 MB	128 MB	96 MB	25,00 %
I13	120 MB	120 MB	96 MB	20,00 %

Abb. 3: Lasttest – Indizes mit Basic Compression

Mit Oracle 12c ist eine weitere Methode hinzugekommen: Die Advanced Index Compression. Dieses Verfahren darf nur bei Lizenzierung der Advanced-Compression-Option genutzt werden und ist für sehr ungleichmäßig verteilte Indizes gedacht. Die Erfahrungen aus der Praxis in 12.1 sind bisher nicht vielversprechend.

LOB Compression

Mit Oracle 12c sind die SecureFiles die Standardimplementierung der LOBs geworden. SecureFiles können komprimiert, dedupliziert und verschlüsselt werden.

Die Komprimierung basiert auf klassischen zip-Verfahren, bei der Deduplizierung werden identische LOBs lediglich einmal gespeichert. Sowohl die Komprimierung als auch die Deduplizierung fordern die Lizenzierung der Advanced-Compression-Option.

rman Compression

Hier bietet Oracle vier verschiedene Komprimierungsverfahren an, die Basic-Methode ist schon seit Oracle 8i verfügbar und nach wie vor kostenfrei. Alle anderen Verfahren sind mit Oracle 11g eingeführt worden und erfordern die Advanced-Compression-Option.

Zusätzlich werden seit Oracle 8i alle niemals genutzten Blöcke bei Nutzung der Enterprise Edition komprimiert. Die Laufzeiten und Komprimierraten sind sehr unterschiedlich.

Data Pump Compression

Die verwendeten Verfahren sind identisch zur rman Compression. Allerdings sind beim DataPump alle Komprimierungsmethoden für Tabellendaten kostenpflichtig, während die Basisvariante zur Komprimierung der Metadaten kostenfrei ist.

Tabelle	Info / Aktion	LOB Segment					
				Compress			
		basic	deduplicate	low	medium	high	high deduplicate
T1	Größe Tabelle (in MB)	0,92	0,92	0,92	0,92	1,91	1,91
	Größe LOB (in MB)	4027	3984	2814	2688	2624	2613
	LOB Lesen (in sec)	20,60	16,89	17,71	17,99	16,32	16,24

Aktion	Details	unkomprimiert	komprimiert
Größe des LOB Segments	70.000 LOBs	57,00 GB	11,54 GB
Masseninsert	Laufzeit	6172 sec	8948 sec
Full Table Scan (mit LOBs)	Laufzeit	235 sec	195 sec

Abb. 4: Lasttest – Tabelle mit LOB

Glossar

EE
Die Oracle Enterprise Edition bietet sowohl in Cluster- als auch in Einzelsystemen branchenweit führende Skalierbarkeit und Zuverlässigkeit. Sie verfügt über umfassendste Funktionen für OLTP und Business Intelligence und bietet gleichzeitig die niedrigste TCO.

IOT
Index Organized Table – Eine Index-organisierte Tabelle ist ein B-Tree-Index ohne Heap-Tabelle. Hierdurch wird der Speicherplatz für die Heap-Struktur gespart.

LOB
Unter dem Oberbegriff LOB (Large Object) sind in SQL und bei Oracle PL/SQL diverse Datentypen für die Speicherung großer unstrukturierter Objekte zusammengefasst.

SE2
Oracle Standard Edition – Aus der Perspektive der Software handelt es sich bei der Oracle Standard Edition 2 (SE2) um die gleichen Binaries wie bei den Vorgänger-Editionen, jedoch ist sie aus lizenzrechtlicher Sicht ein eigenständiges, neues Produkt mit neuen Limitationen und bewährten Funktionalitäten.

Erfahrungsbericht/Lasttest

In den letzten 24 Monaten haben wir bei einigen Kunden Komprimierungen in Oracle-Datenbanken implementiert und dabei auch diverse Lasttests gemacht. Einen Auszug dieser Ergebnisse stellen wir hier dar.

Der erste Lasttest wurde auf einer Tabelle ohne LOBs durchgeführt, die mit der Advanced Compression komprimiert worden ist (siehe Abbildung 2).

Inserts wurden etwa um den Faktor 2 langsamer, Löschungen doppelt so schnell, Änderungen wurden fundamental schneller. Die Ursache dieser Verbesserungen liegt vor allem in der enormen Minimierung der Plattenzugriffe.

Der nächste Test hat sich mit der Komprimierung von Indizes befasst. Hier haben wir vor allem eine mögliche Minimierung des Plattenplatzes und damit auch eine Minimierung der Plattenzugriffe betrachtet (siehe Abbildung 3). Die Indizes wurden um bis zu 75% verkleinert. Als Konsequenz davon wurde auch die Trefferquote im Buffer Cache besser.

Der letzte Test wurde dann auf Tabellen mit LOBs durchgeführt. In diesem haben wir fünf Tabellen unter den verschiedenen Möglichkeiten getestet. Die LOBs wurden nacheinander mit den unterschiedlichen Verfahren komprimiert. Dabei hat sich gezeigt, dass man kein allgemeingültiges Komprimierverfahren empfehlen kann, die Ergebnisse sind je LOB unterschiedlich (siehe Abbildung 4). Exemplarisch zeigen wir hier die Ergebnisse für eine Tabelle (hier genannt T1). Der eigentliche Lasttest hat dann gezeigt, dass Massen-Inserts langsamer wurden, während die lesenden Zugriffe schneller wurden.

Fazit

Komprimierungen können zu sehr positiven Laufzeitverbesserungen führen, ändernde Zugriffe werden unter Umständen schlechter. Wir empfehlen jedem Kunden, der sich mit dieser Thematik auseinandersetzen möchte, einen PoC aufzusetzen, in dem die erwartete Verbesserung getestet und verifiziert wird. Wir möchten nochmals betonen, dass die kostenpflichtige Option nicht grundsätzlich für jede Variante notwendig ist. Wenn Sie Hilfe bei der Anwendung der Oracle Compression benötigen, helfen Ihnen unsere Experten unter 0 52 51 / 10 63 -0 gerne weiter.

Klaus Reimers
(info@ordix.de)

Oracle Network Encryption:

Sichere Übertragung von Daten zwischen Client und Server

Datenbanken enthalten oft sensible und schützenswerte Daten. Anwendungen greifen in der Regel über das Netzwerk auf diese Informationen zu. Eine Standard-Datenbank-Installation überträgt solche Informationen unverschlüsselt zwischen Client und Server. Dieser Umstand bietet potenziellen Angreifern die Möglichkeit, den Datenverkehr zu belauschen. Deshalb ist die Absicherung des Netzwerkverkehrs genauso wichtig, wie die der Datenbank selbst. Oracle bietet hier zum einen die Netzwerkverschlüsselung über SQL*Net oder über SSL/TSL. Selbstverständlich kann die Netzwerkübertragung auch mit einem Fremdprodukt auf der Netzwerkebene verschlüsselt werden. In diesem Artikel werden wir eine Lizenzübersicht geben und die Möglichkeiten mit Oracle-Mitteln kurz darstellen.

Lizenzsituation

Oracle bietet die Netzwerkverschlüsselung über SQL*Net oder über SSL/TLS an. Beides war Teil der „Advanced Security Option“ und steht mit Erscheinen von Oracle 12cR1 im Juni 2013 in den kostenpflichtigen Datenbank-Editionen (Standard/Enterprise), aller aktuell unterstützten Releases als Datenbank-Feature, zur Verfügung.

Überblick

Mit beiden Verschlüsselungsmethoden ist es möglich den Netzwerkverkehr zu verschlüsseln. Sie unterscheiden sich hinsichtlich der Art ihrer Implementierung und den notwendigen Voraussetzungen.

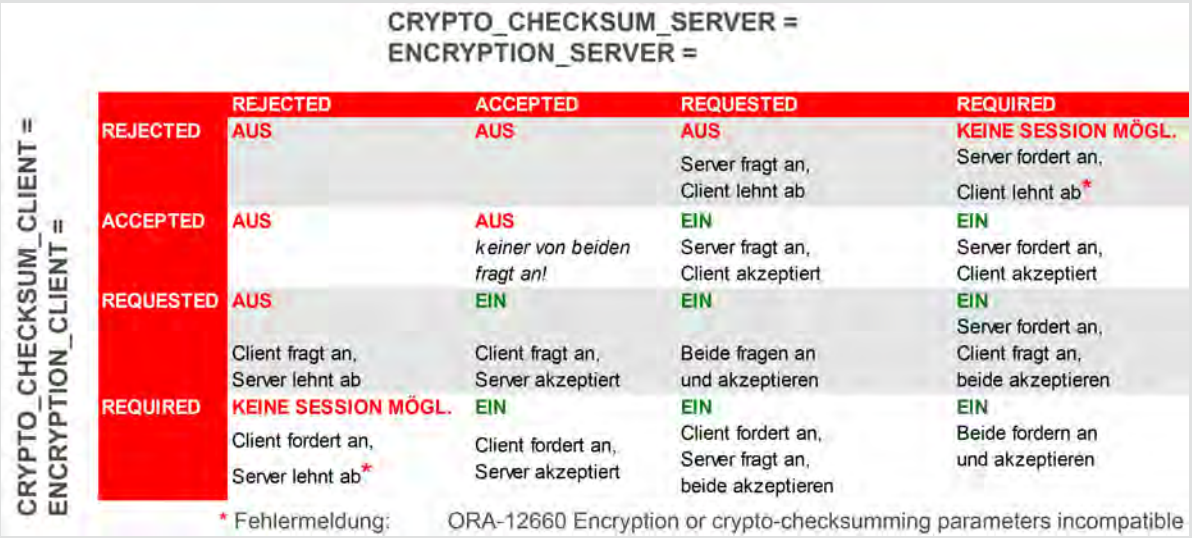
Oracle hat den Secure-Hash-Algorithmus SHA2 zur Sicherstellung der Datenintegrität mit Oracle 12c eingeführt. Einen Überblick über die wichtigsten Algorithmen sehen Sie in Abbildung 1.

Vorüberlegungen

Bei der Aktivierung der Oracle-Netzwerkverschlüsselung muss vorab noch zwischen der Übertragung von Daten nach dem Aufbau einer neuen User Session und der Datenübertragung einer eventuell noch bestehenden User Session unterschieden werden. Im ersten Fall werden die Daten verschlüsselt übertragen. Hier besteht kein Handlungsbedarf.

Oracle Client/Server	Verschlüsselungsalgorithmus	Integritätsalgorithmus
Oracle 10gR2	AES256*,AES192*,3DES168,RC4_256	SHA1*, MD5 * Nicht JDBC Thin Client
Oracle 11gR1	AES256,AES192,3DES168,RC4_256	SHA1, MD5
Oracle 11gR2	AES256,AES192,3DES168,RC4_256	SHA1, MD5
Oracle 12cR1	AES256,AES192,3DES168,RC4_256	SHA512, SHA384, SHA256, SHA1, MD5

Abb. 1: Lizenzübersicht (Secure-Hash-Algorithmus SHA2 mit Version 12c eingeführt)



CRYPTO_CHECKSUM_SERVER =
ENCRYPTION_SERVER =

CRYPTO_CHECKSUM_CLIENT = ENCRYPTION_CLIENT =	REJECTED	ACCEPTED	REQUESTED	REQUIRED
REJECTED	AUS	AUS	AUS Server fragt an, Client lehnt ab	KEINE SESSION MÖGL. Server fordert an, Client lehnt ab*
ACCEPTED	AUS	AUS *keiner von beiden fragt an!*	EIN Server fragt an, Client akzeptiert	EIN Server fordert an, Client akzeptiert
REQUESTED	AUS Client fragt an, Server lehnt ab	EIN Client fragt an, Server akzeptiert	EIN Beide fragen an und akzeptieren	EIN Server fordert an, Client fragt an, beide akzeptieren
REQUIRED	KEINE SESSION MÖGL. Client fordert an, Server lehnt ab*	EIN Client fordert an, Server akzeptiert	EIN Client fordert an, Server fragt an, beide akzeptieren	EIN Beide fordern an und akzeptieren

* Fehlermeldung: ORA-12660 Encryption or crypto-checksumming parameters incompatible

Abb. 2: Verschlüsselungsmatrix (SQL*Net)

Anders ist es bei der Übertragung der Daten einer bestehenden User Session. Diese Daten werden nicht automatisch verschlüsselt und die Verbindung bleibt unverschlüsselt und damit für einen potenziellen Angreifer im Klartext lesbar. Alle verbundenen User Sessions müssen bei der Aktivierung der Netzwerkverschlüsselung getrennt und neu verbunden werden. Nur so wird eine Absicherung gegen das Mitlesen des Netzwerkverkehrs erreicht.

Ziel

Ziel der Netzwerkverschlüsselung ist es, basierend auf einem Schlüssel, den Klartext in unlesbaren Text umzuwandeln und es damit einem Angreifer zu erschweren, den Datenverkehr mitzulesen. Um dies zu erreichen kann eine der beiden Verschlüsselungsmethoden über SQL*Net oder über SSL/TLS zum Einsatz kommen. Netzwerkverschlüsselung und zusätzliche Integritätsprüfung bieten:

- Schutz der Daten als auch der Datenintegrität bei der Übertragung zwischen Client und Server
- Schutz vor packet sniffing
- Schutz vor man-in-the-middle-Angriffen

Allerdings entsteht auch Aufwand bei Einrichtung, Konfiguration und Test der Verschlüsselung. Bei SSL/TLS-Verschlüsselung müssen zusätzlich Zertifikate erstellt werden und eine Public Key Infrastructure (PKI) zur Verfügung stehen. Eventuell müssen auch Programme und Verbindungsparameter angepasst werden.

Verschlüsselung über SQL*Net

Diese Verschlüsselungsmethode wird auch als native-Oracle-Verschlüsselung bezeichnet. Sie benötigt keine Public Key Infrastructure (PKI) und verwendet zur Erzeugung sicherer Schlüssel den Diffie-Hellman-Algorithmus. Auf beiden Seiten wird ein temporärer Schlüssel erzeugt. Dieser wird benutzt, um daraus einen symmetrischen Session-Schlüssel zu erzeugen (Einwegfunktion), mit dem die eigentliche Kommunikation verschlüsselt wird. Hierbei macht man sich die Tatsache zunutze, dass es zwar sehr einfach ist, eine Zahl zu potenzieren, dass es jedoch nur mit sehr großem Aufwand möglich ist, den diskreten Logarithmus einer Zahl zu berechnen.

Datenintegrität

Durch Verschlüsselung kann die Integrität der Daten nicht gewährleistet werden, daher ist es ratsam, diese durch Checksummen sicherzustellen. Sämtliche Modifikationen am Datenverkehr können somit aufgedeckt werden und führen zur Beendigung der Session.

Dabei kann zwischen drei Verfahren gewählt werden: SHA2 (ab Oracle 12c), SHA1 und MD5. Da mittlerweile Schwächen in MD5 aufgedeckt wurden, sollte dieser Algorithmus nicht mehr verwendet werden. SHA2 ist hierbei der modernste Algorithmus und die bessere Alternative, sofern Client und Server diesen Algorithmus unterstützen. Dazu müssen beide Seiten Oracle 12c verwenden.

Aktivierung

Für die Aktivierung sind in der sqlnet.ora des Servers und ggf. des Clients zusätzliche Parameter zu setzen.

Die Parameter **`SQLNET.ENCRYPTION_SERVER`** und **`SQLNET.CRYPTO_CHECKSUM_SERVER`** können dabei jeweils vier verschiedene Werte annehmen. Je nachdem,

ob der Server eine Verschlüsselung/Integritätsprüfung verlangt (**REQUIRED**), anfragt (**REQUESTED**), akzeptiert (**ACCEPTED**) oder zurückweist (**REJECTED**). Da die Encryption-Parameter **SQLNET.ENCRYPTION_CLIENT** und **SQLNET.CRYPTO_CHECKSUM_CLIENT** ebenfalls auf dem Client gesetzt werden können, ergeben sich 4*4=16 Möglichkeiten. Es ist empfehlenswert, Verschlüsselung und Integritätsprüfung auf dem Server jeweils auf **REQUIRED** zu setzen, um so eine verschlüsselte Verbindung zu erzwingen, ohne die eine Session nicht ausgehandelt werden kann. Die Matrix beschreibt dies für jeden der 16 möglichen Fälle (siehe Abbildung 2).

Die Art der Verschlüsselung kann mittels der Parameter **SQLNET.ENCRYPTION_TYPES_SERVER** und **SQLNET.ENCRYPTION_TYPES_CLIENT** eingestellt werden. Hier gilt: Je höher die Bitrate, desto sicherer ist die Verschlüsselung. Allerdings steigt natürlich mit einer höheren Bitrate auch der Aufwand für die CPU. AES256 und 3DES168 haben sich in der Praxis bewährt und sind zu empfehlen. Wobei letzterer Algorithmus 3DES168 weniger performant als AES256 ist und auch deutlich CPU-intensiver. Stromverschlüsselung RC4 ist ein sehr performanter Algorithmus, und es wird davon abgeraten. Im Zweifelsfall gilt, besser schlecht verschlüsselt als überhaupt nicht verschlüsselt.

Den Parameter **SQLNET.CRYPTO_SEED** sollten Kunden, die native-Oracle-Verschlüsselung einsetzen, auf jeden Fall verwenden und einen zufälligen Wert mit maximaler Länge konfigurieren, um eine höhere Absicherung zu erreichen.

Verschlüsselung über SSL/TSL

Um bei der Nutzung von asymmetrischen Kryptosystemen den Einsatz falscher (z.B. untergeschobener) Schlüssel zu verhindern, wird eine Garantie benötigt, dass der verwendete, öffentliche Schlüssel auch zum designierten Empfänger der verschlüsselten Nachricht bzw. zum Sender einer elektronisch signierten Nachricht gehört. Diese Garantie stellt eine vertrauenswürdige Stelle in Form eines digitalen Zertifikates aus. Oracle unterstützt die sogenannte Public Key Infrastructure (PKI) und bietet die Möglichkeit, die erstellten Zertifikate bequem im Oracle-Internet-Directory oder typischerweise im Dateisystem (Wallet) abzulegen.

Für den Einsatz von SSL-Verbindungen über Oracle Net sind die folgenden Konfigurationsschritte durchzuführen:

- Anlegen eines Wallets
- Erstellung eines Zertifikats-Requests
- Export eines Zertifikats-Requests in eine Datei
- Zertifizierungsanforderung unterschreiben/ beglaubigen lassen
- Zertifikatskette und Zertifikat importieren
- Konfiguration eines zusätzlichen SSL Listeners

Auf jedem Client muss ebenfalls mindestens ein Wallet/ Zertifikats-Store angelegt und dort die Zertifikatskette importiert werden. Gegebenenfalls sollten auch Client-Zertifikate verwendet werden – in diesem Fall müssen für jeden Client ebenfalls Zertifikats-Requests erstellt und das Zertifikat importiert werden.

Die Absicherung von Oracle-Verbindungen mittels SSL ermöglicht das Erreichen einer hohen Sicherheitsstufe. Nachteilig anzusehen ist der hohe administrative Aufwand und die dafür benötigten Kenntnisse im Bereich der Public Key Infrastructure.

Fazit

Oracle bietet mit Netzwerkverschlüsselung über SQL*Net oder SSL/TLS zwei Varianten der Netzwerkverschlüsselung in allen aktuell unterstützten Releases an. Netzwerkverschlüsselung nimmt in immer mehr Sicherheitskonzepten eine zentrale Rolle ein. Auch in Ihrem? Bei der Analyse Ihres Systems und der Implementierung von Sicherheitsrichtlinien unterstützen wir Sie gerne. Wir empfehlen jedem Kunden, der sich mit dieser Thematik auseinandersetzen möchte, einen PoC aufzusetzen, in dem der erwartete Overhead gemessen und verifiziert wird.

Glossar

EE
Die Oracle Enterprise Edition bietet sowohl in Cluster- als auch in Einzelsystemen branchenweit führende Skalierbarkeit und Zuverlässigkeit. Sie verfügt über umfassendste Funktionen für OLTP und Business Intelligence und bietet gleichzeitig die niedrigste TCO.

PKI
Mit Public Key Infrastructure (PKI) bezeichnet man in der Kryptologie ein System, das digitale Zertifikate ausstellen, verteilen und prüfen kann. Die innerhalb einer PKI ausgestellten Zertifikate werden zur Absicherung rechnergestützter Kommunikation verwendet.

PoC
Im Projektmanagement ist ein Proof of Concept (PoC) ein Meilenstein, an dem die prinzipielle Durchführbarkeit eines Vorhabens belegt ist.

SE2
Oracle Standard Edition - Aus der Perspektive der Software handelt es sich bei der Oracle Standard Edition 2 (SE2) um die gleichen Binaries wie bei den Vorgänger-Editionen, jedoch ist sie aus lizenzrechtlicher Sicht ein eigenständiges, neues Produkt mit neuen Limitationen und bewährten Funktionalitäten.

SSL/TLS
Transport Layer Security (TLS), weitläufiger bekannt unter der Vorgängerbezeichnung Secure Sockets Layer (SSL), ist ein hybrides Verschlüsselungsprotokoll zur sicheren Datenübertragung.

Thilo Fleischhauer (info@ordix.de)

ORDIX® seminare

Seminarzentrum:
ORDIX AG
Kreuzberger Ring 13
65205 Wiesbaden
Tel.: 0611 77840-00

Online-Anmeldung, aktuelle Seminarinhalte und Termine unter:
https://seminare.ordix.de

Unser Seminarstandort ist Wiesbaden.

Die hier angegebenen Termine entsprechen dem ersten Seminartag.

Die Preise gelten pro Seminar pro Teilnehmer in Euro zzgl. ges. MwSt., Inhouse-Preise auf Anfrage.

Big Data und Data Warehouse

BIG Data				
DB-BIG-01	Big Data: Informationen neu gelebt	1 Tag	590,00 €	26.06. | 11.09. | 20.11.
DB-BIG-02	Big Data: Apache Hadoop Grundlagen	3 Tage	1.290,00 €	03.07. | 18.09. | 04.12.
Data Warehouse				
DB-DB-03	Data Warehouse Grundlagen	3 Tage	1.290,00 €	27.06. | 12.09. | 21.11.
DB-NSQL-01	Einführung in NoSQL-Datenbanken	2 Tage	1.090,00 €	06.07. | 21.09. | 07.12.

PostgreSQL

DB-PG-01	PostgreSQL Administration	5 Tag	2.150,00 €	28.08. | 20.11.

Oracle

Entwicklung				
DB-ORA-01	Oracle SQL	5 Tage	1.890,00 €	26.06. | 11.09. | 06.11.
DB-ORA-01A	Oracle SQL Power Workshop	3 Tage	1.290,00 €	21.08. | 27.11.
DB-ORA-02	Oracle Datenbankprogrammierung mit PL/SQL Grundlagen	5 Tage	1.890,00 €	10.07. | 25.09. | 13.11.
DB-ORA-34	Oracle Datenbankprogrammierung mit PL/SQL Aufbau	3 Tage	1.290,00 €	24.07. | 04.10.
DB-ORA-42	Oracle PL/SQL für Experten - Performance Analyse & Laufzeitopt.	3 Tage	1.290,00 €	06.06. | 09.10.
DB-ORA-53	Oracle Text	3 Tage	1.390,00 €	25.09. | 04.12.
DB-ORA-51	Oracle Spatial	3 Tage	1.290,00 €	16.10. | 04.12.
DB-ORA-46	Oracle APEX Anwendungsentwicklung Grundlagen	3 Tage	1.290,00 €	17.07. | 11.09. | 06.11.
DB-ORA-47	Oracle APEX Anwendungsentwicklung Aufbau	3 Tage	1.290,00 €	07.06. | 25.09. | 27.11.
Administration				
DB-ORA-03	Oracle Datenbankadministration Grundlagen	5 Tage	1.990,00 €	19.06. | 04.09. | 06.11.
DB-ORA-04	Oracle Datenbankadministration Aufbau	5 Tage	1.990,00 €	17.07. | 23.10.
DB-ORA-07	Oracle Tuning - Theorie und Interpretation von Reports	5 Tage	2.290,00 €	26.06. | 11.09. | 20.11.
DB-ORA-11	Oracle Troubleshooting Workshop	3 Tage	1.390,00 €	Termine auf Anfrage
DB-ORA-08	Oracle 12c Real Application Cluster (RAC) und Grid Infrastructure	5 Tage	2.290,00 €	31.07. | 09.10. | 11.12.
DB-ORA-49	Oracle 12c Neuheiten	5 Tage	2.090,00 €	03.07. | 25.09. | 04.12.
DB-ORA-49E	Oracle 12c Neuheiten für Entwickler	3 Tage	1.390,00 €	25.09. | 13.11.
DB-ORA-52W	Oracle Lizenz Workshop Webinar	1 Tage	590,00 €	Termine auf Anfrage
DB-ORA-33	Oracle Security	3 Tage	1.290,00 €	14.08. | 16.10.
DB-ORA-35	Oracle Cloud Control	3 Tage	1.290,00 €	24.07. | 04.10. | 04.12.
DB-ORA-48	Oracle Golden Gate	3 Tage	1.290,00 €	31.07. | 23.10.
Backup und Recovery				
DB-ORA-32	Oracle Backup und Recovery mit RMAN	5 Tage	1.990,00 €	07.08. | 27.11.
DB-ORA-31	Oracle Data Guard	4 Tage	1.690,00 €	28.08. | 27.11.
MySQL				
DB-MY-01	MySQL Administration	3 Tage	1.290,00 €	17.07. | 25.09. | 27.11.

IBM Datenbanksysteme

Informix				
DB-INF-01	IBM Informix SQL	5 Tage	1.790,00 €	10.07. | 23.10.
DB-INF-02	IBM Informix Administration	5 Tage	1.990,00 €	24.07. | 13.11.
DB2				
DB-DB2-01	IBM DB2 für Linux/Unix/Windows SQL Grundlagen	5 Tage	1.890,00 €	31.07. | 16.10.
DB-DB2-02	IBM DB2 für Linux/Unix/Windows Administration	5 Tage	1.990,00 €	19.06. | 28.08. | 27.11.
DB-DB2-05	IBM DB2 für Linux/Unix/Windows Monitoring und Tuning	3 Tage	1.290,00 €	31.07. | 20.11.
DB-DB2-06	IBM DB2 für Linux/Unix/Windows Backup und Hochverfügbarkeit mit HADR	3 Tage	1.390,00 €	03.07. | 04.10.

Microsoft

Entwicklung				
MS-SQL-01	Querying Data with Transact-SQL	5 Tage	1.990,00 €	10.07. | 18.09. | 06.11.
MS-SQL-07	Updating Your Skills to Microsoft SQL Server 2016	5 Tage	1.990,00 €	19.06. | 21.08. | 23.10.
Administration				
MS-SQL-02	Administering a SQL Database Infrastructure	5 Tage	1.990,00 €	17.07. | 09.10. | 27.11.
MS-SQL-05	Implementing a SQL Data Warehouse	5 Tage	1.990,00 €	28.08. | 13.11.
MS-SQL-11	Microsoft SQL Server for Oracle DBAs	4 Tage	1.790,00 €	14.08. | 16.10.
MS-SQL-05W	Microsoft SQL Server 2016 Upgrade Webinar	1 Tag	99,00 €	Termine auf Anfrage

Storage

STORAGE01	Storage Grundlage	2 Tage	1.190,00 €	24.08. | 26.10.

Rechenzentrum

ANSIB-01	Konfigurationsmanagement mit Ansible	3 Tage	1.350,00 €	12.06. | 25.09. | 11.12.
E-Dock-01	Docker DevOps Workshop	1 Tag	405,00 €	10.08. | 21.09. | 26.10. | 14.12.
SM-NAG-01	Systemüberwachung mit Nagios - Workshop	3 Tage	1.190,00 €	28.08. | 23.10.

Web und Application-Server

INT-04	Apache HTTP Server Administration	3 Tage	1.190,00 €	07.08. | 04.10. | 11.12.
INT-07	Tomcat Konfiguration und Administration	3 Tage	1.290,00 €	24.07. | 18.09. | 20.11.
INT-08	WebSphere Application Server Installation und Administration	3 Tage	1.390,00 €	04.09. | 04.12.
INT-12	WildFly Application Server Administration	3 Tage	1.290,00 €	10.07. | 11.09. | 13.11.
INT-11-7	JBoss 7 Administration und Konfiguration	3 Tage	1.290,00 €	03.07. | 25.09. | 27.11.
DB-ORA-50	Oracle WebLogic Administration Grundlagen	3 Tage	1.390,00 €	17.07. | 16.10.

Projekt- und IT-Management

Klassisches Projektmanagement				
PM-01	IT-Projektmanagement - Methoden und Techniken	3 Tage	1.690,00 €	03.07. | 20.09. | 22.11.
PRINCE-01	PRINCE2® Foundation	3 Tage	1.225,00 €	26.06. | 21.08. | 16.10. | 11.12.
PRINCE-02	PRINCE2® Practitioner	3 Tage	1.560,00 €	28.06. | 23.08. | 18.10. | 13.12.
PRINCE-03	PRINCE2® kompakt	5 Tage	2.595,00 €	26.06. | 21.08. | 16.10. | 11.12.
PM-06	Projekte souverän führen - Systemisches Projektmanagement	4 Tage	1.850,00 €	04.09. | 06.11.
PM-05	Projektcontrolling in der IT	2 Tage	1.090,00 €	22.06. | 18.09. | 30.11.
PM-07	Krisenmanagement in Projekten - Projektkrisen meistern	2 Tage	1.100,00 €	10.08. | 26.10.
PM-14	Anforderungsmanagement in IT-Projekten	2 Tage	1.090,00 €	08.06. | 28.09. | 16.11.
PM-T-01	Testmanagement Grundlagen	2 Tage	1.190,00 €	06.06. | 17.08. | 26.10.
Agiles Projektmanagement				
PM-08	Agiles Projektmanagement mit Scrum	2 Tage	1.190,00 €	26.06. | 28.08. | 20.11.
PM-08-Z	Scrum Praxis und Zertifizierung	1 Tage	690,00 €	28.06. | 30.08. | 22.11.
PM-15	Leading Excellence for Scrum Master	3 Tage	1.340,00 €	06.06. | 01.11.
PM-24	KANBAN in der IT	2 Tage	1.190,00 €	29.06. | 31.08. | 23.11.
IT-Management, IT-Strategie und IT-Organisation				
IT-SEC-01	IT-Sicherheit für Projektmanager und IT-Leiter	5 Tage	2.700,00 €	21.08. | 23.10.
PM-29	Systemische Führung	3 Tage	1.650,00 €	13.11.
MGM-07	IT-Strategie - strategische IT-Planungen	3 Tage	1.650,00 €	07.08. | 23.10.
MGM-02	IT-Architekturen	3 Tage	1.590,00 €	02.08. | 30.10.
PM-10	IT-Controlling	3 Tage	1.590,00 €	14.08. | 11.12.
MGM-04	Geschäftsprozessmanagement (BPM)	3 Tage	1.590,00 €	19.06. | 25.09. | 04.12.
ITIL-01	ITIL® V3 Foundation	3 Tage	922,50 €	19.06. | 07.08. | 09.10. | 04.12.
ITIL-02	ITIL® V3 Practitioner	3 Tage	1.380,00 €	22.06. | 10.08. | 12.10. | 07.12.
ITIL-03	ITIL® V3 kompakt	3 Tage	2.250,00 €	19.06. | 07.08. | 09.10. | 04.12.
PM-28	IT-Organisation	3 Tage	1.650,00 €	10.07. | 16.10.
Kommunikation und Selbstmanagement				
PM-17	IT-Management Soft Skills	3 Tage	1.650,00 €	09.10.
PM-27	Social Skills für IT-Consultants	3 Tage	1.650,00 €	04.10.
PM-16	Kommunikation in Projekten - Power Workshop	2 Tage	1.090,00 €	07.09. | 16.11.
PM-11	Konfliktmanagement	2 Tage	1.100,00 €	13.07. | 19.10.
PM-19	Zeit- und Selbstmanagement	2 Tage	1.090,00 €	12.10.
PM-30	Verhandlungstechniken	2 Tage	1.100,00 €	07.12.

Betriebssysteme & Monitoring

Unix/Linux				
BS-01	Unix/Linux Grundlagen für Einsteiger	5 Tage	1.690,00 €	19.06. | 18.09. | 13.11.
BS-02	Linux Systemadministration	5 Tage	1.690,00 €	03.07. | 09.10. | 27.11.
BS-25	Unix Power Workshop für den Datenbank- & Applikationsbetrieb	5 Tage	1.890,00 €	28.08. | 13.11.
BS-27	Neuerungen SUSE Linux Enterprise Server 12	3 Tage	1.290,00 €	14.08. | 23.10.
BS-26	Einführung in die Administration von OpenStack	5 Tage	2.990,00 €	04.09. | 06.11.
BS-09	Linux Hochverfügbarkeits-Cluster	5 Tage	1.890,00 €	21.08. | 06.11.
Solaris				
BS-03-11	Solaris 11 Systemadministration Grundlagen	5 Tage	1.990,00 €	26.06. | 25.09. | 20.11.
BS-04-11	Solaris 11 Systemadministration Aufbau	5 Tage	1.990,00 €	17.07. | 11.12.
BS-06-11	Solaris 11 für erfahrene Unix/Linux-Umsteiger	5 Tage	1.990,00 €	24.07. | 11.09. | 27.11.
BS-24	Solaris 11 Administration Neuheiten	3 Tage	1.290,00 €	19.06. | 04.09. | 09.10.
BS-18	Solaris Virtualisierung mit ZFS und Container (Zonen)	5 Tage	1.990,00 €	07.08. | 04.12.
IBM AIX				
AIX-01	IBM AIX Systemadministration Grundlagen	5 Tage	1.990,00 €	26.06. | 11.09. | 06.11.
AIX-04	IBM AIX Systemadministration Power Workshop	3 Tage	1.290,00 €	10.07. | 04.10. | 11.12.
AIX-02	IBM AIX Installation, Backup und Recovery mit NIM	3 Tage	1.290,00 €	07.06. | 25.09. | 13.11.
Microsoft				
MS-W-10-01	Windows 10 für Administratoren	3 Tage	1.650,00 €	18.09. | 13.11.
MS-HV-03	Microsoft Hyper-V-Workshop unter Server 2016 (& frühere Versionen)	2 Tage	1.150,00 €	04.09. | 06.11.
MS-HV-01	Microsoft Hyper-V Deep-Dive unter Server 2016 (& frühere Versionen)	3 Tage	1.650,00 €	06.09. | 08.11.
Monitoring				
SM-NAG-01	Systemüberwachung mit Nagios - Workshop	3 Tage	1.190,00 €	28.08. | 23.10.

Entwicklung

Allgemeines				
OO-01	Einführung in die objektorientierte Programmierung und UML	3 Tage	1.190,00 €	03.07. | 18.09. | 20.11.
E-SWA-01	Softwarearchitekturen	5 Tage	1.890,00 €	24.07. | 13.11.
Script-Sprachen				
P-PERL-01	Perl Programmierung Grundlagen	5 Tage	1.690,00 €	19.06. | 18.09. | 20.11.
P-PERL-02	Perl Programmierung Aufbau	5 Tage	1.690,00 €	03.07. | 04.09. | 11.12.
P-UNIX-01	Shell, Awk und Sed	5 Tage	1.690,00 €	17.07. | 11.09. | 13.11.
XML				
P-XML-01	XML Grundlagen	3 Tage	1.190,00 €	07.08. | 06.11.
Java				
P-JAVA-01	Java Programmierung Grundlagen	5 Tage	1.690,00 €	17.07. | 09.10. | 04.12.
P-JAVA-03	Java Programmierung Aufbau	5 Tage	1.690,00 €	14.08. | 27.11.
P-JEE-08	Java Performance Tuning	3 Tage	1.290,00 €	10.07. | 11.09. | 06.11.
P-JAVA-11	Java 8 Neuheiten	2 Tage	990,00 €	03.08. | 26.10.
Java EE				
P-JAVA-12	Java EE Power Workshop	5 Tage	1.890,00 €	10.07. | 20.11.
J-HIB-01	Java Persistence API mit Hibernate	5 Tage	1.690,00 €	21.08. | 16.10.
INT-05	Java Web Services	3 Tage	1.190,00 €	18.09. | 27.11.
P-JEE-06	Spring Power Workshop	5 Tage	1.590,00 €	24.07. | 09.10.
Web- und GUI-Entwicklung				
P-JAVA-15	Einstieg in JavaFX	3 Tage	1.390,00 €	31.07. | 23.10.
P-PHP-01	PHP Programmierung	5 Tage	1.690,00 €	07.08. | 16.10.
P-JEE-05	Rich Internet Application mit JSF und Primefaces	5 Tage	1.590,00 €	26.06. | 28.08. | 13.11.
P-JEE-05A	Webanwendungen mit JavaServer Faces (JSF)	5 Tage	1.590,00 €	19.06. | 04.09. | 11.12.
E-ANG-02	Webanwendungen mit Angular 2	3 Tage	1.590,00 €	28.06. | 23.08. | 25.10. | 13.12.
E-TYPSC-01	TypeScript Grundlagen	2 Tage	1.190,00 €	26.06. | 21.08. | 23.10. | 11.12.
Tools und Verfahren				
P-CI-01	Continuous Integration (CI) Workshop	3 Tage	1.190,00 €	03.07. | 25.09. | 11.12.

WebLogic-Server - Node Manager

Magier im Verborgenen

Mit dem Node Manager steht ein mächtiges Werkzeug im WebLogic-Server zur Verfügung, um Managed-Server überwachen und bei Bedarf nachstarten zu können. Managed-Server sind die Arbeitspferde in einer WebLogic-Infrastruktur, denn auf ihnen laufen geschäftskritische JEE-Applikationen – zumindest sollte es so sein. Da diese Arbeitspferde von zentraler Bedeutung sind, müssen sie sorgfältig und zuverlässig überwacht werden. Und genau da kommt der Node Manager ins Spiel – ein Tool, das schon sehr lange zum WebLogic-Server-Repertoire gehört, welches allerdings im Ruf steht, kompliziert und schwer handhabbar zu sein. Das muss überprüft werden und daher schauen wir uns den Node Manager mit seinen Prinzipien, seinen Einsatzmöglichkeiten und seiner Wirkungsweise mal etwas genauer an.

Node-Manager-Motivation

Beim WebLogic-Server ist die Domain die große, umfassende Klammer, die alle wichtigen WLS-Komponenten wie Managed-Server, Administration Server, Cluster etc. beinhaltet. Dabei übernehmen die Managed-Server die Rolle der produktiven Einheit, d.h. sie hosten JEE-Anwendungen und stellen ihnen ihre Ressourcen, wie CPU und Hauptspeicher, zur Verfügung.

Für den reibungslosen Ablauf von JEE-Anwendungen ist somit die ständige Verfügbarkeit der Managed-Server von existentieller Bedeutung. Sie müssen sorgfältig überwacht werden und in Stresssituationen sind geeignete Reparatur- oder Restart-Maßnahmen vorzunehmen. Und genau da kommen die Stärken des Node Managers ins Spiel.

Node Manager: Simple Aufgabe, schwierige Erfüllung

Der Node Manager hat eine gleichermaßen einfach zu beschreibende wie auch hochkomplexe Aufgabenstellung zu bewältigen – nämlich die Sicherstellung, dass sich Managed-Server jederzeit in einem arbeitsfähigen Zustand

befinden, wobei „jederzeit“ natürlich nicht wortwörtlich gemeint ist. Denn einem großflächigen Stromausfall mit gleichzeitigem Versagen sämtlicher Sicherungssysteme kann man nicht entkommen. In solchen Fällen ist es schlicht nicht möglich, Managed-Server am Leben zu erhalten.

Wir nehmen den eher praxisrelevanten Blickwinkel ein. Der Prozess eines Managed-Servers kann durch unterschiedliche Ereignisse in Schwierigkeiten geraten, z.B. indem der Überlauf des Hauptspeichers droht oder die maximale Anzahl gleichzeitig geöffneter Ressourcen (Dateideskriptoren, Socket-Verbindungen o.Ä.) überschritten wird. Prozessabstürze, die aus so etwas resultieren, sind im Fokus des Node Managers. Sie sollen detektiert und die Server zügig wieder hochgefahren werden.

Die Basisarchitektur

Zunächst jedoch wollen wir einige grundlegende Prinzipien des Node Managers thematisieren. Die Basisfähigkeit des Node Mangers besteht darin, Server starten zu können. Das klingt erstmal wenig aufregend, liefert aber den Schlüssel für das Einsatzgebiet dieses Gehilfen, welches auch in Abbildung 1 schematisch dargestellt ist.

Die Funktionalität und Grundeigenschaften des Node Managers in stringenter, kontinuierlicher Erzählweise vorzustellen, fällt etwas schwer. Daher soll dies stichpunktartig erfolgen, in der Hoffnung, den etwas sperrigen Zugang zu erleichtern.

- In typischen WLS-Umgebungen gibt es genau einen Node Manager pro Rechner. Rechner werden in WLS als Machine repräsentiert und deshalb bearbeitet man in der Admin-Konsole automatisch einen Node Manager, sobald man eine Machine selektiert.
- Das Konzept der Machine-Zugehörigkeit bewirkt, dass alle Konfigurationsinformationen (wie **`host`**, **`port`**, **`protocol`**, …) bei dieser Machine abgelegt sind.
- Dieses Konzept hat zur Folge, dass man standardmäßig jedem Rechner eines Netzwerkes bei der Definition einer Domain eine Machine zuordnet. In Virtualisierungsumgebungen kann es sich natürlich anders verhalten.
- Node Manager sollten ständig laufen, daher empfiehlt es sich, diese als Dienst auf Windows oder Daemon auf Unix zu installieren, sodass sie beim Boot-Vorgang automatisch starten. Deshalb existieren im Gegensatz zu WLS-Servern auch keine expliziten Stop-Skripte.
- Starten lässt sich der Node Manager auch per WebLogic Scripting Toolkit (WLST) mit dem Kommando **`startNodeManager(….)`**.
- In der Datei **`nodemanager.properties`** steht die Hauptkonfiguration. Die Datei wird automatisch beim ersten Start erzeugt, sofern sie nicht schon vorliegt. Ein sehr hilfreiches Prinzip, welches einem auch an anderen Stellen im WLS-Kontext begegnet, z.B. bei der Passwortdatei **`boot.properties`**.
- Beim Starten von Managed-Servern verwendet Node Manager standardmäßig eigene Parameter, die sich in der Admin-Konsole einstellen lassen (siehe Abbildung 2).

Node-Manager-Ausprägung: Shell-Skripte oder Java-Prozess

Die Beschaffenheit des Node Managers kommt in zwei Varianten daher, einmal als Shell-Skripte oder aber als eigenständiger Java-Prozess.

Die Ansammlung von Shell-Skripten gilt als leichter handhabbar, kann allerdings häufig nicht die Einhaltung von allerstrengsten Sicherheitsrichtlinien gewährleisten. Im Fall des Remote-Zugriffs auf den Node Manager (die Admin-Konsole möchte z.B. remote einen Managed-Server starten) muss dieser Zugriff per SSH bewerkstelligt werden.

Zum anderen kann der Node Manager als echter Java-Prozess laufen, der seinen Dienst ressourcenschonend im Hintergrund versieht und im Fall von externer Kommunikation SSL (Secure-Socket-Layer)-Verbindungen nutzen sollte. Das ist unter sicherheitsrelevanten Aspekten betrachtet dringend anzuraten, lässt sich aber auch umgehen, indem man **`plain text`** als Protokoll verwendet. Hierbei darf man nicht vergessen, die Einstellung auf beiden Seiten der Kommunikation vorzunehmen, also sowohl im Administrationsserver als auch beim Node Manager.

Die Funktionsweise des Node Managers lässt sich gut anhand einer kleinen Sequenz von Linux-Kommandos veranschaulichen, mit dem user **`orcale`** auf dem Rechner **`wls01`**. Die Domain besteht aus einem Admin-Server und einem Managed Server (**`Managed01`**).

Der Node Manager läuft im Hintergrund und wenn man den Prozess des Managed-Servers per Kommando **`kill -9`** hart abschießt, wird er unverzüglich neu gestartet (siehe Abbildung 3).

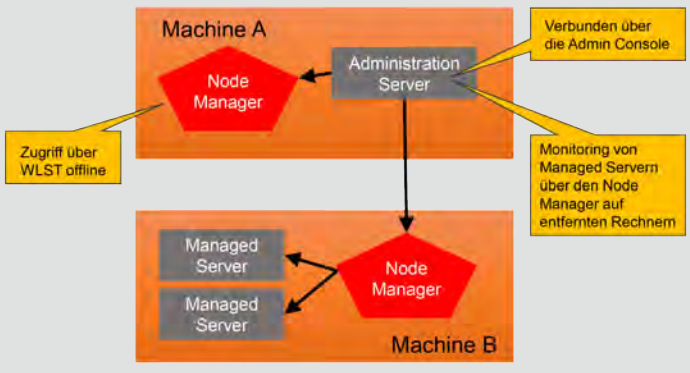


Abb. 1: Übersicht von wichtigen Komponenten in einer WLS-Umgebung. Zwei Rechner (Machine A und B) sind dargestellt, auf denen Node Manager und weitere Server laufen

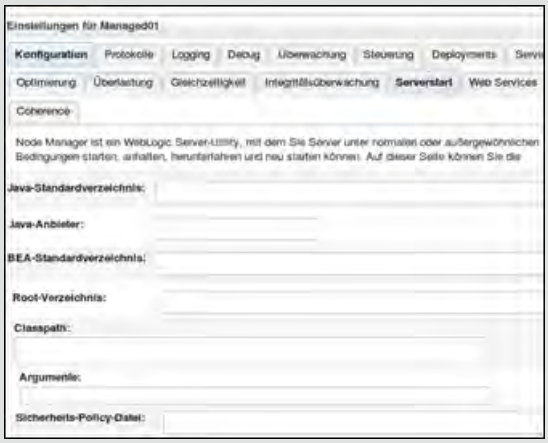


Abb. 2: Hier werden Startparameter für Managed-Server eingetragen, die der Node Manager verwendet

```
[oracle@wls01]$ date; jps –lv|grep weblogic.Server
Mo 27. Feb 10:39:54 CET 2017
4162 weblogic.Server -Dweblogic.Name=Managed01
2693 weblogic.Server -Dweblogic.Name=AdminServer
[oracle@wls01]$ kill -9 4162
[oracle@wls01]$ date; jps –lv|grep weblogic.Server
Mo 27. Feb 10:39:56 CET 2017
4395 weblogic.Server -Dweblogic.Name=Managed01
2693 weblogic.Server -Dweblogic.Name=AdminServer
```

Abb. 3: Neustart des Managed-Servers

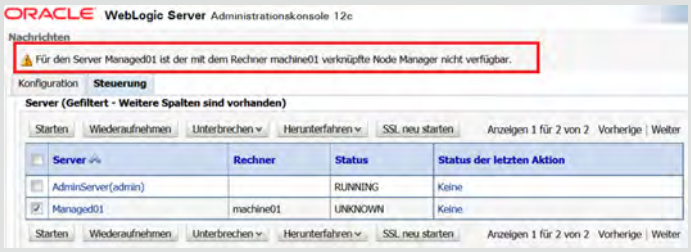


Abb. 4: Auszug der Admin-Konsole beim vergeblichen Versuch, einen Managed-Server zu starten. Das kann nicht funktionieren, denn der zugehörige Node Manager ist nicht verfügbar

Links

[1] Oracle WebLogic Server 12c: Configuring and Using Node Manager
http://www.oracle.com/webfolder/technetwork/tutorials/obe/fmw/wls/12c/12_2_1/01-12-001-ConfiguringandUsingNodeManager/Configuring_and_Using_NM.html#section1

[2] Produktseite Oracle WebLogic Server
https://docs.oracle.com/middleware/1221/wls/

Bildnachweis

Node Manager, Admin-Konsole und WLST: eine Ménage-à-trois

Es umgibt den Node Manager eine gewisse Aura der Geheimnisse oder des unnahbaren, spröden Zugangs zu ihm. Das hat sehr wahrscheinlich seine Ursache darin, dass das Tool wenig bis gar keine Interaktionen mit menschlichen Anwendern benötigt. Der Node Manager ist lediglich initial von Administratoren korrekt einzurichten, agiert als Dienst im Hintergrund und fährt automatisch bei Systemstart hoch. Andererseits kann man ihm keine Beziehungslosigkeit vorwerfen. Im Gegenteil, der Node Manager unterhält eine Ménage-à-trois, eine heftige, vielfältige Dreiecksbeziehung, und zwar einmal mit WLST (WebLogic Scripting Toolkit) und zum anderen mit der Admin-Konsole. Mittels WLST kann der Node Manager z.B. gestartet werden, das Kommando **`nmStart()`** erlaubt dieses und stellt eine variantenreiche Auswahl an Parametereinstellungen bereit. Der Start kann entweder lokal (eigener Rechner) oder remote unter Angabe von Host und Port erfolgen.

Der Zugriff aus der Admin-Konsole heraus wurde schon thematisiert. Beim Start des Managed-Servers aus der Konsole heraus, hat der Node Manager seine Finger im Spiel (siehe Abbildung 4).

Fazit

Dieser Artikel handelt vom Node Manager im WebLogic-Server mit seinen Aufgaben, seinem Einsatzgebiet und der Funktionsweise. Seine oft unterschätze Leistungsfähigkeit ist mit diesem Artikel im Ansatz sichtbar geworden. In einem weiteren Artikel zu diesem Tool wollen wir dann mehr in technische Details eintauchen, denn das WLST-Kommando **`nmEnroll(…)`** beispielsweise, bietet ganz hilfreiche Dienste, die anfangs überraschen und die man später nicht mehr missen möchte.

Selbstverständlich bietet die ORDIX AG im Seminar „WebLogic-Server-Administration“ auch eine Einführung in diese wichtige und spannende Thematik an. Wir helfen Ihnen sehr gerne bei weiteren Fragen.

Hubert Austermeier
(info@ordix.de)


Die ORDIX AG unterstützt gemeinnützige Organisationen

Soziales Engagement

Die Unterstützung von karitativen Organisationen ist ein wesentlicher Bestandteil der ORDIX Philosophie. Es ist zur Tradition geworden, den Verein Sterntaler – Hilfe für schwerkranke Kinder e.V. in Paderborn, die Bärenherz Stiftung in Wiesbaden und das Elisabethstift in Berlin zu mindestens einmal im Jahr finanziell zu unterstützen.

Die ORDIX AG hilft schwerkranken Kindern in Ostwestfalen-Lippe

Die ORDIX AG unterstützt mittlerweile so regelmäßig den Verein Sterntaler e.V., dass es bei der diesjährigen Spendenübergabe zu einem Gegengeschenk kam. Die Initiative hat Ihren Sitz in Paderborn und so bleibt die ORDIX AG damit ihrem Motto treu: „Hilfe vor Ort – Soziales Engagement in Ostwestfalen-Lippe“.

Zur diesjährigen Spendenübergabe am 21. Februar 2017 war Vorstand Benedikt Georgi bei Sterntaler e.V. zu Besuch, um mit einem symbolischen Sparschwein die Spende zu übergeben und sich über die aktuellen Projekte zu erkundigen. Ein ganz besonderes Highlight in diesem Jahr kam von Leiter Wilhelm Stute: Er überreichte Benedikt Georgi im Namen von Sterntaler e.V. die erste Ehrenmitgliedschaft für ORDIX im Club „Sterntaler-Freunde“ für die langjährige und großzügige Unterstützung.

Der gemeinnützige Verein Sterntaler – Hilfe für schwerkranke Kinder e.V. in Paderborn hat es sich zum Ziel gemacht, schwerkranke Kinder, ihre Eltern und Familien in individuellen Notlagen zu unterstützen. Die Hilfe des Vereins besteht insbesondere darin, dass sie kranken Kindern und ihren Familien Wünsche erfüllt, die ohne eine finanzielle Unterstützung nicht möglich wären.

Spendenübergabe in Wiesbaden - Bärenherz Stiftung für schwerstkranke Kinder

Zur Spendenübergabe am 15. Februar 2017 in der Bärenherz Stiftung war auch in diesem Jahr der ORDIX Geschäftsstellenleiter aus Wiesbaden Matthias Jung vor Ort, um die finanzielle Unterstützung symbolisch zu übergeben.

„Für uns und mich war es eine ganz besondere Freude, Herrn Jung erneut hier in der Geschäftsstelle zur Spendenübergabe begrüßen zu dürfen“, so Anja Eli-Klein, stellvertretende Geschäftsführerin und Stiftungsmanagerin.

Die Stiftung unterstützt seit 1999 Einrichtungen für Familien mit Kindern, die unheilbar erkrankt sind und eine geringe Lebenserwartung haben, insbesondere Kinderhospize.

Spendenübergabe in Berlin im Elisabethstift

Der Förderverein durfte am 28. Februar 2017 Martin Hoermann, Ausbildungsleiter der ORDIX AG, in der Zentrale in Berlin begrüßen. Für die erneute Unterstützung bedankte sich Helmut Wegener, Geschäftsführer und Leiter des Elisabethstifts, herzlich und informierte den Spendenüberbringer über die aktuelle Lage der Stiftung.

„Wir sind für in Not geratene Menschen“, so das Leitbild der Berliner Stiftung, die neben finanzieller Hilfe auch dringend auf Sachspenden angewiesen ist. Zudem liegt es den Verantwortlichen auch sehr am Herzen, Patenschaften für Kinder und Gruppen zu vergeben, Werbeträger oder ehrenamtlicher Helfer zu finden. Das Elisabethstift in Berlin ist eine Kinder- und Jugendhilfeeinrichtung in Berlin-Brandenburg und eines der ältesten Kinderheime Berlins.

Die Stiftung Talentengel fördert Hochbegabte

Für die gemeinnützige Stiftung Talentengel unseres Firmengründers Wolfgang Kögler und seiner Frau Ulrike Kögler haben wir ein Stipendium eingerichtet, welches junge und motivierte Talente in Ostwestfalen unterstützen soll.

„Uns als Stiftern liegt die Förderung von Talenten im musischen, sportlichen und mathematisch-naturwissenschaftlichen Bereich von jeher am Herzen. Aus unserer persönlichen Verbundenheit zu der Stadt Paderborn und der Region OWL möchten wir unseren Beitrag leisten, um dauerhaft und zukunftsorientiert Talente im ostwestfälischen Bereich zu fördern“, so Ulrike Kögler.

Alle Informationen zur Vergabe der Stipendien sowie zur Spendenunterstützung finden Sie auf unserer Internetseite www.ordix.de/unternehmen/soziales-engagement.html und natürlich unter www.talentengel.org.

Softwareentwicklung auf einem höheren Level:

Continuous Integration – Warum und wofür?

Die agile Softwareentwicklung versucht mit wenigen festgelegten Regeln, geringem Aufwand und meist einem iterativen Vorgehen, den Entwicklungsprozess voranzutreiben. Doch oft kommt es zu Problemen, wie instabile Build-Prozesse, schwierige Integration von Komponenten und unzureichende Testabdeckungen. Continuous Integration (CI) ist hier ein bewährtes Mittel zur Verbesserung und Optimierung des Softwareentwicklungsprozesses. Doch wie setzt man CI richtig ein und welche Regeln sind zu beachten? In diesem Artikel zeigen wir auf, wie der Einsatz von Continuous Integration dabei helfen kann, den Entwicklungsprozess zu optimieren. Zudem kann über entsprechende Analysewerkzeuge die Softwarequalität gesteigert und die Testabdeckung verbessert werden.

Einführung

Neue Softwarekomponenten entwickeln, Anwendungsteile integrieren und verlinken, Unit-Tests schreiben, Build-Prozess starten – fertig!?

Der Softwareentwicklungsprozess funktioniert nicht nach diesem Schema und nicht so „einfach“ wie beschrieben. In der Realität kommt es oftmals zu fehlschlagenden Builds, die nicht zuletzt auf mangelnde Testabdeckung zurückzuführen sind. Das Ganze spiegelt oft die Qualität der Software wieder. Gerade moderne Softwareprojekte gewinnen immer mehr an Komplexität und lassen sich immer schwieriger verwalten. Anwendungsteile werden unabhängig an verschiedenen Standorten und von unterschiedlichen Entwicklerteams implementiert. Entsprechend gibt es schon seit langem Ideen und Methoden, um den Softwareentwicklungsprozess zu optimieren und zu verbessern. Doch erst mit Martin Fowlers, im Jahre 2000 veröffentlichten Artikel „Continuous Integration“, gewann das Thema unter dieser Definition zunehmend an Bedeutung. Seither werden in vielen Projekten Build-Server eingesetzt, um den Softwareentwicklungsprozess zu unterstützen, zu überwachen und zu automatisieren. In agilen Softwareentwicklungsprojekten ist Continuous Integration Pflicht!

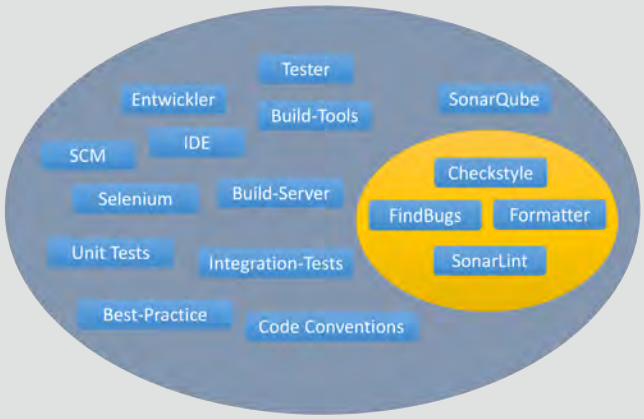


Abb. 1: Was ist Continuous Integration?

Continuous Integration?

Die Frage, was genau Continuous Integration ist, kann nicht mit einem Satz beantwortet werden. Vielmehr lässt sich Continuous Integration als ein Vorgehen bezeichnen, das definierte Tools, Systeme und Methoden verbindet, um eine Automatisierung und Verbesserung des Entwicklungsprozesses zu erreichen. Das Ziel besteht darin, komplexe Softwareprojekte beherrschbar zu gestalten und damit gleichzeitig die Softwarequalität zu steigern. Continuous Integration soll dabei aber nicht nur helfen, das Softwareprojekt automatisiert zu kompilieren, sondern auch automatisiert Modul- und Integrationstests durchzuführen. Es gilt die Prämisse: Die Software befindet sich immer auf einem auslieferbaren Stand.

Abbildung 1 soll verdeutlichen, dass Continuous Integration eine Vielzahl von Elementen verbindet. Der Fokus aller

Aspekte liegt immer auf der Optimierung des Entwicklungsprozesses und der frühzeitigen Erkennung von Fehlern bei der Integration von Anwendungsteilen. Darüber hinaus soll CI den effektiven Einsatz von Techniken und Werkzeugen unterstützen, um die Softwarequalität auf einem hohen Level zu halten. So können vor allem nicht funktionale Anforderungen, wie beispielsweise Wartbarkeit und Erweiterbarkeit, verbessert und so die Lebensdauer des Softwareproduktes verlängert werden.

Abbildung 1 zeigt auch Begriffe, wie Best Practice, Code Coventions und Code-Analyse-Werkzeuge, FindBugs, Checkstyle und SonarLint, die zunächst nur mittelbar mit Continuous Integration in Verbindung stehen. Aber gerade diese mittelbaren Komponenten des CI-Systems helfen komplexen Softwareprojekten, auch die Softwarequalität zu wahren und zu verbessern. Diese Plugins sollten mit Projektbeginn in den Entwicklungsumgebungen (aller Entwickler) installiert werden, um eine einheitliche Codebasis zu schaffen und so das Risiko möglicher auftretender Fehler bereits im Vorfeld zu minimieren.

Die genannten Werkzeuge erfüllen dabei unterschiedliche Aufgaben. Formatter dienen der einheitlichen Darstellung und Strukturierung des Sourcecodes. Checkstyle übernimmt die Prüfung des Programmierstils und erzwingt die Einhaltung von Codierrichtlinien, während FindBugs eine statische Codeanalyse zur Erkennung bekannter Fehlermuster durchführt.

Funktionsweise des CI-Systems

Der Build-Server (CI-Server genannt) fungiert als zentrale Integrationsinstanz und steuert den gesamten Build-Prozess. Als Beispiele für Build-Server seien an dieser Stelle CruiseControl, Hudson/Jenkins und Bamboo (Atlassian) genannt. Ein CI-System besteht neben dem Build-Server noch aus weiteren notwendigen Komponenten. Abbildung 2 zeigt die Hauptbestandteile eines CI-Systems.

Wie aus Abbildung 2 hervorgeht, wird der CI-Server mit einem Source-Code-Management-System verbunden, das den Quellcode des Projektes zentral verwaltet. In der Praxis handelt es sich hierbei oft um SVN- oder Git-Repositories. Der CI-Server erzeugt dabei ähnlich wie auf den Entwicklungsrechnern eine lokale Arbeitskopie des Projektes. In regelmäßigen Abständen schickt der CI-Server-Anfragen an das angebundene Versionsverwaltungssystem und prüft, ob es Änderungen am Quellcode gegeben hat. Ausgelöst durch Änderungen im SCM seit dem letzten CI-Build, wird ein neuer Durchlauf angestoßen. Dabei wird zunächst die lokale Arbeitskopie auf dem CI-Server aktualisiert (Update aus dem SCM) und anschließend das gesamte Projekte „gebaut“. Dieser Vorgang beinhaltet das Kompilieren, Packen und das Ausführen automatisierter Unit-Tests. Der Build-Prozess wird dabei nicht vom Build-Server direkt ausgeführt, sondern von einem integrierten Build-Tool, wie beispielsweise Apache Ant, Maven oder Gradle.

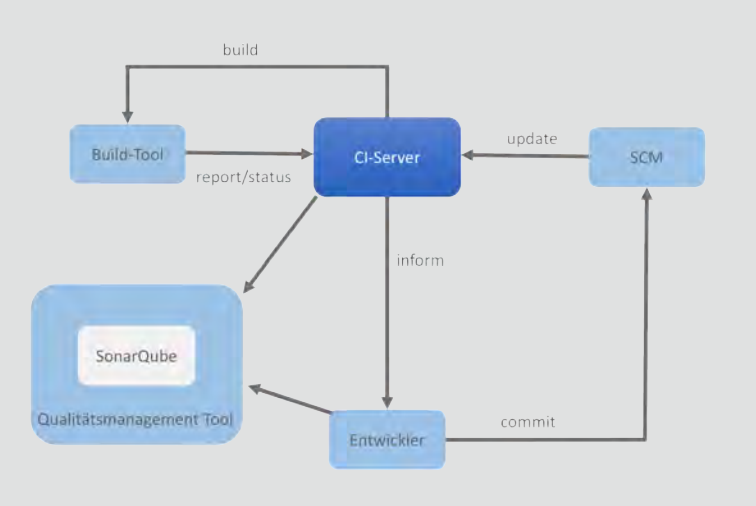


Abb. 2: Bestandteile eines CI-Systems

Ein CI-System lässt sich um beliebige weitere Komponenten erweitern. So kann ein Code-Review-System, wie zum Beispiel BitBucket (Atlassian) oder Gerrit (bei Git als SCM), vorgeschaltet werden. Die Änderungen an der Software können so vor Integration in das Projekt nach dem „Vier-Augen-Prinzip“ von einem oder mehreren Teammitgliedern überprüft werden. Erst wenn Änderungen freigegeben wurden, werden diese in das zentrale Repository übernommen und vom Build-Server abgegriffen und gebaut.

Der Continuous-Integration-Server liefert für jeden Build-Prozess einen Status zurück und meldet diesen an das Entwicklungsteam. Fehlerhaften Builds ist dabei immer, vor der Implementierung neuer Features, oberste Priorität einzuräumen!

➔ Die Software ist wieder auf einen auslieferungsfähigen Stand zu bringen!

Grundprinzipien und Voraussetzungen für den Einsatz von CI

Die Verwendung von Continuous Integration ist an bestimmte Voraussetzungen gebunden. Um Continuous Integration erfolgreich zu betreiben, bedarf es der Einhaltung definierter Grundprinzipien:

- ein gemeinsames zentrales Repository
- automatisierte Build-Prozesse
- automatisierte Tests
- jede Änderung wird mit Integration in das Gesamtsystem gebaut und getestet
- Änderungen sollen regelmäßig und oft in das Gesamtsystem integriert werden
- der Build sollte stabil und schnell laufen
- produktionsnahe Umgebung für Testzwecke
- einfache Verwaltung aller Sourcen und Dokumente durch gemeinsame Repositorien und Wiki-Seiten

Die aufgeführten Punkte stellen zunächst einmal keine Neuheiten in der Softwareentwicklung dar. In den meisten Projekten finden Build-Tools und Repositorien bereits Verwendung. Vielmehr verfolgt CI den Ansatz, dass das genutzte zentrale Repository von einem Build-Server überwacht und regelmäßig gebaut wird und Modul- und Integrationstests durchgeführt werden. Das Feedback des Build-Servers hilft dem Entwicklungsteam, Fehler frühzeitig zu erkennen und auf das Fehlerverhalten der Software sofort reagieren zu können, wenn die Software sich nicht mehr in einem auslieferbaren Zustand befindet.

Es obliegt dem Entwicklerteam, keinen nicht funktionalen und ungetesteten Sourcecode einzuchecken. Das klingt zunächst banal, wird aber leider oft nicht beachtet. Weiterhin dürfen keine Änderungen eingecheckt werden, solange der lokale (Entwickler-) Build nicht stabil läuft und alle Fehler beseitigt sind. Fehlschlagende Builds haben immer Priorität vor der Implementierung neuer Features, da der Entwicklungsstand gefährdet ist.

Fungiert der CI-Server auch als Analysewerkzeug, müssen Entwickler ebenfalls kontinuierlich die erstellten Auswertungen beobachten und damit die Qualität der Software im Auge behalten.

Software-Qualitätsmanagement mit CI

Der CI-Server ist noch viel mehr als nur eine Code-Verwaltungszentrale zur Überwachung von Build-Prozessen. Wie im vorherigen Abschnitt angedeutet ist er ein zentraler Baustein, um ein umfangreiches Software-Qualitätsmanagement zu erreichen. Die automatisiert erstellten Build-Ergebnisse lassen sich mit Tools wie z.B. SonarQube automatisch analysieren. Mithilfe dieser Analysen können u.a. Aussagen zu Qualität und Testabdeckung der Software gemacht werden.

SonarQube erfindet das Rad hierbei nicht neu, sondern integriert bereits bekannte Funktionen aus Werkzeugen, wie Checkstyle, Findbugs, PMD und weitere. Die ausgewerteten Build-Ergebnisse werden an den SonarQube-Server übermittelt und auf einer Weboberfläche in Form von Metriken und Grafiken visualisiert. Die Entwicklung der Software bzgl. Qualitätsmerkmalen und Metriken lassen sich so über die Zeit hinweg vergleichen und helfen, unerwünschte Veränderungen zu erkennen.

Seit der SonarQube Version 4.3 lassen darüber hinaus sogenannte Quality-Gates bestimmen. Diese definieren, ob die aktuelle Version der Software einen bestimmten Reifegrad (qualitatives Level der Software) hat, der z.B. für ein Release ausreichend gut ist. Diese Quality-Gates können für Projekte auf Basis verschiedener Metriken definiert und angepasst werden.

In Verbindung mit der Nutzung eines SonarQube-Servers kann über das Plugin SonarLint die technische Bewertung der Softwarequalität „on-the-fly“, während des Entwicklungsprozesses durchgeführt und eingesehen werden. Der genutzte SonarQube-Server wird dabei in der Entwicklungsumgebung an das Projekt gebunden. Die Ergebnisse der SonarQube-Analyse werden direkt in der IDE angezeigt und die betroffenen Stellen im SourceCode markiert.

Ausblick

Aktuell lässt sich beobachten, dass die Anforderungen an moderne Softwaresysteme einem starken Wandel unterworfen sind. Höhere Flexibilität und kürzere Deploymentzyklen sollen für ein schnelleres „Time To Market“ sorgen. Zu meist streben auch das Management und oft die Fachbereiche an, Softwarepakete, Produkte oder Änderungen häufiger und flexibler am Markt zu platzieren. Microservices sollen große und träge monolithische Systeme ablösen.

In diesem Zusammenhang haben sich Methoden, wie DevOps [1] und Microservices [2] etabliert, die einen Umbruch in der Prozess- und Team-Kultur mit sich bringen. Der DevOps-Gedanke sieht vor, dass ein Team für ein Produkt als Ganzes von der Planung bis zum Betrieb verantwortlich ist. Grenzen zwischen Entwicklung und Betrieb werden aufgelöst. In Konsequenz wird ein Prozess angestrebt, der eine Software qualitätsgesichert und höchstmöglich automatisiert und bis in die Produktion bringt. Im Kontext von Microservices werden sogenannte Querschnitt-Teams gebildet, die für einen definierten Themenkomplex komplett zuständig sind. Die Verantwortung wird damit auf ein Team verlagert und so Reibungsverluste, sowie Interessenkonflikte (Blame Game) an Abteilungsgrenzen vermieden.

Continuous Integration liefert als Ergebnis ein Stück funktionsfähige (ausführbare) Software. Eine sogenannte Continuous Delivery Pipeline erweitert nun den CI-Prozess um die Teststufen zur Qualitätssicherung. Dazu gehören z.B. Last- und Performance-Tests oder Acceptance-Tests. Möglich sind u.a. auch manuelle Smoke-Tests. Ziel bleibt immer ein möglichst hoher Automatisierungsgrad, auch wenn am Ende eine manuelle Freigabe für das Deployment in Produktionsumgebungen stehen kann.

Fazit

Continuous Integration ist ein bewährtes Mittel, komplexe Projekte beherrschbarer zu gestalten, und zwar unter Einhaltung von Regeln und Richtlinien sowie der notwendigen Disziplin von allen Projektbeteiligten. Nur so lässt sich der Entwicklungsprozess optimieren und die Softwarequalität durchweg kontinuierlich einhalten und verbessern. Continuous Integration ist kein Allheilmittel, jedoch als Basis für moderne Softwareentwicklungsprozesse heute unverzichtbar und sollte zum Standardrepertoire eines jeden Teams gehören.

Fehler halten sich nicht an einen Release-Plan und können immer auftreten! Continuous Integration soll jedoch das Risiko der Auslieferung von fehlerhafter Software minimieren. Eine möglichst vollständig getestete Software

setzt viele Unit-Tests voraus, für die oft zu wenig Zeit eingeschätzt und aufgewendet wird. Des Weiteren lässt sich die konsequente Einhaltung aller Methoden und Regeln nicht immer mit den Projektanforderungen und Terminen vereinbaren. Doch die Verwendung von Plugins, wie Checkstyle, Findbugs und Code-Fromattern, funktioniert auch ohne Termine und Projektanforderungen. Solche Mittel und Tools sind notwendig, um die Softwarequalität von vornherein zu schützen.

Dennoch ist es an den Entwicklern, die Verantwortung für den Code und für seine gewissenhafte Programmierung zu übernehmen. Das bedeutet auch, diesen zu testen. Unterschreiben Sie einen Vertrag, dessen Inhalt und Details Ihnen nicht vollumfänglich bekannt sind?

- Ein Commit in das SCM ist die Unterschrift eines Entwicklers, dass der Code nicht nur funktioniert, sondern auch lesbar, dokumentiert und ausreichend getestet ist -

Sebastian Grimm
(info@objectsystems.de)

Glossar

Blame Game
Als Blame Game werden Interessenkonflikte und gegenseitige Schuldzuweisen zwischen Abteilungen (oft Entwicklung und Betrieb), bei Produktivsetzung einer neuen Software/Software-Version bezeichnet.

CI
Continuous Integration beschreibt den Prozess des fortlaufenden Zusammenfügens von Komponenten zu einer Anwendung. Das Ziel der kontinuierlichen Integration ist die Steigerung der Softwarequalität. Typische Aktionen sind das Übersetzen und Linken der Anwendungsteile, prinzipiell können aber auch beliebige andere Operationen zur Erzeugung abgeleiteter Informationen durchgeführt werden. Üblicherweise wird dafür nicht nur das Gesamtsystem neu gebaut, sondern es werden auch automatisierte Tests durchgeführt und Softwaremetriken zur Messung der Softwarequalität erstellt. Der gesamte Vorgang wird automatisch ausgelöst durch Einchecken in die Versionsverwaltung.

VCS
Version Control System - Die Versionsverwaltung ist eine Form des Variantenmanagements; dort sind verschiedene Sprachvarianten oder modal auch anders bestimmte Varianten möglich.

Links/Quellen

[1] DevOps:
https://de.wikipedia.org/wiki/DevOps

[2] Microservices:
https://de.wikipedia.org/wiki/Microservices

[3] ORDIX seminare - Continuous Integration Workshop:
https://seminare.ordix.de/seminare/entwicklung/course/1855-P-CI-01

[Q1] Flower, Martin (2006): Continuous Integration
http://martinfowler.com/articles/continuousIntegration.html

Data Mining in der Praxis (Teil II):

Klassifikation

Ist eine E-Mail Spam oder nicht? Welcher Kundengruppe gehört ein Neukunde wahrscheinlich an und mit welcher Werbeansprache kann ich ihn am besten erreichen? Solche und verwandte Fragen sind typische Aufgaben für eine Klassifikation. Aufgrund vorhandener Daten werden Modelle erstellt, die für neue Daten Vorhersagen treffen können. In diesem Artikel erhalten Sie einen ersten Überblick über diesen, im Detail komplexen Teilbereich, des Data Mining.

Was ist Klassifikation?

Aufgrund der Ausprägung der Attribute eines Datensatzes kann dieser einer bestimmten Kategorie zugeordnet werden. So entscheidet der Inhalt, Betreff und Absender einer E-Mail darüber, ob diese Mail der Kategorie „Spam“ oder „kein Spam“ angehört. Ein zweites Beispiel sind die Warenkörbe eines Onlinekunden, aus denen sich Rückschlüsse über das Geschlecht, die Altersklasse und Preisaffinität ziehen lassen. Die gewonnenen Informationen können im Fall der E-Mail zur automatischen Verschiebung einer Spam-Mail in einen bestimmten Ordner, oder bei dem Onlinekunden zur gezielten Ansprache mit Werbung genutzt werden.

```
In [1]: import matplotlib.pyplot as plt
        %matplotlib inline
        import random
        import numpy as np
        import pandas as pd
        from sklearn import datasets, svm, cross_validation, tree, preprocessing, metrics
        import sklearn.ensemble as ske
```

Abb. 1: Importieren der eingesetzten Bibliotheken

Wie funktioniert Klassifikation?

Aufgrund der vorhandenen, historischen Daten wird ein Modell erstellt. Dieses Modell wird bei einem neuen Datensatz genutzt, um eine Aussage über die wahrscheinlichste Kategorie zu treffen, der dieser Datensatz angehört. Bei dem Modell handelt es sich um eine mathematische Funktion, die aufgrund statistischer Zusammenhänge ermittelt wird. Die Daten sind Eingabewerte für die Funktion, die Kategorie ist das ermittelte Ergebnis.

```
In [2]: titanic_df = pd.read_excel('titanic3.xls', 'titanic3', index_col=None, na_values=['NA'])
        titanic_df.head()
```

Out[2]:

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	1	1	Allen, Miss. Elisabeth Walton	female	29.0000	0	0	24160	211.3375	B5	S	2	NaN	St Louis, MO
1	1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.5500	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON
2	1	0	Allison, Miss. Helen Loraine	female	2.0000	1	2	113781	151.5500	C22 C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON
3	1	0	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1	2	113781	151.5500	C22 C26	S	NaN	135.0	Montreal, PQ / Chesterville, ON
4	1	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1	2	113781	151.5500	C22 C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON

Abb. 2: Laden der vorhandenen Daten und Ausgabe der ersten fünf Datensätze

Abgrenzung Klassifikation zu Clusteranalyse

Im Einführungsartikel (Seite 6) der Reihe Data Mining wurde neben der Klassifikation auch die Clusteranalyse als eine mögliche Aufgabe des Data Mining erwähnt. Bei der Clusteranalyse werden aus Daten einer bestimmten Fragestellung im Vorfeld, nicht bekannte Kategorien ermittelt. Es werden ähnliche Datensätze gruppiert und diese Gruppen bilden die Kategorien. So kann zum Beispiel die geografische Position des Wohnortes eines Kunden genutzt werden, um eine Kategorisierung bezüglich des Wohnortes vorzunehmen. Je nach gewählter Granularität ergeben sich für die deutsche Bevölkerung Kategorien um die großen Städte und Ballungszentren herum. Die Grenzen zwischen zwei Kategorien sind nicht fest gezogen, sondern werden vom Algorithmus bei jedem zusätzlich untersuchten Datensatz neu gesetzt. Wertvoll kann eine solche Analyse zum Beispiel bei der Planung der Filialen eines Unternehmens sein. Wenn zum Beispiel fünf Filialen geplant sind, wird die Clusteranalyse mit dem Ziel von fünf Kategorien durchgeführt. Wenn die neuen Filialen im geografischen Zentrum dieser Kategorien liegen, ist der durchschnittliche Weg der Kunden zu einer Filiale minimiert.

Während bei der Clusteranalyse die Kategorien nicht bekannt sind, sondern gerade die Ermittlung dieser die Aufgabe der Analyse darstellt, sind bei der Klassifikation die Kategorien vorab bekannt. Neue Datensätze werden entsprechend ihrer „Ähnlichkeit“ zu vorhandenen Datensätzen einer Kategorie zugeordnet.

Wegen dem engen inhaltlichen Zusammenhang erfolgt in der Praxis häufig auf noch nicht untersuchten Daten zuerst eine Clusteranalyse, um sinnvolle Kategorien zu ermitteln. In einem zweiten Schritt werden diese dann im Rahmen einer Klassifikation auf neue Daten angewendet.

Vorbereitung

Bevor ein Modell zur Klassifikation erstellt werden kann, muss sichergestellt werden, dass die zugrunde liegenden Daten „sauber“ sind.

```
In [3]: titanic_df['survived'].mean()
Out[3]: 0.3819709702062643

In [4]: titanic_df.groupby('pclass').mean()
```

Out[4]:

pclass	survived	age	sibsp	parch	fare	body
1	0.619195	39.159918	0.436533	0.365325	87.508992	162.828571
2	0.429603	29.506705	0.393502	0.368231	21.179196	167.387097
3	0.255289	24.816367	0.568406	0.400564	13.302889	155.818182

Abb. 3: Durchschnittliche Überlebensrate allgemein und nach Passagierklasse gruppiert

```
In [5]: class_sex_grouping = titanic_df.groupby(['pclass','sex']).mean()
        class_sex_grouping['survived'].plot.bar()
Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x7f0653a65150>
```

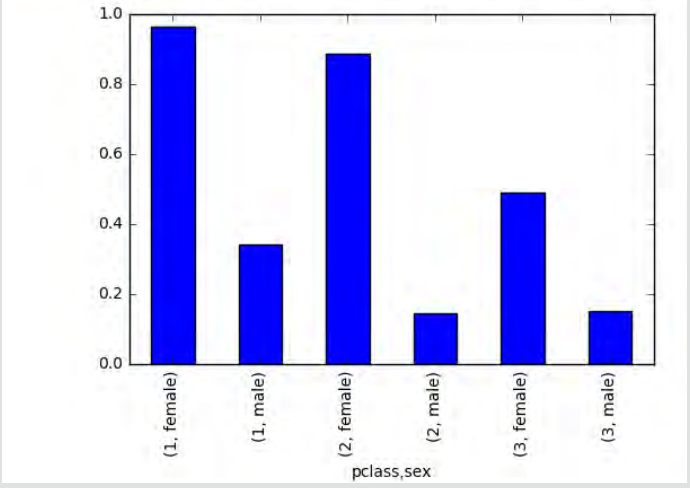


Abb. 4: Durchschnittliche Überlebensrate in Abhängigkeit vom Geschlecht

Die Modellerstellung ist ein mathematisches Verfahren, das von sich aus nicht in der Lage ist, korrekte von nicht korrekten Daten zu unterscheiden.

In einem Beispiel verkauft ein Onlineshop Hörgeräte. Die aus fachlicher Sicht erwartete Kundschaft ist von eher gesetztem Alter. Bei der Registrierung im Onlineshop setzt die Software das Alter auf 0, wenn keine Angabe durch den Kunden gemacht worden ist. Bei der Kategorisierung wird nun das Alter, neben weiteren Attributen, zur Ermittlung des zu erwartenden Umsatzes je Kunde herangezogen. Als Ergebnis ermittelt das Modell, dass Kunden unter 20 Jahren einen hohen Umsatz erwarten lassen. Tatsächlich sind es aber nicht die 18 und 19 Jahre alten Kunden, die vermehrt Hörgeräte kaufen, sondern die vielen Kunden mit „0 Jahren“, die die Daten verfälschen. Wenn das Modell nun auf einen neuen Kunden mit 18 Jahren angewendet wird, wird diesem eine hohe Umsatzkategorie prognostiziert und somit unnötige Werbegelder für eine persönliche Ansprache ausgegeben.

```
In [6]: titanic_df.count()

Out[6]: pclass       1309
        survived     1309
        name         1309
        sex          1309
        age          1046
        sibsp        1309
        parch        1309
        ticket       1309
        fare         1308
        cabin         295
        embarked     1307
        boat          486
        body          121
        home.dest     745
        dtype: int64

In [7]: titanic_df = titanic_df.drop(['body','cabin','boat'], axis=1)
        titanic_df["home.dest"] = titanic_df["home.dest"].fillna("NA")
        titanic_df = titanic_df.dropna()
        titanic_df.count()

Out[7]: pclass       1043
        survived     1043
        name         1043
        sex          1043
        age          1043
        sibsp        1043
        parch        1043
        ticket       1043
        fare         1043
        embarked     1043
        home.dest    1043
        dtype: int64
```

Abb. 5: Datenbereinigung

```
In [8]: def preprocess_titanic_df(df):
            processed_df = df.copy()
            le = preprocessing.LabelEncoder()
            processed_df.sex = le.fit_transform(processed_df.sex)
            processed_df.embarked = le.fit_transform(processed_df.embarked)
            processed_df = processed_df.drop(['name','ticket','home.dest'],axis=1)
            return processed_df
        processed_df = preprocess_titanic_df(titanic_df)
```

Abb. 6: Transformation der Daten entsprechend der Anforderungen des Algorithmus

```
In [9]: X = processed_df.drop(['survived'], axis=1).values
        y = processed_df['survived'].values
        X_train, X_test, y_train, y_test = cross_validation.train_test_split(X,y,test_size=0.2)
```

Abb. 7: Aufteilen der Daten in Trainings- und Validierungsdaten

```
In [10]: clf_dt = tree.DecisionTreeClassifier(max_depth=10)
         clf_dt.fit (X_train, y_train)
         clf_dt.score (X_test, y_test)

Out[10]: 0.78947368421052633
```

Abb.8: Ausführen der eigentlichen Klassifikation [6]

Einem Fachanwender, oder in diesem einfachen Fall jedem Menschen, wird auffallen, dass an dem Modell etwas nicht stimmt. Der Algorithmus selber kann dies nicht feststellen – er arbeitet immer mit den gegebenen Daten, mögen sie sinnvoll sein oder nicht.

Das Beispiel verdeutlicht, wie wichtig die Verwendung valider Daten für die Modellerstellung ist. Im obigen Beispiel hätte man sinnvoll Kunden mit Alter „0 Jahre“ aus der Modellbildung herausgelassen. Ein Modell wird mit steigender Anzahl an Trainingsdaten immer besser, wenige falsche Daten kehren diesen Effekt aber schnell ins Gegenteil.

Trainingsdaten und Validierungsdaten

Zur Erstellung eines Modells werden Daten benötigt. Die Daten, die hierfür verwendet werden, bezeichnet man als Trainingsdaten. Um die Qualität eines Modells zu testen, wendet man es nach dem Training auf bekannte Daten an. Für jeden Datensatz aus diesen Validierungsdaten ist die korrekte Kategorie bekannt. Sie kann mit der durch das Modell ermittelten Kategorie verglichen werden. Der prozentuale Anteil korrekt klassifizierter Datensätze ist eine Kennzahl, die die Qualität eines Modells beschreibt.

Je größer die zur Verfügung stehende Menge an historischen Daten ist, desto besser wird ein Modell sich in der Klassifikation bewähren. Auf der anderen Seite benötigt man auch eine nennenswert große Menge an Testdaten, um die Qualität des Modells zu bestimmen. Wenn der Umfang der Validierungsdaten zu gering gewählt ist, können statistische Ausreißer das Ergebnis verfälschen. Als Daumenregel hat es sich bewährt, die zur Verfügung stehenden Daten zufällig verteilt auf 80% als Trainingsdaten und 20% als Validierungsdaten zu verwenden.

Featureselection und Overfitting

In einem Datensatz sind in der Regel viele Attribute vorhanden. Nicht alle sind für die Ermittlung der angestrebten Klassifikation relevant. So mag in einem Kundendatensatz der Wohnort, das Alter und Geschlecht sowie der ausgeübte Beruf relevant für die Klassifikation bezüglich des zu erwartenden Umsatzes mit Handys sein. Die

Attribute „Augenfarbe“ oder „Schuhgröße“ sind es wahrscheinlich nicht. Bei diesem einfachen Beispiel ist sofort ersichtlich, dass die beiden genannten Attribute nicht zur Qualität des gesuchten Modells beitragen. In der Praxis existieren aber unter Umständen Hunderte Attribute. Selbst eine Fachabteilung kann oftmals keine fundierte Aussage treffen, welche Attribute für ein Modell relevant sind und welche nicht. Ein einfacher Ansatz wäre es, alle zur Verfügung stehenden Attribute in die Modellerstellung mit einzubeziehen. Es gibt zwei Argumente, die gegen einen solchen Ansatz sprechen:

- Viele Attribute machen ein Modell komplex. Ist zum Beispiel das Alter aus dem Beispiel der Umsätze mit Hörgeräten ein integraler Bestandteil eines Modells, dann können keine guten Vorhersagen für Datensätze getroffen werden, bei denen die Information Alter nicht vorliegt. Je komplexer ein Modell wird, desto größer ist die Gefahr, dass bei der anschließenden Klassifikation neue Datensätze einzelne Attribute nicht enthalten und somit nicht zuverlässig klassifiziert werden können.
- Ein Modell kann einen vermeintlichen Zusammenhang des Umsatzes mit der Augenfarbe „erkennen“. Tatsächlich ist dieser ermittelte Zusammenhang aber rein zufällig. Insbesondere wenn die Menge der Validierungsdaten zu gering gewählt wurde, neigen Modelle zu solchen nicht korrekten Klassifikationen. Man spricht davon, dass ein Modell „overfitted“ wurde. Es ist also zu sehr an Trainingsdaten angepasst und verallgemeinert zu wenig. Es ist zu stark abhängig von einzelnen, mitunter fachlich irrelevanten Attributen.

Ein gutes Modell verallgemeinert gut und verwendet dazu eine möglichst kleine Anzahl von Attributen. Die Auswahl der tatsächlich eingesetzten Attribute wird mit Featureselection bezeichnet und ist ein wesentlicher Aspekt bei der Modellbildung.

Beispiel Klassifikation

Die folgenden Beispiele sind in der Sprache Python erstellt. Python und R sind die beiden wichtigsten Sprachen im Umfeld von Data Mining. Es handelt sich jeweils um Skriptsprachen, die sich durch eine einfache Syntax und eine Vielzahl existierender Hilfsbibliotheken auszeichnen. Zum Einsatz von Python ist es empfehlenswert, anaconda [1] zur Verwaltung der Bibliotheken und Jupyter Nodebook [2] als interaktiven Editor zu installieren.

Ähnlich dem „Hello World“ für einen Programmierer ist der Titanic-Datensatz im Bereich Data Mining ein häufig genutzter Einstieg für die Beschäftigung mit dem Thema Klassifikation. Zu jedem Passagier des untergegangenen Schiffes ist bekannt, ob er in die Kategorie **`survived`** oder **`not survived`** gehört.

Nachdem die benötigten Bibliotheken importiert worden sind (siehe Abbildung 1), werden die eigentlichen Daten [3] geladen (siehe Abbildung 2). Der Befehl **`head()`** gibt die ersten fünf Datensätze aus. Neben dem Alter, Geschlecht und Namen sind auch Informationen zur Passagierklasse, dem Ticketpreis und dem Abfahrtsort vorhanden. Und natürlich mit **`survived`**, die Kategorie, wegen der die Klassifikation durchgeführt werden soll.

Vor der Modellbildung ist es sinnvoll, sich einen Überblick über die Daten zu verschaffen. In Abbildung 3 wird die durchschnittliche Überlebensrate mithilfe des Befehls **`mean()`** errechnet. Angewendet wird der Befehl auf dem Datenobjekt **`titanic_df`** auf dem Attribut **`survived`**. Sie ist mit 38,2% leider recht gering. Die Vermutung, dass die Chance, den Untergang zu überleben, für Passagiere der ersten Klasse höher lag als für das Fußvolk in der dritten Klasse, wird mit einem **`groupby`** auf dem Attribut **`survived`** bestätigt. Wer sich also die durchschnittlichen £ 87,51 für ein Ticket der ersten Klasse nicht leisten konnte, hatte in der dritten Klasse nur noch eine Überlebenschance von 25,6%. In Abbildung 4 werden die nach Passagierklasse und Geschlecht gruppierten Daten in einer neuen Variablen **`class_sex_grouping`** gespeichert und dann mithilfe von **`plot.bar()`** als Balkengrafik ausgegeben. Man erkennt, dass Frauen über alle Klassen hinweg eine deutlich höhere Überlebenschance hatten als Männer. Da sich dies adäquat auch für Kinder sagen lässt, galt damals in jedem Fall: „Frauen und Kinder zuerst“.

Bereits dieser erste Überblick über die Daten bescheinigt Frauen aus der ersten Klasse die höchste Überlebenschance. Wenn man die Daten genauer untersucht, findet man schnell weitere derartige Abhängigkeiten. Ein Klassifikationsalgorithmus dient dazu, in den Daten befindliche Zusammenhänge automatisiert zu entdecken. Bevor ein solcher Algorithmus allerdings eingesetzt werden kann, müssen die Daten aufbereitet werden. Die Abbildung 5 zeigt, dass einige Spalten nicht vollständig gefüllt sind. Die Attribute **`body`**, **`cabin`** und **`boat`** enthalten wenig Information, deshalb werden sie mit **`drop()`** aus den Daten entfernt. Der Zielhafen ist zumindest bei etwa der Hälfte der Personen ausgefüllt. Die restlichen werden mit **`NA`**, also „not available“ gefüllt. Bei der Modellerstellung ist **`NA`** für den Algorithmus ein Hafen, wie jeder andere auch. 1043 Datensätze von insgesamt 1309 bleiben nach dieser Phase der Datenbereinigung noch übrig.

Es folgt ein technischer Schritt (Abbildung 6), in dem die Daten in ein Format transformiert werden, das der Algorithmus verarbeiten kann. Insbesondere werden die beiden für das Geschlecht auftretenden Strings **`Male`** und **`Female`** durch 0 und 1 ersetzt. Außerdem werden die Attribute **`name`**, **`ticket`** und **`home.dest`** aus dem Datensatz entfernt, da sie nicht-kategoriale Werte beinhalten. Hierunter versteht man Daten mit sehr vielen unterschiedlichen Ausprägungen, bei denen nur selten identische Werte austreten. Solche Attribute eignen sich nicht gut für eine Klassifikation und tragen somit wenig zur Qualität des Modells bei.

Die Daten werden entsprechend Abbildung 7 in zwei Teile gesplittet. Die Variable X enthält alle Daten, bis auf die Information, ob ein Passagier überlebt hat. Hierzu wird mittels **`drop(['survived'], axis=1)`** das entsprechende

Attribut aus den Daten entfernt. Die Variable y enthält ausschließlich die aus X entfernten Informationen. Mit **`cross_validation.train_test_split(X,y,test_size=0.2)`** werden die Daten zu 80% in Trainingsdaten und zu 20% in Validierungsdaten aufgeteilt.

Nach doch recht umfangreichen Vorbereitungen erfolgt nun der eigentliche Analyseschritt. In Abbildung 8 wird ein Modell mithilfe des Entscheidungsbaum-Algorithmus erstellt. Es handelt sich um einen sehr leistungsfähigen Klassifikationsalgorithmus aus einer ganzen Reihe verschiedener, in der Literatur beschriebener Ansätze. Detaillierte Informationen zur zugrunde liegenden Mathematik findet sich zum Beispiel im hervorragenden Buch „An Introduction to Statistical Learning“ von James/Witten/Hastie/Tibshirani [4]. In diesem Buch werden auch alternative Algorithmen vorgestellt. Die Dokumentation zur, in diesem Beispiel verwendeten, Bibliothek **`scikit-learn`** [5] enthält ebenfalls Informationen zu weiteren Algorithmen und insbesondere eine Dokumentation der Syntax, mit der sie eingesetzt werden können.

Im vorliegenden Beispiel erreicht der Algorithmus eine Trefferquote von 78,9%. Für einen ersten Versuch ist dieser Wert schon recht gut. Er lässt sich durch Einsatz von komplexeren Algorithmen und insbesondere durch die gezielte Einbeziehung weiterer Attribute noch deutlich verbessern. Der Befehl **`clf_dt.predict(data)`** bietet die Möglichkeit, für einen einzelnen Datensatz **`data`** eine Vorhersage für die wahrscheinlichste Kategorie zu erhalten.

Fazit

Mithilfe der Klassifikation lassen sich aus vorhandenen Daten Modelle erstellen, die in der Lage sind, für neue Datensätze Vorhersagen über die wahrscheinlichste Kategoriezugehörigkeit zu treffen. Die eingesetzten Algorithmen sind mathematisch recht komplex, verdichten sich aber mit geeigneten Python-Bibliotheken auf zwei Zeilen Code. Um gute Ergebnisse zu erzielen, ist eine Menge Vorarbeit in Form von Datenbereinigung und Transformation zu leisten. Hierzu ist spezielles Wissen erforderlich, welches aber nur im engen Zusammenspiel mit eingehender Kenntnis der Daten seine vollen Möglichkeiten entfaltet. Aus dieser Sicht macht es Sinn, bereits früh in einem Projekt eine Fachabteilung und Experten aus dem Bereich Data Mining an einen Tisch zu bringen. Gerne stehen unsere Experten sowohl zur Beratung als auch zur Umsetzung Ihres ersten eigenen Projektes zur Verfügung. Bei Interesse kontaktieren Sie uns gerne telefonisch unter 0 52 51 / 10 63 -0.

Links

[1] Anaconda - Verwaltung von Entwicklungsumgebungen für Python
https://www.continuum.io/downloads

[2] Jupyter Notebook - interaktive Entwicklungsumgebung für Python
http://jupyter.org/

[3] Vanderbilt University USA - Titanic Datensatz
http://biostat.mc.vanderbilt.edu/wiki/pub/Main/DataSets/titanic3.xls

[4] Stanford University USA - The Elements of Statistical Learning
von Trevor Hastie, Robert Tibshirani, Jerome Friedman
http://www-stat.stanford.edu/~tibs/ElemStatLearn/download.html

[5] Python Bibliothek zum Thema Machine Learning
http://scikit-learn.org/stable/

[6] Socialcops - englischsprachiger Blog zu Themen Data Intelligence
https://blog.socialcops.com/engineering/machine-learning-python/

Bildnachweis

Frank Heimerzheim
(info@ordix.de)

Single-Page-Application-Framework

Angular 2

Mit Angular 2 ist im September ein weiteres JavaScript-Framework auf den Markt gekommen. Typensicherheit, Performance und hohe Produktivität: Ist das mit einem JavaScript-Framework überhaupt möglich? Ja – mit Angular 2. Dieser Artikel erklärt die Architektur hinter dem neuem Wunderkind und zeigt die grundlegenden CLI-Befehle, die das Arbeiten mit Angular 2 noch produktiver machen.

Einleitung

AngularJS ist ein von Google entwickeltes JavaScript-Framework, das 2009 zum ersten Mal das Licht der Welt erblickte. Seit kurzer Zeit steht nun der Nachfolger Angular 2 zur Verfügung. Im Gegensatz zu AngularJS setzt die neue Version nicht mehr direkt auf JavaScript, sondern auf TypeScript.

Tools für die Entwicklung

Als Runtime-Umgebung für die Entwicklung mit Angular 2 wird Node.JS [Q1] sowie der dazugehörige Paketmanager npm eingesetzt.

Angular CLI ist das Build-Werkzeug für die Entwicklung von Angular-2-Projekten. Es bietet unterschiedliche Optionen: Von der Erstellung eines Projektes über das Hinzufügen neuer Komponenten bis hin zum Build-Prozess [Q2]. Angular CLI wird über npm durch folgenden Befehl installiert:

```
npm install -g @angular/cli
```

Nach der Installation werden alle Kommandos in Bezug auf den Angular CLI durch das Präfix **ng** eingeleitet. Durch den Befehl

```
ng new <Projekt-Name>
```

wird die Grundstruktur für ein neues Projekt angelegt.

Zur Überprüfung der Installation kann die Anwendung einfach gestartet werden. Im Projektordner wird dazu der folgende Befehl ausgeführt:

```
ng serve
```

Durch diesen Aufruf werden alle Dateien innerhalb des Projektes zu JavaScript kompiliert und die Anwendung unter **`localhost:4200`** bereitgestellt. Eine kurze Einführung in das Setup der Entwicklungsumgebung für Angular wird in [Q6] und [Q7] gegeben.

TypeScript als Grundlage

Die Programmiersprache von Angular 2 ist TypeScript [Q8], eine von Microsoft entwickelte Programmiersprache, die auf JavaScript aufbaut. TypeScript wird beim Kompilieren in JavaScript umgewandelt und ist damit für jeden Browser verständlich. TypeScript ermöglicht Entwicklern, die aus dem Java- oder .Net-Umfeld kommen, einen relativ einfachen Einstieg. Der Aufbau von Klassen ähnelt diesen Sprachen stark.

Die Zugriffsarten **`Public`**, **`Private`** und **`Protected`** werden von TypeScript vollständig unterstützt. Klassen können vererbt und durch Interfaces ergänzt werden – der Name TypeScript legt diese Eigenschaft nahe – und zudem können Variablen mit primitiven Datentypen oder Klassen deklariert werden. Der Compiler meldet bei einer Verletzung des Datentyps einen Compiler-Fehler.

Architektur

Angular 2 versteht sich als modulares Webframework. Alle Bestandteile einer Web-Anwendung sollen in kleine Einheiten aufgeteilt werden, um so eine modulare, verständliche und skalierbare Anwendung zu schaffen.

Während der Erstellung einer neuen Angular-Anwendung mittels Angular CLI werden bereits alle Dateien erstellt, die für einen Best-Practice-Ansatz erforderlich sind. Die generierten Dateien vermitteln bereits einen Eindruck, wie die Bestandteile (siehe Abbildung 1) miteinander agieren.

Der Einstiegspunkt der Webanwendung ist die Datei **`index.html`** (siehe Abbildung 5). Auf den ersten Blick wirkt diese Seite wie eine statische HTML-Seite. Der einzige Unterschied ist der Tag **`app-root`**. Dieser Tag wird zur Laufzeit von Angular durch die passende Komponente aufgelöst. Da in der Klasse **`AppModule`** alle Komponenten der Anwendung im Attribut **`Declarations`** des Decorators **`NgModule`** aufgelistet werden, ist die Suche nach der passenden Komponente

einfach (siehe Abbildung 4). Gesucht wird die Komponente mit dem Attribut Selector **`app-root`** des Decorators Component. Ist diese Komponente gefunden (Abbildung 2), wird sie angezeigt.

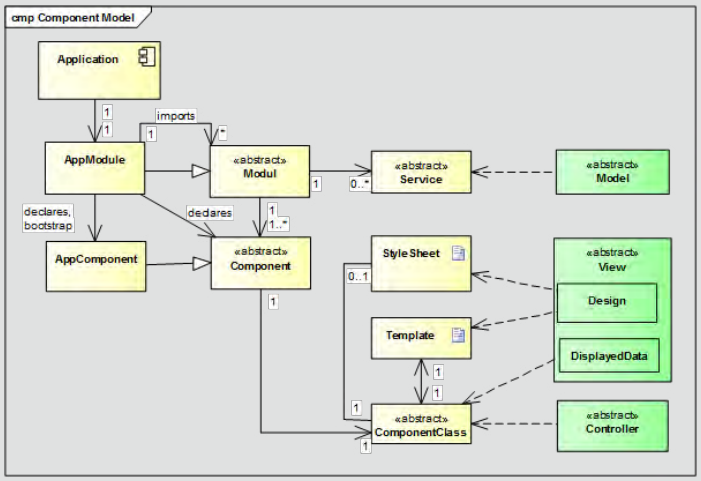


Abb. 1: Die Grundbestandteile von Angular 2

```
import { Component } from '@angular/core';
@Component({
   selector: 'app-root',
   templateUrl: './app.component.html',
   styleUrls: ['./app.component.css']
})
export class AppComponent {
   title = 'app works!';
}
```

Abb. 2: Aufbau einer Komponente (app.component.ts)

```
<h1>
  {{title}}
</h1>
```

Abb. 3: Aufbau eines Templates (app.component.html)

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';

@NgModule({
declarations: [ AppComponent ],
imports: [ BrowserModule ],
providers: [],
bootstrap: [ AppComponent ]
})
export class AppModule { }
```

Abb. 4: Aufbau eines Modules (app.module.ts)

Die Modularisierung von Angular-2-Anwendungen ist in Abbildung 1 dargestellt. Da eine vollständige Erläuterung der Angular-Bestandteile sehr umfangreich wäre, werden hier nur die zwei Hauptbestandteile jeder Anwendung erläutert:

- Komponenten
- Module

Darüber hinaus gibt es noch weitere Bestandteile einer Angular-Anwendung, wie beispielsweise einem Service: Dieser beherbergt die eigentliche Businesslogik der Anwendung, beispielsweise das Konsumieren von Web-Services. Services ermöglichen eine strikte Trennung von Präsentations- und Geschäftslogik.

Komponenten

Beim Anlegen eines neuen Projektes mit Angular CLI wird automatisch eine Komponente in Form der Klasse **`AppComponent`** (Datei **`app.component.ts`**, Abbildung 2), der Template-Datei **`app.component.html`** und einer Style-Sheet-Datei **`app.component.css`** erzeugt.

Der Aufbau der Komponente folgt dem Architektur-Muster Model-View-Controller. Die View wird weitgehend durch das Template dargestellt, der Controller durch die Komponenten-Klasse. Da die Komponenten-Klasse auch die dargestellten Daten enthält, übernimmt sie die Rolle eines View Controllers. Das Model wird per **`Injection`** als Service zur Verfügung gestellt.

Im Template wird das Aussehen der Komponente in Form von HTML definiert (siehe Abbildung 3). Die dazugehörige TypeScript-Datei bereitet die Daten für das Template auf und fungiert als Event-Handler für das Template. In diesem Beispiel wird deutlich, dass die Template-Datei durch Attribute der Klassen-Datei Informationen enthält. In diesem Fall ist es die Anzeige des Titels.

Die TypeScript-Klasse ist das Herzstück einer Komponente: Die Annotation **`@Component`** definiert die Zusammensetzung der Komponente (siehe Abbildung 2). Das Attribut **`Selector`** entspricht dem Tag-Namen im Template (siehe Abbildung 5).

Die hier beschriebene Aufgabenteilung ist richtungsweisend für die Implementierung. Werden in Angular Formulare verarbeitet, so wird die Gestalt in der Template-Datei festgelegt. Die eingegebenen Daten finden sich in der Komponenten-Klasse wieder. In der täglichen Implementierung greift der Entwickler sicher auf das Modul **`FormsModule`** zu – am Architekturmuster einer Komponente ändert sich jedoch nichts.

Module

Dialoge lassen sich in Features unterteilen. Ein solches Feature kann ein Menüpunkt im Hauptmenü der Anwen-

dung sein. Jedes dieser Features benötigt eine Reihe von Komponenten, die in anderen Features nicht benötigt werden. Als Beispiel kann hier ein Formular dienen, das für die Rechnungserstellung benötigt wird. Dieses Formular wird als Komponente innerhalb des Rechnungsmoduls definiert, da es innerhalb eines anderen Moduls, zum Beispiel der Mitarbeiterverwaltung, nicht benötigt wird.

Ein Modul in Angular sollte immer eine abgeschlossene Einheit bilden, dies ermöglicht eine Wiederverwendbarkeit an anderer Stelle, da es keine direkten Abhängigkeiten zu anderen Modulen bzw. Klassen gibt. Jede Angular-Anwendung besteht immer aus mindestens einem Modul, dem **`AppModule`**. Dieses Modul beherbergt den Einstiegspunkt der Anwendung (siehe Abbildung 4).

Ein Modul wird durch die Annotation **`@NgModule`** gekennzeichnet und beinhaltet folgende Elemente:

- **Declarations:**
  Listet alle Komponenten auf, die innerhalb des Moduls benutzt werden.
- **Imports:**
  Hier werden weitere Module importiert, die zusätzliche Features oder Services für die Anwendung bereitstellen.
- **Providers:**
  Deklariert Services, die innerhalb der einzelnen Komponenten über Dependency Injection injiziert werden.
- **Bootstrap:**
  Legt die Start-Komponente der Anwendung fest. In diesem Fall wird die bereits oben genannte **`AppComponent`** aufgerufen.

Werden weitere Module für die Anwendung geplant, müssen sie dem AppModule bekannt gemacht werden, damit dieses das Feature auch ansprechen kann.

Zur Erstellung eines weiteren Moduls benutzen wir wieder das Angular CLI, das analog zum Anlegen von Komponenten auch Module generieren kann. Der Befehl lautet hierfür wie folgt:

```
ng generate module <Modul-Name>
```

Performance für Mobiles

Bei Web-Anwendungen fällt immer das Stichwort „Mobile“. Unter Angular 2 ist dies ebenfalls ein großes Thema, da es hier erhebliche Unterschiede in Bezug auf die Performance geben kann:

Im Regelfall werden von Angular alle Klassen zu einem sehr kleinen Build-Bundle zusammengefasst. Dieses Bundle besteht aus wenigen JavaScript-Dateien, welche je nach Umfang der Anwendung mehrere Megabyte groß sein können. Das ist problematisch, weil in diesem Fall die gesamte Anwendung an den Client geliefert werden muss, bevor die erste Zeile JavaScript ausgeführt wird.

Damit der Client bei schlechter Konnektivität (Smartphone mit Mobilfunk-Anbindung) keine zu langen Wartezeiten in Kauf nehmen muss, stellt Angular 2 zwei Lösungsansätze zur Verfügung, die es ermöglichen, die Wartezeiten auf mobilen Endgeräten zu reduzieren.

Zum einen kann das Lazy-Loading verwendet werden: Die JavaScript-Dateien werden in mehrere kleine Dateien gesplittet und je nach Bedarf an den Client geschickt. Diese Methode ist besonders sinnvoll, wenn die Anwendung aus mehreren logisch voneinander getrennten Anwendungsbereichen besteht.

Das zweite Verfahren verfolgt den Ansatz des serverseitigen Renderns von JavaScript.

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>HelloWorld</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, ini-
tial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root>Loading...</app-root>
</body>
</html>
```

Abb. 5: Aufbau der index.html

Quellen

[Q1] Webseite von Node.JS und npm:
https://nodejs.org/de/

[Q2] Angular CLI:
https://cli.angular.io/

[Q3] ng-Bootstrap:
https://ng-bootstrap.github.io/#/home

[Q4] Angular Material:
https://material.angular.io/

[Q5] Angular Universal:
https://universal.angular.io/

[Q6] ORDIX Blog: Angular 2 – Arbeiten mit Visual Studio Code
https://blog.ordix.de/technologien/angular-arbeiten-mit-visual-studio-code.html

[Q7] ORDIX Blog: Debuggen von Angular 2 mit Visual Studio Code
https://blog.ordix.de/technologien/debuggen-von-angular-2-unter-visual-studio-code.html

[Q8] TypeScript:
https://www.typescriptlang.org/

Glossar

TypeScript
TypeScript ist eine vom Unternehmen Microsoft entwickelte Programmiersprache, die auf den Vorschlägen zum zukünftigen ECMAScript-6-Standard [2] basiert. Sprachkonstrukte von TypeScript, wie Klassen, Interfaces, Vererbung, Module, anonyme Funktionen, Generics und eine statische Typisierung, sollen auch in ECMAScript 6 übernommen werden.

Links

[1] Blogbeitrag im ORDIX Blog:
https://blog.ordix.de/tags/angular-2.html

[2] Produktseite Angular 2:
https://angular.io/docs/ts/latest/

Phillipp Kürsten
(info@ordix.de)

Dieses Verfahren kann verwendet werden, wenn die Unterteilung nicht zum gewünschten Erfolg führt. Für dieses Verfahren ist das Zusatztool Angular Universal notwendig, welches die Angular-Anwendung auf dem Server rendert und an den Client liefert [Q5].

Fazit

Durch die Runtime- und Build-Werkzeuge von Angular 2 ist die Entwicklung sehr effektiv. Der Entwickler kann mühelos ein neues Projekt aufsetzen oder erweitern, in seiner Umgebung das Programm ausprobieren und die Auslieferung vorbereiten. Mit TypeScript erhält der Entwickler eine Programmiersprache, die sich für den professionellen Einsatz auch bei großen Projekten eignet.

Die Architektur mag auf den ersten Blick komplex erscheinen, ist aber nach mehrfacher Anwendung intuitiv einsetzbar. Durch die Implementierung des Model-View-Controllers wird ein bekanntes Muster der Softwareentwicklung aufgegriffen, was die Einstiegshürden weiter senkt.

Mit Angular 2 wurde ein lebender Standard initiiert, der stetig weiterentwickelt wird und in diesem Jahr bereits in die dritte Version übergehen soll. Technologische Sprünge, wie es von AngularJS zu Angular 2 gab, sollen der Vergangenheit angehören: Angular 3 soll kompatibel zu Angular 2 werden.

SCCM Deployment (Teil I):

Automatisiertes Verteilen von Software mittels System Center Configuration Manager

Viele Clients müssen in Unternehmen mit Standardsoftware ausgestattet sein bzw. vorhandene Clients mit neuer Software versorgt werden. Bevor jedoch auf jedem Client eine eigene Installation durchgeführt wird, kann mittels des Microsoft System Center Configuration Manager viel Zeit eingespart werden.

Einleitung

In dieser Artikelreihe wird beschrieben, wie Software mithilfe des Microsoft System Center Configuration Manager (SCCM) unbeaufsichtigt verteilt und installiert werden kann. Im ersten Teil dieser Reihe wird aufgezeigt, wie Software durch den SCCM auf den Zielrechnern installiert wird, hier am Beispiel von Notepad++. Der zweite Teil beschäftigt sich in der nächsten ORDIX® news mit der automatischen Verteilung und Installation einer Oracle-Datenbank auf mehreren Servern. Die Installation und Konfiguration des SCCM wird in den Artikeln nicht beschrieben, allerdings werden die wichtigsten Einstellungen und Konfigurationen für eine Softwareverteilung mittels SCCM-Paketen in diesem Artikel aufgeführt.

Vorbereitungen

Für ein erfolgreiches Verteilen der Software müssen die Installationsmedien zunächst auf einer Freigabe abgelegt und dem SCCM zur Verfügung gestellt werden (Abbildung 1). Dies erfordert, dass das Benutzerkonto des SCCM entsprechende Berechtigungen hat, um auf die Freigabe zugreifen zu können. Innerhalb der späteren SCCM-Konfiguration wird auf die Dateien mittels UNC-Pfad zugegriffen.

SCCM-Konfiguration und Paketbereitstellung

Für die automatisierte Verteilung von Notepad++ werden in diesem Artikel sogenannte SCCM-Pakete (engl. packages) verwendet. Dazu wird im ersten Schritt ein neues Paket erstellt. Wichtige Parameter für das Paket sind:

- Paketname: Install Notepad++ (Abbildung 2)
- Quellordner: \\192.168.200.15\SOURCES\notepad++ (Abbildung 2)
- Programmtyp:Standard-Programm (Abbildung 3)

Nachdem der Typ ausgewählt ist, werden im nächsten Dialogfeld die Spezifikationen für das gewünschte Programm festgelegt.

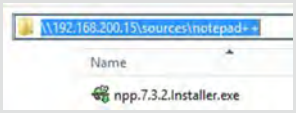


Abb. 1: Freigabepfad der Quelldateien

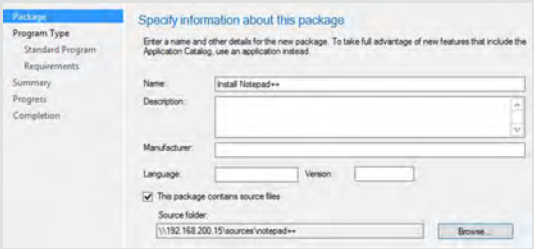


Abb. 2: Paket-Konfiguration 1/3

Abb. 3: Paket-Konfiguration 2/3

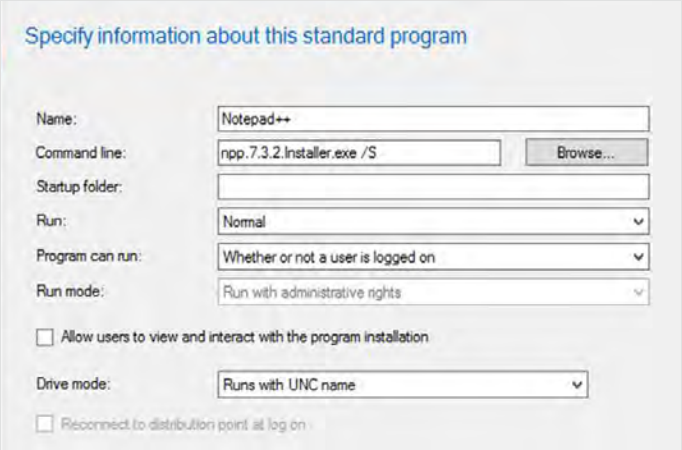


Abb. 4: Paket-Konfiguration 3/3

Abb. 5: Paket erfolgreich auf DP geladen

Abb. 6: Erstellung von Device Collections

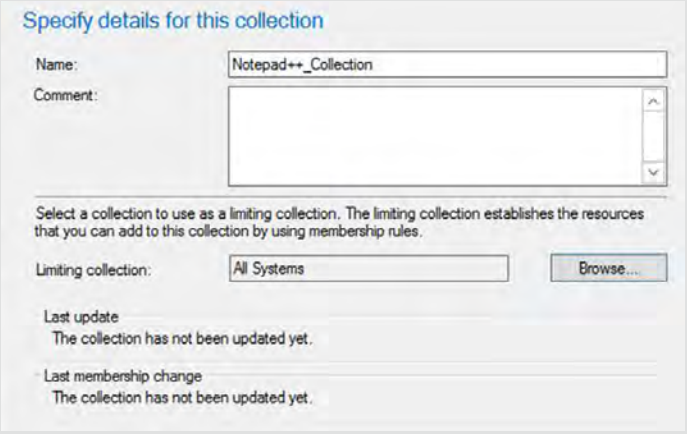


Abb. 7: Konfiguration Device Collection 1/2

Für die Notepad++-Verteilung und -Installation sind die Einstellungen aus Abbildung 4 einzutragen. Wichtig ist der /S -Parameter, da dieser für eine stille Installation von Notepad++ sorgt.

Nach erfolgreicher Konfiguration ist das Paket jetzt mit dem Namen „Install Notepad++“ unter „Application Management->Packages“ erstellt worden.

Bereitstellung des Pakets

Damit ein Paket auf die Zielrechner (Clients) verteilt werden kann, muss dieses zunächst auf einem oder mehreren vorher konfigurierten Distribution Point(s) (DPs) bereitgestellt werden. Hierzu kann über einen Rechtsklick auf das gewünschte Paket und einem Klick auf „Distribute Content“ das Laden auf den DP bzw. die DPs gestartet werden.

Ein erfolgreiches Bereitstellen auf den DP lässt sich anhand des grünen Kreises im unteren rechten Bildschirm feststellen (Abbildung 5). Solange das Bereitstellen noch durgeführt wird, hat der Kreis eine gelbe Farbe. Bei einer fehlerhaften Bereitstellung ist der Kreis rot.

Erstellung einer Gerätesammlung

Bevor das Paket und somit die Notepad++-Installation auf den Clients verteilt wird, sollte eine Gerätesammlung erstellt werden, in der die gewünschten Zielgeräte eingeschränkt werden. Natürlich kann auch die standardmäßig vorhandene „All-Systems“-Gerätesammlung verwendet werden, nur dann würde die Notepad++-Installation auf allen dem SCCM bekannten Geräten verteilt und durchgeführt werden.

Hierzu ist innerhalb der SCCM-Oberfläche ein Wechsel auf „Assets and Compliance“ (unten links) notwendig. Mit einem Rechtsklick auf Device Collection kann jetzt eine neue Collection erstellt werden (Abbildung 6).

Der Gerätesammlung wird ebenfalls ein Name gegeben und anschließend eine Eingrenzung auf einer vorhandenen Gerätesammlung durchgeführt. Hier empfiehlt sich die Standard Collection „All Systems“ (Abbildung 7).

Im nächsten Fenster werden sogenannte Erkennungsregeln erstellt, welche auf die Zielgeräte verweisen. Mit einem Klick auf „Add Rule“ und „Direct Rule“ können nun die gewünschten Geräte auf verschiedene Art und Weise ausgewählt und der Gerätesammlung hinzugefügt werden (Abbildung 8).

In diesem Beispiel ist nur ein Computer mit dem Rechnernamen „Windows7“ per Regel definiert worden. Dieser Computer war innerhalb der Standardsammlung „All Systems“ vorhanden und kann somit ausgewählt und zugeordnet werden. Nachdem die Sammlung erstellt ist, wird eine automatische Überprüfung der enthaltenen Geräte/Computer durchgeführt. Nach kurzer Zeit ist inner-

halb der neuen Gerätesammlung der Rechner WINDOWS7 vorhanden (Abbildung 9).

Natürlich können auch mehrere Computer einer Gerätesammlung per Regel zugeordnet werden, wodurch dann eine automatisierte Verteilung auf mehreren Geräten möglich ist.

Paket-Deployment auf die gewünschte Device Collection

Nachdem die Gerätesammlung erfolgreich erstellt ist, kann jetzt das Deployment auf diese und somit auf den Zielrechner (WINDOWS7) durchgeführt werden. Hierzu wird das gewünschte Paket ausgewählt und im oberen Menü auf den grünen Pfeil (Deploy) geklickt.

Innerhalb des Deployments wird die zuvor erstellte Gerätesammlung ausgewählt. Dadurch wird auf allen Geräten, innerhalb dieser Gerätesammlung das Deployment angewendet (Abbildung 10).

Im nächsten Schritt kann die Bereitstellung auch zeitgesteuert werden. In diesem Beispiel ist über den Button „New“ die Eigenschaft „As soon as possible“ eingestellt (Abbildung 11).

Abschluss

Nach erfolgreichem Deployment ist nun Notepad++ auf dem gewünschten Rechner installiert worden. In diesem Beispiel war nur ein Rechner innerhalb der Gerätesammlung enthalten. Hier können aber auch, wie zuvor erwähnt, mehrere Rechner eingetragen und somit auf allen eine automatisierte und stille Installation durchgeführt werden. In wenigen Konfigurationsschritten kann somit das Installieren von Standardsoftware auf mehreren Zielrechnern automatisiert ablaufen und erspart eine Menge an Arbeitszeit.

In der nächsten Ausgabe erwartet Sie die automatische Installation von Oracle über den SCCM. Hierbei werden wir auch mittels einer stillen Installation von Oracle und einem Paket-Deployment des SCCMs arbeiten.

Sebastian Herd
(info@ordix.de)

Links

[1] Microsoft System Center Configuration Manager
https://www.microsoft.com/en-us/cloud-platform/system-center-configuration-manager

[2] Microsoft Technet „About Distribution Points“
https://technet.microsoft.com/en-us/library/bb681012.aspx

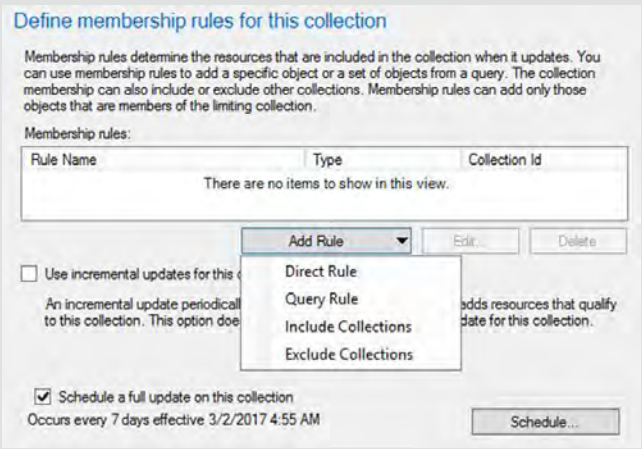


Abb. 8: Konfiguration Device Collection 2/2

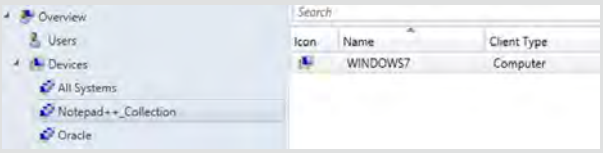


Abb. 9: Neue Device Collection mit Rechner

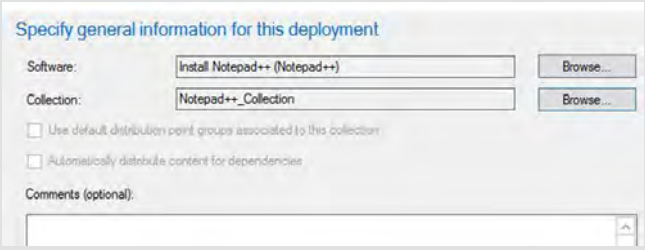


Abb. 10: Paket-Deployment auf Clients 1/2

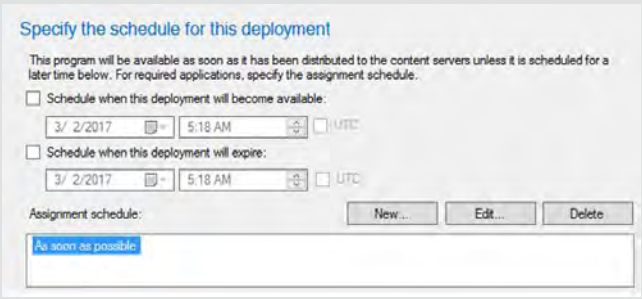


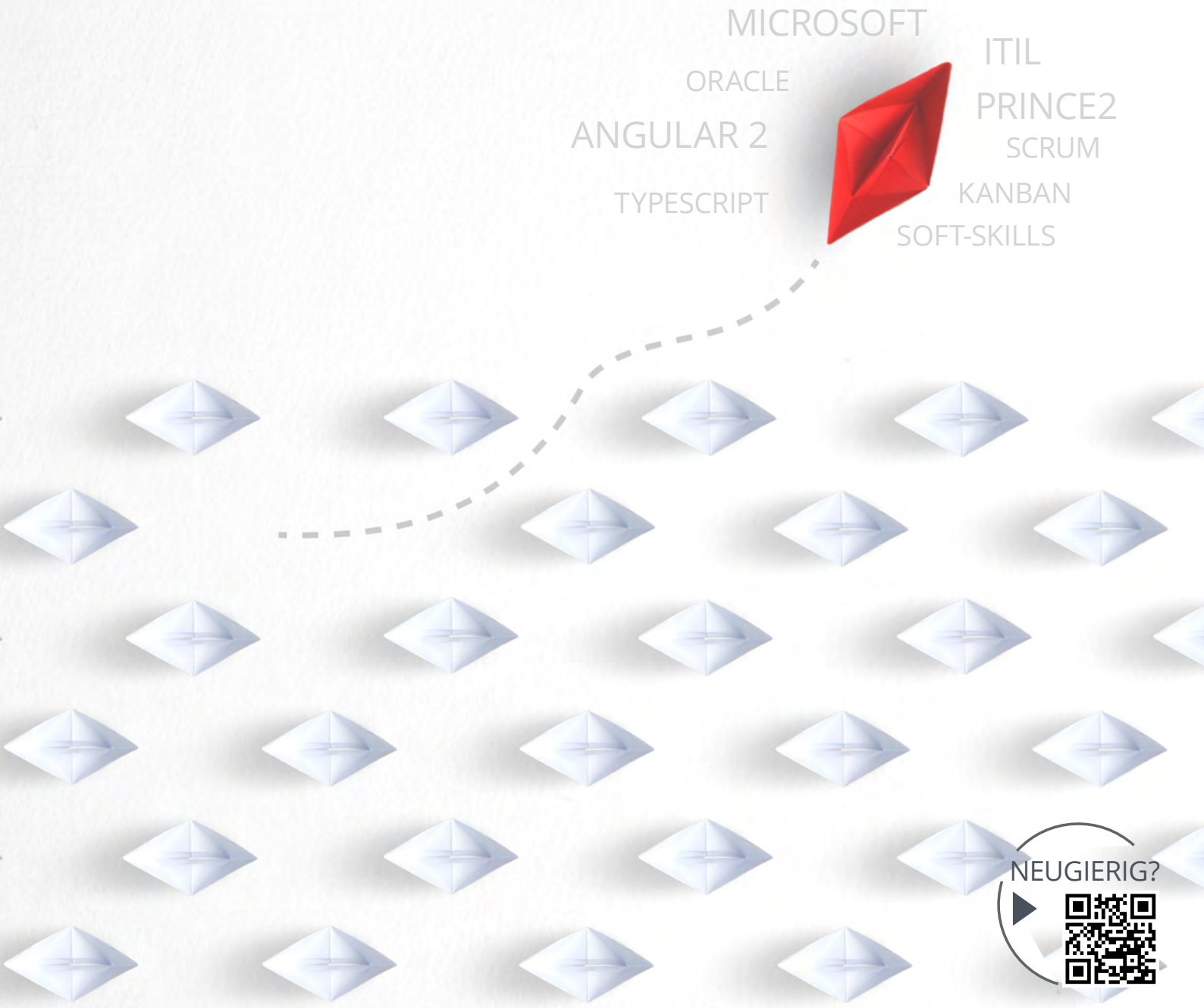
Abb. 11: Paket-Deployment auf Clients 2/2

ORDIX® seminare
einfach. gut. geschult.

SCHWIMMEN SIE MIT DEM STROM?
WIR ENTWICKELN IHR WISSEN!

Vom Entwickler bis zum Abteilungsleiter – wir begleiten Sie, egal welchen beruflichen Werdegang Sie einschlagen.

Mit einer Vielzahl an IT- und Projektmanagement-Seminaren – dabei sind Sie sowohl im technischen als auch im sozialen Umfeld bestens betreut.



NEUGIERIG?

Unser umfangreiches Seminarprogramm finden Sie unter: **seminare.ordix.de**