Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

Abkürzung	Langschreibweise
API	**A**pplication **P**rogramming **I**nterface
AJAX	**A**synchronous **J**avaScript **a**nd **X**ML
DBMS	**D**ata**b**ase **M**anagement **S**ystem
DTD	**D**ocument **T**ype **D**efinition
MySQL	**My S**tructured **Q**uery **L**anguage
PB	**P**reis**b**ock
PHP	**P**ersonal **H**ome **P**age Tools
SGML	**S**tandard **G**eneralized **M**arkup **L**anguage
SQL	**S**tructured **Q**uery **L**anguage
GML	**G**eneralized **M**arkup **L**anguage
WP	**W**ord**p**ress
XML	E**x**tensible **M**arkup **L**anguage

1 Allgemeines zur geplanten Datenmigration

Es wird eine Datenmigration für die Liveshoppingplattform Preisbock[1] vom alten stark erweiterten Wordpresssystem[2] zu dem als neue Grundlage beschlossenen Shopsystem Magento[3] geplant. Während Wordpress in seiner ursprünglichen Version als Blogsystem konzipiert wurde, ist Magento ein seit 2007 für Produktivsysteme einsetzbares Shopsystem. Der Preisbock wurde und wird in allen Versionen von der Social-Commerce-Agentur dotSource[4] entwickelt.
Ziel dieser Arbeit ist die Erarbeitung detaillierter Informationen darüber, wie sich die Daten des momentanen Preisbocksystems in das neue Shopsystem Magento integrieren lassen. Die eigentliche Datenmigration erfolgt mit Hilfe einer speziell entwickelten Importschnittstelle auf XML-Basis. Im Laufe des Imports müssen vorhandene Datenbestände angepasst und validiert werden. Die hierfür notwendigen Metainformationen sowie Funktionen zur Anpassung der Daten sind im Rahmen dieser Arbeit zu erarbeiten.

1.1 Gründe für die Datenmigration

Die momentane Onlineversion der Liveshoppingplattform Preisbock ist auf der openSource-Blogsoftware Wordpress aufgebaut. Diese wurde von der dotSource GmbH um viele Shopfunktionalitäten für einen Liveshop erweitert. Der interne Aufbau der Software Wordpress ist nicht als Shopsystem konzipiert. Somit ist das momentane Produktivsystem des Preisbocks den wachsenden Anforderungen an die Plattform nicht mehr gewachsen.
Durch diese Zweckentfremdung sowie alternierender Entwicklung des Systems litt die allgemeine Datenbasis in ihrer Konsistenz und der Struktur, was sich primär an mehreren IDs für Produkte und Kunden zeigt. Eine Analyse der nötigen

[1] http://www.preisbock.de/

[2] http://wordpress.org/

[3] http://www.magentocommerce.com/

[4] http://www.dotsource.de

Refaktorisierungen führte dazu, dass entschlossen wurde direkt ein neues Shopsystem einzusetzen um den Aufwand zu minimieren.[5]

1.2 Ist- und Soll-Situation

Die Datenbasis des momentanen Produktivsystems des Preisbocks weist in sich viele Inkonsistenzen und eine nicht normalisierte Datenbasis auf. Somit gibt es bspw. zwei Produkttabellen in der Datenbank (product_informaiton, wp_posts), die eine eineindeutige 1:1-Beziehung zueinander aufweisen. Weiterhin sind Datenbestände mancher Spalten in sich inkonsistent. Ein Beispiel hierfür ist die Spalte b_land der Händlertabelle (power_partner). Hierbei soll es sich eigentlich um einen Fremdschlüssel zur Ländertabelle (power_countries) handeln. Dies ist jedoch praktisch nicht möglich, da in dieser Spalte nicht nur Zahlenwerte, sondern auch bezeichnende Strings wie „Germany“ oder „Deutschland“ gespeichert werden. Des Weiteren ist keine vollständige Datenintegrität vorhanden, da Fremdschlüssel in der Tabelle power_customer zur Tabelle wp_posts IDs mit Index 0 aufweisen, die auf der Gegenseite nicht definiert sind.
Diese Datenbasis soll logisch mit Hilfe u.a. von Metainformationen in den soliden Datenbestand der für den Preisbock neuen Plattform Magento eingepflegt werden. Der Import wird aufgrund der großen Datenmenge weitestgehend automatisiert erfolgen. Die Daten werden aus dem alten Preisbock-System ausgelesen und in die Modelstrukturen des Magentosystems (siehe 2.3) eingepflegt, dort validiert und final gespeichert. Dabei ist es notwendig möglichst jederzeit einen Rückschluss auf die Datensätze im Altsystem ziehen zu können.

Abbildung 1: Logo Wordpress DE

Abbildung 2: Logo Preisbock

Abbildung 3: Logo Magento

[5] Vgl. [GFo08] S. 26

2 technologische Erläuterungen

2.1 Extensible Markup Language (XML)

Die Extensible Markup Language (XML) ist eine Erweiterung der Standard Generalized Markup Language (SGML), welche wiederum auf der allgemeingültigen Generalized Markup Language (GML) basiert. Die Sprache XML wurde in der momentanen Version 1.0[6] vom World Wide Web Consortium (W3C)[7] am 16. August 2006 definiert. Die fünfte Überarbeitung vom 26. November 2008 stellt den momentan aktuellen Standard dar. In dieser Version definiert das W3C eine Metasprache mit welcher durch strukturelle und inhaltliche Einschränkungen eine anwendungsspezifische Sprache entwickelt werden kann. Somit stellt XML eine Auszeichnungssprache für allgemeine Dokumente dar. In dieser Sprache können Daten in frei benannten Tags gespeichert werden. Jedem Tag können beliebig Attribute hinzugefügt werden, welche die Inhalte der Tags semantisch näher beschreiben sollen. Einschränkungen in Struktur und Inhalt eines XML-Dokumentes werden mit Hilfe von Document Type Definition (DTD) oder XML-Schema-Dateien definiert.
Die für die Magento-Version des Preisbocks geschaffene Importschnittstelle verwendet XML-Dokumente zur logischen Abbildung der Datenbankstruktur des momentanen Preisbocksystems. Weiterhin werden die für den Import notwendigen Einstellungen in speziellen XML-Dateien abgebildet. Zur Überprüfung der korrekten Syntax stehen XML-Schemata zur Verfügung.

Abbildung 4: Logo World Wide Web Consortium

[6] http://www.w3.org/TR/xml/
[7] http://www.w3.org/

2.2 Datenbanksichten

Eine Sicht – auch View genannt – ermöglicht die Speicherung einer bestimmten Anfrage, welche dadurch wieder verwendet werden kann. Eine View stellt also einen Bereich dar, deren Inhalt sich aus den Datenbeständen verschiedener Tabellen zusammensetzt und zum Zeitpunkt einer Anfrage jeweils neu berechnet wird. Durch dieses Konzept wird die logische Datenunabhängigkeit unterstützt. Des Weiteren kann dadurch die Anfrageformulierung enorm vereinfacht werden.
Beim Import der Preisbockdaten werden Views benötigt, damit Daten nicht normalisierter Tabellen geglättet sowie nach Bedingungen zusammengeführt werden können oder allein dafür, Nutzdaten von Fremddaten zu separieren. Somit können die Daten für einen vereinfachten Import aufbereitet werden. Durch die bessere Zwischenspeicherung des Ergebnisses im Datenbankmanagementsystem (DBMS) als bei einer normalen Abfrage, ist auch ein Performanceanstieg zu beobachten. In einer Sicht ist nur begrenzt eine Aktualisierung oder Löschung der Daten möglich, da sich die Daten aus mehreren meist nicht vollständigen Tabellen zusammensetzen und somit durch die Logik der View eine Änderung der eigentlichen Ausgangsdaten nicht möglich ist. In SQL wird eine Sicht definiert durch den Befehl CREATE VIEW <Sichtname> AS <SELECT-Statement>. Über den Sichtnamen kann wie über eine normale Tabelle von der Anwendung heraus auf die generierten Daten lesend zugegriffen werden.

2.3 Modell-View-Controller (MVC)

Das Entwurfsmuster des Model-View-Controller (MVC – Modell / Präsentation / Steuerung) teilt die Struktur einer Software in drei logische Bereiche. Somit soll die Entwicklung der Anwendung flexibel erfolgen und eine bessere Wiederverwendbarkeit einzelner Codesegmente gefördert werden. Das Modell enthält alle relevanten Daten der Anwendung und ist unabhängig von dem Controller und der Anzeigeebene. Es stellt quasi den Zugriff auf alle wichtigen Daten her und verwaltet diese.

Der Controller ist das Bindeglied zwischen der Präsentationsschicht und dem Modell. Er nimmt Benutzeraktionen entgegen und werte diese entsprechend aus. Der Controller veranlasst das Schreiben aller Daten im Modell, sodass später in der View wieder lesend darauf zugegriffen werden kann.
In der View als Präsensschicht der Anwendung erfolgt eine Ausgabe generierter Daten aus dem Modell. Diese Schicht greift also lesend auf das Modell zu. Meist sind so genannte Templates Bestandteil dieser Softwareschicht. Diese stellen eine fertige Ausgabe mit Platzhaltern da, die dann in der View mit Daten gefüllt und an den Nutzer ausgegeben werden.

3 Vorgang des Imports ins neue System

3.1 Allgemeine Möglichkeiten beim Import

Für den Import wurde eine spezielle Schnittstelle geschaffen, die die Datenmigration der Preisbockdaten in das Magentosystem automatisiert ermöglichen soll. Bei diesem Import stehen einige Funktionalitäten bereit, damit importierte Datenbestände an das Magentosystem angepasst werden können. Diese lassen sich mit Hilfe einer XML-Datei leicht konfigurieren. Da Magento dem Design-Pattern des Model-View-Controller-Prinzips folgt, muss jedem Importprofil genau ein Model zugewiesen werden (bspw. customer/customer für Kunden), da die Daten in das Model importiert und von dort aus gespeichert werden. Mit Hilfe einer Helperklasse im Magento können dem Projekt zusätzliche Funktionalitäten zur Verfügung gestellt werden. Diese Funktionalitäten können zusätzlich sowohl vor, als auch nach dem Import der Datensätze aufgerufen werden, dienen aber primär als Filter für die komplexe Bearbeitung der zu importierenden Attribute. Des Weiteren existiert die Möglichkeit, dass Werte der importierten Felder umgeschrieben werden. Dies kann mittels einfachen Suchens und Ersetzens, aber auch mittels komplexer regulärer Ausdrücke umgesetzt werden. Zudem ist ein Dublettenfinder implementiert, der im Kopfbereich der XML konfiguriert werden kann.

3.2 Abhängigkeiten

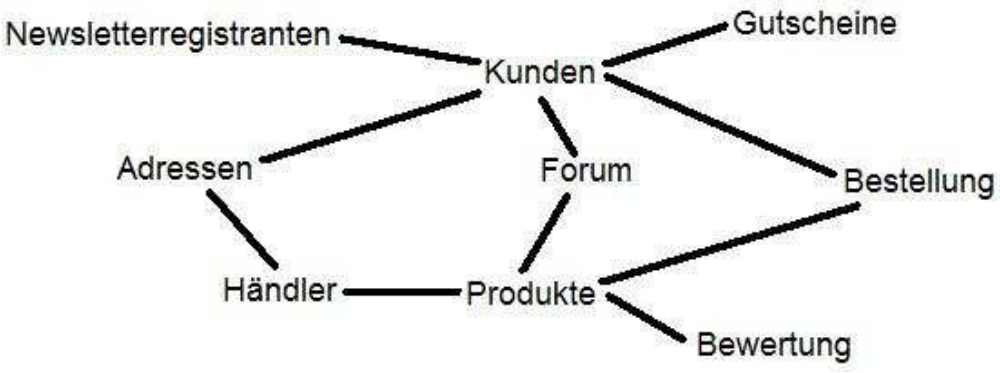

Abbildung 5: Abhängigkeiten importierter Elemente

Für einen erfolgreichen Import sind mehrere kleinere Teilimporte notwendig, da einzelne Objekte logisch voneinander abhängig und an die Models gekoppelt sind. Des Weiteren kann sich dies als notwendig erweisen, wenn sich beim Import eines Objektes auf mehrere Model im Magento bezogen werden muss (bspw. bei Produktinformationen Preis etc. separat). Die einzelnen Teilimporte müssen durch diese Abhängigkeit teilweise in einer bestimmten Reihenfolge erfolgen. Eine praktisch mögliche Reihenfolge wird folgend gegeben:

Reihenfolge	Beschreibung	Model	abhängig
1	Kunden, Nutzer	customer/customer	
2	Kundenadressen	customer/address	1
3	Newsletterregistrierungen	newsletter/subscriber	1
4	Händler	customer/customer	
5	Händleradressen	customer/address	4
6	Produkte	catalog/product	1, 4
7	Gutscheine	dotsource/spusercoupon	1
8	Bestellungen	sales/order	1, 6
9	Bewertungen	dotsource/scrating	1, 6
10	Forum	dotsource/scforum	1, 6

Tabelle 1: Reihenfolge der Teilimporte und ihre Abhängigkeiten

3.3 Generelle Informationen zur Logik der Datenbankstruktur

Damit die geschaffene Importschnittstelle im Magento vollautomatisiert die Daten aus dem Fremdsystem gewinnen kann, benötigt es Informationen über die logische interne Datenbankstruktur. Da bei der momentanen Version des Preisbocks keine Fremdschlüssel oder ähnliches existieren, müssen logisch gleiche Attribute speziell definiert werden. Diese werden unter einem gemeinsamen, eindeutigen Namen als Gruppe zusammengefasst.

Gruppenname	Gleichartige Attribute
wp_customer_id	wp_users.ID power_customer.wp_id power_allocate.wp_id remind_activation.wp_id remind_login-wp_id wp_forum_posts.author_id wp_forum_threads.starter
kkid	power_customer.kkid power_customer_address.kkid power_allocate.kkid
wp_post_id	product_information.aid wp-posts.ID processbar_start.aid
product_id	product_information.pid power_allocate.pid product_newsletter.pid
address_id	power_customer_address.ID
hid	power_partner.hid product_information.hid
order_id	power_allocate.vid Survey_votes.vid
newsletter_subscriber	wp_newsletter.ID
pid	gfo_product_simple.pid
wp_product_id	gfo_product_simple.ID
variation_id	gfo_product_simple.variation_id
forum_forum	wp_forum_forums.id wp_forum_threads.forum_id
forum_thread	wp_forum_threads.id wp_forum_posts.thread_id
forum_post	wp_forum_posts.id

Tabelle 2: Gruppierung gleichartiger Attribute

3.4 zu migrierende Komponenten

Damit der Importvorgang aus dem Preisbocksystem in das neue Shopsystem Magento Magento vonstattengehen kann, wurde eine spezielle XML basierte Pluginschnittstelle entwickelt, auf die in dieser Arbeit nicht näher eingegangen werden soll. Die für einen Import erforderlichen Informationen werden in der Importschnittstelle aus speziell konfigurierten XML-Dateien entnommen. Dabei gibt es eine allgemeingültige XML-Datei, die den Grundaufbau der Datenbank des Fremdsystems mit seinen Abhängigkeiten logisch abbildet und zudem wird für jedes zu importierende Model eine spezielle XML-Datei mit näheren Metainformationen zum Import benötigt. Der Start des jeweiligen Imports ist durch einfachen Klick im Backend realisierbar. Dabei muss auf die logischen Abhängigkeiten der Datensätze geachtet werden. Damit das System vollautomatisiert die Abhängigkeiten der verschiedenen Datensätze mehrerer importierter Objekte auflösen kann, ist es nötig, wichtige Schlüsselwerte wie bspw. Primärschlüssel und Fremdschlüssel für diesen Vorgang vorzumerken. Wird dies getan, so werden Mappinginformationen von den alten ID zu den im Magento neu vergebenen Primärschlüsseln in den Metainformationen der Importschnittstelle abgelegt. Dieser Vorgang ist auch mit mehreren verschiedenartigen IDs für ein Model möglich (bspw. Kunde und Händler). Über ein speziell definiertes XML-Konstrukt kann folgend bei weiteren Importen anderer Entitäten auf dieses Mapping zugegriffen werden. Weiterhin werden beim Importvorgang die Daten validiert und ihre Konsistenz im Sinne der Konfigurationen in den XML-Dateien bereinigt.
Damit Zeit bei der Entwicklung eingespart werden kann, werden nicht alle Elemente aus der Preisbockversion des Preisbocks importiert. Die Bestandsdaten werden dann über das alte System weiter gepflegt.

3.4.1 Kunden

Die Informationen über den Kundenstamm sind im momentanen Preisbocksystem weitgehend über die Tabellen power_customer und wp_users verteilt. Hierbei ist zu beachten, dass beide Tabellen einen eigenen Primärschlüssel aufweisen. Somit gibt

es keine einheitliche Kundennummer im System und es müssen beide Primärschlüssel für ein späteres erfolgreiches Mapping in den Metainformationen der Importschnittstelle den neuen Magento-IDs zugeordnet werden. Diese Informationen sind in den Spalten power_customer.kkid sowie wp_users.ID im Preisbocksystem gespeichert. Weiterhin ist darauf zu achten, dass mehrfach vorkommende E-Mail-Adressen im Preisbock-System möglich, bei Magento jedoch verboten sind. Datensätze, die diesen Kriterien entsprechen, müssen notiert und im Anschluss manuell bearbeitet werden. Weiterhin muss vor dem Import geprüft werden ob alle Fremdschlüssel zwischen den beiden Tabellen korrekt vorhanden sind.
Für den vollständigen Import der Kundendaten sind Attribute für das XML-Response der Gefährlichkeitsprüfung sowie die manuelle Verifikation der Volljährigkeit der Installationsroutine des Preisbockprojekts im Magento hinzuzufügen. Das Attribut XML-Response wird von der Komponente SpWirecard eingerichtet, welches somit eine Grundlage für den Import eines Kunden darstellt.
Bevor ein Import in das neue System stattfinden kann, muss die Konsistenz der Kundentabelle im alten System wieder hergestellt werden. So gibt es im Attribut power_customer.wp_id momentan 16 Datensätze, die hier die falsche ID 0 aufweisen.

Model:	customer/customer
Notwendige IDs für spätere Imports:	power_customer.kkid, wp_users.id

Tabelle 3: Allgemeine Definitionen für Kunden

Attribut	Quelle	Weitere Definitionen
email	power_customer.email	
nickname	wp_users.user_login	
prefix	power_customer.anrede	
firstname	power_customer.firstname	
lastname	power_customer.lastname	
password_hash	wp_users.user_pass	
website_id	-	getWebsiteId(): gibt ID der momentanen Shopseite zurück
created_at	wp_users.user_registered	

dob	power_customer.birthday	
is_active	wp_users.user_status	
gender	power_customer.anrede	Einfaches Umschreiben: • Herr → 1 • Frau → 2

Tabelle 4: Spezielle Definitionen für Kundenimport

3.4.2 Adressen

Die Adressen im wordpressbasierten Preisbocksystem sind über die Tabellen power_customer und power_customer_address verteilt. Während es in der Tabelle power_customer Spalten für die Speicherung von Liefer- und Rechnungsadressen gibt, werden in der power_customer_address meist nur Rechnungsadressen gespeichert, wobei aber die Unterscheidung von Liefer- und Rechnungsadressen anhand eines Flags möglich ist. Diese Situation ist durch den Umbau des Preisbocks entstanden, der erlaubt, dass mehrere Adressen möglich sind.
Um die Daten vollständig von der alten Version in die Magentoversion mit Hilfe der Importschnittstelle zu übertragen, bedarf es einer MySQL-View, die den eigentlichen Selektionsvorgang während des automatisierten Imports vereinfacht und eine Sicht schafft, bei der nur Adressen ohne behindernde Fremddaten vorhanden sind. Diese wird unter dem Namen gfo_customer_addresses definiert.

```
CREATE VIEW gfo_customer_addresses AS
SELECT pca.kkid, pca.title, pca.firstname, pca.lastname, pca.company, pca.street,
pca.city, pca.postcode, pca.country, pca.is_shipping_address,pca.is_billing_address
  FROM power_customer_address pca
UNION SELECT pc.kkid, pc.titel, pc.firstname, pc.lastname, pc.b_company,
pc.b_address, pc.b_city, pc.b_postcode, pc.b_land, 1,0
  FROM power_customer pc
  WHERE pc.b_address IS NOT NULL AND pc.b_address != ''
```

```
UNION SELECT pc.kkid, NULL, pc.rechnung_firstname, pc.rechnung_lastname,
pc.rechnung_company, pc.rechnung_street, pc.rechnung_city,
pc.rechnung_postcode, pc.b_land, 0, 1
  FROM power_customer pc
  WHERE rechnung_street IS NOT NULL AND rechnung_street != '' AND
(rechnung_name IS NULL OR rechnung_name = '')
UNION SELECT pc.kkid, NULL, SUBSTRING(pc.rechnung_name, LOCATE(' ',
pc.rechnung_name) + 1) AS firstname, SUBSTRING(pc.rechnung_name, 1,
LOCATE(' ', pc.rechnung_name)) AS lastname, pc.rechnung_company,
pc.rechnung_street, pc.rechnung_city, pc.rechnung_postcode, pc.b_land, 0, 1
  FROM power_customer pc
  WHERE rechnung_street IS NOT NULL AND rechnung_street != '' AND
    LOCATE(' ', pc.rechnung_name) > 1
```

Abbildung 6: SQL-View für Adressimport

Des Weiteren sind die Landangaben in der Tabelle power_customer in sich nicht konsistent und manchmal nicht vorhanden. Die Länderangaben werden mittels einer kleinen Funktion, die diese nach ISO 3166 A2[8] mit Deutschland als Standardland zurückgibt, bereinigt. Länderangaben im Magento werden nach diesem Standard gespeichert.

Model:	customer/address
Notwendige IDs für spätere Imports:	<keine einh. ID durch View vorhanden>

Tabelle 5: Allgemeine Definitionen für Adressen

Attribut	Quelle	Weitere Definitionen
parent_id	gfo_customer_addresses.kkid	ID aus customer/customer
city	gfo_customer_addresses.city	
company	gfo_customer_addresses.company	
country_id	gfo_customer_addresses.country	getCountryId(): gibt ISO 3166 des Landes zurück

[8] http://userpage.chemie.fu-berlin.de/diverse/doc/ISO_3166.html

firstname	gfo_customer_addresses.firstname	
lastname	gfo_customer_addresses.lastname	
postcode	gfo_customer_addresses.postcode	
street	gfo_customer_addresses.street	

Tabelle 6: Spezielle Definitionen für Adressimport

3.4.3 Newsletterregistrierungen

Jeder Besucher kann einen regelmäßig erscheinenden Newsletter abbonieren. Wird der Newsletter von einem schon registrierten Kunden abboniert, so wird dessen Kunden-ID bei den Daten der Newsletterregistrierung gespeichert. Bei dem Prozess der Registrierung zum Newsletter wird nach den Double-Opt-In-Verfahren vorgegangen. Dabei wird dem potentiellen Abbonenten eine E-Mail an die angegebene Adresse geschickt, die einen Bestätigungslink mit einem zufällig generierten Hash enthält. Somit wird verhindert, dass jemand für jemanden anderen einen Newsletter abboniert, auch wenn dadurch die Rate der erfolgreichen Abbonements leicht abnimmt. Magento unterscheidet zwischen drei Status bei der Zusendung der Newsletter:

1. SUBSCRIBED → erfolgreiches Abonnement ➔ Newsletter wird geschickt
2. NOT_ACTIVE → Double-Opt-In nicht abgeschlossen ➔ kein Newsletter
3. UNSUBSCRIBED → vom Newsletter abgemeldet ➔ kein Newsletter

Model:	newsletter/subscriber
Notwendige IDs für spätere Imports:	newsletter_subscriber

Tabelle 7: Allgemeine Definitionen für Newsletterregistranten

Attribut	Quelle	Weitere Definitionen
store_id	-	getStoreId(): Gibt ShopID Preisbock zurück
change_status_at	wp_newsletter.regist	timeFromDate(): Wandelt Date in Datetime um
customer_id	wp_newsletter.email	getCustomerIdByMail():

		Sucht KundenID anhand E-Mail
subscriber_email	wp_newsletter.email	
subscriber_status	wp_newsletter.unregist	setNewsletterStatus(): Newsletterstatus ermitteln
subscriber_confirm_code	wp_newsletter. validate_hash	

Tabelle 8: Spezielle Definitionen für Newsletterregistrantenimport

3.4.4 Händler

Händler sind in der Wordpressvariante des Preisbocks in der Tabelle power_partner gespeichert. Im neuen Magentosystem sollen die Händler als Kunden mit speziell erweiterten Attributen angelegt werden. Des Weiteren bilden alle Händler zusammen die Nutzergruppe External Dealer. Dafür ist die Installation der Komponente SpExternalDealer notwendig. Da Händler in der neuen Version erweiterte Kunden sind, müssen sie sowohl eine E-Mail-Adresse, als auch einen Benutzernamen und ein Passwort aufweisen. Dies ist in der Wordpressvariante des Preisbocks nicht der Fall und muss bei einem Import automatisch generiert werden. Der Benutzername entspricht dabei dem Nutzernamen der E-Mailadresse, als Passwort ist ein einheitliches Passwort zu nehmen, welches bei Bedarf für den Händler im Backend des Magento geändert werden kann. Eine Mailadresse muss somit vorher manuell in den Datenbestand des Preisbocks integriert werden.
Adressen der Händler müssen als separater Import behandelt werden, da diese nicht dem Kunden, sondern dem Adressmodel zugewiesen werden. Ein entsprechendes korrektes Mapping zwischen Händler und Adressen – also zwischen den Entitäten Customer und Adresses – muss umgesetzt werden. Weiterhin bedarf es besonders bei den Adressen der Händler einer Bereinigung der Adressinformationen bspw. beim Händlerland.

Model:	customer/customer
Notwendige IDs für spätere Imports:	hid

Tabelle 9: Allgemeine Definitionen für Händler

Attribut	Quelle	Weitere Definitionen
email	power_partner.email	
nickname	power_partner.companyname	createUsernameFromInput(): valide Nutzername aus Firmennamen generieren
firstname	power_partner.firstname	
lastname	power_partner.lastname	
password_hash	-	createVendorPassword(): Händlerpasswort generieren
website_id	-	getWebsiteId(): gibt Id der momentanen Shopseite
disabled	power_partner.verified	Einfaches Umschreiben: • 0 → 1 • 1 → 0
verified	power_partner.verified	
comment	power_partner.comment	
general_terms	power_partner.tac	
depositor	power_partner.depositor	
account_number	power_partner.account_number	
bank_code	power_partner.bank_code	
bank	power_partner.bank_name	
website	power_partner.internet	Umschreiben nach RegEx: • ^(http://)?(.+) → http://$2
mobile	power_partner.mobil	
group_id	-	getExternalDealerGroupId(): Gruppennummer Händler

Tabelle 10: Spezielle Definitionen für Händlerimport

3.4.5 Produkte

Im Magento gibt es mehrere Produkttypen. Die Typen Singleprodukt und Configurable Product werden vom Preisbock benötigt. Dabei sind alle einfachen Produkte und die einzelnen Variationen an sich als Singleprodukte zu importieren. Mehrere Singleprodukte werden folgend als ein Configurable Product als logisches Verkaufsprodukt mit Variation zusammengefügt.
Beim Import der Produkte in den Preisbock muss sehr viel über individuelle Funktionen nachbearbeitet werden. Dies kommt daher, dass die Daten des Preisbocks in die nicht dafür vorgesehenen Funktionalitäten und Tabellenstrukturen des Wordpress übernommen wurden. Diese wurde um eine zusätzliche Produkttabelle des Preisbocks erweitert. Aufgrund der Tatsache, dass das Produkt eines der wichtigsten Objekte in dem Onlineshopsystem Preisbock ist, weist es viele Abhängigkeiten mit verschiedenen anderen Objekten im Onlineshop auf (bspw. Nutzer, der es anlegt, Lieferant, Bilder, ID für den Logistiker TVA, etc.). Weiterhin sind viele Informationen gemeinsam in der Spalte wp_posts.post_content gespeichert und müssen für das neue System separiert werden. Dies sind unter anderem die allg. Produktbeschreibung, die technische Beschreibung, der Störer, der MP3-Song, die Variationsbilder, sowie Videos. Hierbei sollen in eigenen Filterfunktionen für die Importschnittstelle die Preisbockfunktion cut_content( ) sowie reguläre Ausdrücke zum Einsatz kommen.
Da aufgrund der Struktur des Wordpress in der Produkttabelle wp_posts sowohl Produkte, aber auch reine Content-Seiten und Anhänge zu den Produkten gespeichert werden, ist es notwendig eine SQL-View über die Produkttabellen zu erstellen, bei der Fremdinformationen aus der Tabelle wp_posts herausgefiltert werden. Gleiches ist auch bei dem Import des Bildmaterials inklusive Zuweisung zum Produkt notwendig.

```
CREATE VIEW gfo_product_simple AS
SELECT
wp.ID,wp.post_author,wp.post_date,wp.post_content,wp.post_title,wp.post_excerpt,
wp.post_name,wp.post_modified,wp.guid,pi.pid,pi.tva_id,pi.hid,pi.price,pi.uvp,pi.disp
osable,pi.tax,pi.cooperate_persale,pi.cooperate_percent,pi.cooperate_absolut,pi.max
```

```
orderamount,pi.shipping_days,pi.purchase_price,pi.subvention,pi.other_revenue,pi.tv
a_parcelway,pi.payremind,pv.id AS variation_id,pv.name AS variation_name,pv.price
AS variation_price,pv.disposable AS variation_disposable,pva.name AS
variationgroup ,ps.step2percent
FROM wp_posts wp
INNER JOIN product_information pi ON wp.ID = pi.aid
LEFT JOIN product_variation pv ON pi.pid = pv.pid
LEFT JOIN product_variation_name pva ON pi.pid = pva.pid
INNER JOIN processbar_start ps ON wp.ID = ps.aid
WHERE wp.post_type='post'
```

Abbildung 7: SQL-View für Produkte

Da im Standard-Magento nicht alle Attribute eines Produktes, die beim Preisbock benötigt werden, vorhanden sind, müssen diese implementiert werden, damit sie importiert werden können. Des Weiteren speichert Magento Produkte nicht in einem einzelnen Model ab, wie das beispielsweise bei Adressen oder Kunden der Fall ist, sondern verwendet für Preis oder vorhandene Stückzahl ein Untermodel namens StockItem. Somit muss der Import geteilt werden und es muss chronologisch ein Import Erfolgen von

1. Produkthüllen als Simple Produkte
2. Nähere Informationen zu den Produkten als davon abhängige Stock Items
3. Zusammenfassung von Simple Produkts in Configurable Produkts
4. Import des Bildmaterials und der Attachements

Model:	catalog/product
Notwendige IDs für spätere Imports:	pid, wp_product_id, variation_id

Tabelle 11: Allgemeine Definitionen für Produkte

Attribut	Quelle	Weitere Definitionen
entity_type_id	-	getProductEntityTypeId(): Typengruppe der Produkte
attribute_set_id	-	getGeneralAttributeSet(): Attributgruppe für Produkte
type_id	gfo_product_simple.type_id	
sku	gfo_product_simple.pid	createSku(): Eindeutige ID aus alter Produktnr.
name	gfo_product_simple.post_title	
url_key	gfo_product_simple.post_name	
weight	gfo_product_simple.weight	
price	gfo_product_simple.price	
short_description	gfo_product_simple.post_excerpt	
description	gfo_product_simple.post_content	stripContentNotDescription() Filtert nur reine Beschreibung
Spezielle Preisbockattribute		
delivery_date	gfo_product_simple.shipping_days	calculateDeliveryDate(): Lieferdatum berechnen
payment_remind_date	gfo_product_simple.payremind	
recommended_retail_price	gfo_product_simple.uvp	

label	gfo_product_simple.post_content	seperateLabel(): Störer extrahieren
purchase_price	gfo_product_simple.purchase_price	
dealer_subvention	gfo_product_simple.subvention	
other_earnings	gfo_product_simple.other_revenue	
product_description	gfo_product_simple.post_content	seperateProductDescription(): Separatisiert Beschreibung
technical_description	gfo_product_simple.post_content	seperateTechnicalDescription() Technische Beschreibung
externaldealer	gfo_product_simple.hid	ID aus customer/customer
progressbarstate	gfo_product_simple.step2percent	

Tabelle 12: Spezielle Definitionen für Produktimport

3.4.6 Forum

Das Forum ist ähnlich aufgebaut wie die Version des Wordpress-Preisbocks und kann somit einfach mit der Hilfe von mehreren Importen übertragen werden. Bevor das Forum importiert werden kann, müssen sowohl Kunden als auch Produkte importiert worden sein, da sowohl Threads als auch Posts von diesen abhängig sind bzw. sein können. Weiterhin müssen durch die Abhängigkeiten zwischen Forum, Thread und Post die Daten in einer festgeschriebenen Reihenfolge importiert werden:

1. Forum
2. Thread
3. Post

Die Beziehung zum Produkt rührt daher, dass bestimmten Threads mit ihren Posts als Kommunikationsplattform zu einem speziellen Produkt dienen. Moderatoren des Forums werden nicht importiert, da sie leicht manuell über das Backend eingestellt werden können.

Model:	scforum/forum
Notwendige IDs für spätere Imports:	forum_forum

Tabelle 13: Allgemeine Definitionen für Foren

Attribut	**Quelle**	**Weitere Definitionen**
forum_name	wp_forum_forums.nicename	
forum_desc	wp_forum_forums.name	
forum_sort	wp_forum_forums.sort	
forum_type	-	getForumType(): Produktforum oder Normalforum

Tabelle 14: Spezielle Informationen für Forumsimport

Model:	scforum/topic
Notwendige IDs für spätere Imports:	forum_thread

Tabelle 15: Allgemeine Definitionen für Threads

Attribut	Quelle	Weitere Definitionen
forum_id	wp_forum_threads.forum_id	ID aus scforum/forum
topic_sticky	wp_forum_threads.nicename	
topic_title	wp_forum_threads.subject	
topic_poster_id	wp_forum_threads.starter	ID aus customer/customer
topic_poster_name	wp_forum_threads.starter	getCustomerIdByNick(): sucht NutzerId anhand Namen
topic_poster_email	wp_forum_threads.starter	getCustomerIdByMail(): Sucht KundenID anhand E-Mail
topic_time	wp_forum_threads.date	
topic_views	wp_forum_threads.views	

Tabelle 16: Spezielle Informationen für Threadimport

.

Model:	scforum/post
Notwendige IDs für spätere Imports:	forum_thread

Tabelle 17: Allgemeine Definitionen für Posts

Attribut	Quelle	Weitere Definitionen
topic_id	wp_forum_posts.thread_id	ID aus scforum/topic
forum_id	wp_forum_posts.thread_id	getForumByTopic(): Sucht ForumID anhand ThreadID
poster_id	wp_forum_posts.author_id	ID aus customer/customer
poster_ip	wp_forum_posts.IP	
poster_name	wp_forum_posts.author_id	getCustomerNickById(): sucht Nutzernamen anhand ID
poster_email	wp_forum_posts.author_id	getCustomerMailById(): Sucht E-Mail anhand ID
post_time	wp_forum_posts.date	
post_approved	wp_forum_posts.approved	
post_subject	wp_forum_posts.subject	
post_text	wp_forum_posts.text	

Tabelle 18: Spezielle Informationen für Postimport

4 Synthese

Die Definition der zu importierenden Daten von der Wordpress-Version des Preisbocks hin zum Magento gestaltet sich als sehr schwierig. Dieses resultiert durch die historisch gewachsene Datenbasis des Wordpress-Systemes. Aufgrund dessen, dass jene Version nicht als Shopsystem konzipiert aber trotzdem enorm viel in Richtung eines Shopsystemes erweitert wurde, litt die Datenbasis enorm, da jederzeit ein Update des Wordpress möglich sein musste. Des Weiteren entstanden durch alternierende Entwicklung viele Inkonsistenzen in den Datenbeständen des Preisbocksystems.
Das als zukünftige Shopplattform gewählte System Magento weist intern eine für einen Onlineshop optimierte Datenstruktur auf. Des Weiteren werden die Daten mit der Hilfe von Models automatisiert in die normalisierte Datenbasis eingeflegt. Damit Daten auch im Magentosystem nutzbar werden, müssen diese beim Import angepasst und optimiert werden. Weiterhin dürfen keine Abhängigkeiten zwischen den einzelnen Entitäten verloren gehen. Dafür wird definiert, dass wichtige Schlüsselattribute während des Imports zu den neuen ID im Magento notiert werden sollen. Dies ist jedoch nicht immer möglich, da bspw. bei den Kundenadressen auf eine MySQL-View zurückgegriffen werden muss, in der keine eindeutigen Primärschlüssel generiert werden können.
Die Anpassung der zu importierenden Attribute geschieht über Filterfunktionen. Diese müssen gleichzeitig die Daten validieren und dem Magentosystem anpassen.

Quellen

Kürzel	Quelle
[PWi06]	Peter Winkler: „M+T Computerlexikon“, Markt und Technik Verlag, 2000
[TKu07]	Thomas Kudraß, „Taschenbuch Datenbanken“, Hanser-Verlag, 2007
[PCS06]	Prof. Dr. V. Claus, Prof. Dr. A. Schwill, „Duden Informatik A-Z“, Dudenverlag, 4. Auflage, 2006
[DPb09]	Datenbank des Preisbocks der dotSource GmbH Stand 15.01.2009
[Tec08]	TechDivision GmbH, „Magento-Schnelleinführung“ 2008
[EEF06]	Eric & Elisabeth Freeman, „Entwurfsmuster von Kopf bis Fuß“, O’Reilly-Verlag, 2006
[GFo08]	Gerhard Fobe, „Projektarbeit 3 – Refactoring Preisbock“, 15.08.2008

Ehrenwörtliche Erklärung

Ich erkläre hiermit ehrenwörtlich, dass ich meine Praxisarbeit mit dem Thema

„Migration der Daten des aktuellen Preisbocks auf das Shopsystem Magento der Firma Varien“

ohne fremde Hilfe angefertigt habe,

dass ich die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe und dass ich meine Praxisarbeit bei keiner anderen Prüfung vorgelegt habe.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Ort, Datum	Unterschrift