

User Generated Content

Seminar Webprogrammierung / Web 2.0 Technologien

Sommersemester 2008

Sebastian Kölle, Emilia Wittmers

Hasso-Plattner-Institut, Universität Potsdam

Zusammenfassung Im Rahmen des Seminars Webprogrammierung / Web 2.0 Technologien wurde die Benutzerfreundlichkeit des Webportals ‚Cocato‘ durch Automatisierung der Konferenzerstellung verbessert. Das Anlegen von Konferenzen mit Konferenzdaten aus eingegangenen E-Mails, die zuvor per Hand in das Portal eingepflegt werden mussten, wird nun semi-automatisch vorgenommen. Die Webanwendung ermittelt automatisch zu den gewünschten Daten Vorschläge, die vom Nutzer akzeptiert, abgelehnt oder modifiziert werden können.

1 Einleitung

Cocato ist ein Social Network Prototyp für Teilnehmer von wissenschaftlichen Konferenzen und wurde am Lehrstuhl ‚Internet-Technologien und Systeme‘ des Hasso-Plattner-Institutes entwickelt. Mithilfe von Cocato lassen sich Konferenzeinladungen verwalten und unter Freunden und Kollegen kommunizieren.

Das Konferenzportal gehört in die Kategorie der Web 2.0-Anwendungen, die seit einigen Jahren immer größere Popularität erlangen. Der Begriff Web 2.0 steht für die interaktive Nutzung des Internets und ist durch die veränderte Wahrnehmung des WWW durch seine Nutzer geprägt. Der Benutzer kann in hohem Maße selbst über die Inhalte des WWW bestimmen, diese erstellen und bearbeiten.

Ein wichtiger Aspekt von Web 2.0-Anwendungen ist der sogenannte benutzergenerierte Inhalt (User Generated Content). Automatisch schwer handhabbare Aufgaben lassen sich häufig durch das Einbinden des Nutzers lösen.

In diesem Zusammenhang wurde die Verwaltung von Konferenzeinladungen in Cocato um die Funktion erweitert, das Auffinden von wichtigen Informationen, wie Titel, Veranstaltungsort und –datum, automatisch durchzuführen. Auf Grund der Komplexität der vollautomatischen Suche wird der Nutzer in den Auffindungsprozess eingebunden, um die Qualität der Ergebnisse zu verbessern.

2 Problemstellung

Cocato überprüft in regelmäßigen Zeitabschnitten einen zentralen E-Mail-Account auf neu eingegangene Konferenzeinladungen. Sind neue Nachrichten vorhanden, werden diese in die Datenbank des Portals geladen. Der Inhalt der E-Mail und Metadaten, wie Absender und Ankunftsdatum, die bereits beim Eintreffen der Nachricht vorliegen, werden als Cfp-Objekt (Call For Paper) gespeichert. Diesem Objekt sollen jedoch weitere Attribute wie Titel der Konferenz, Veranstaltungsort und –datum hinzugefügt werden können.

Bisher war es so, dass der Nutzer des Portals sich ein unbearbeitetes Cfp vornahm und aus dem gegebenen Einladungstext per Hand die Daten in die dafür vorgesehenen Felder kopierte.

Abb 1: Die Informationen Titel, Ort, Startdatum, Enddatum etc. müssen per Hand vom Nutzer aus der gegebenen E-Mail in die entsprechenden Felder geschrieben bzw. kopiert werden.

Da dieses Vorgehen aber sehr zeitaufwändig ist, soll dem Nutzer ein Teil der Arbeit abgenommen werden. Der Inhalt der Konferenzeinladungen soll dazu automatisch nach den Daten durchsucht werden.

Zu beachten ist dabei jedoch, dass die gewünschten Angaben in der Regel nur selten einem bestimmten Muster folgen und somit nicht immer ganz leicht durch Algorithmen aufzufinden sind. Die von dem System gefundenen Informationen werden dem Nutzer daher nur vorgeschlagen. Er kann sie gegebenenfalls modifizieren oder Informationen, die von Cocato nicht identifiziert werden konnten, hinzufügen.

3 Lösungsansatz



Abb 2: Ablauf der Cfp-Bearbeitung

Der Ablauf der Konferenzerstellung gliedert sich in mehrere Schritte. Das Portal prüft in regelmäßigen Zeitabständen, ob neue E-Mails auf dem E-Mail-Server vorliegen und holt diese gegebenenfalls ab. Der E-Mail-Text wird automatisch analysiert und die relevanten Informationen werden extrahiert.

Alle eingetroffenen, aber noch unbearbeiteten Einladungen werden dem Benutzer in Cocato in einer Liste präsentiert. Dort kann nun eine Einladung ausgewählt werden, um daraus eine Konferenz zu erstellen. Anschließend werden dem Nutzer die im Text gefundenen Informationen präsentiert, aus denen er die korrekten auswählen bzw. diese von Hand korrigieren kann.

Um Fehler und Manipulationen bei der Erstellung von Konferenzen zu vermeiden, muss eine erstellte Konferenz zunächst noch durch einen anderen Nutzer bestätigt werden (Vier-Augen-Prinzip). Dazu werden alle erstellten, aber unbestätigten Konferenzen in einer Liste präsentiert. Bei der Bestätigung hat der Nutzer noch einmal die Chance, die Konferenzdaten zu korrigieren.

Der komplexeste Schritt dieses Ablaufs ist die automatisierte Analyse der eingehenden E-Mails. Die gesuchten Informationen können in einer Vielzahl unterschiedlicher Formate im Text auftauchen (z. B. verschiedene Datumsformate), was das Extrahieren dieser Informationen erheblich erschwert.

Aus diesem Grund war es eine wichtige Voraussetzung, die Informationsextraktion so erweiterbar und wartbar wie möglich zu gestalten. Um dies zu erreichen wurde die Suche nach Informationen auf verschiedene Agenten aufgeteilt, die unabhängig voneinander den Text durchsuchen. Auf diese Weise ist es sehr einfach, bestehende Agenten zu verbessern oder komplett auszutauschen, oder auch neue Agenten für andere Daten einzufügen.

Bei der Informationsextraktion werden dazu folgende Schritte durchlaufen:

1. Vorbereitung der E-Mail für die Agenten: Ein Wörterbuch wird erstellt, das wichtigen Begriffen (z. B. Ortsnamen) ihre Bedeutung zuordnet. Hierfür wird der OpenCalais-Webservice verwendet [4].
2. Analyse: Alle verfügbaren Agenten untersuchen nacheinander den E-Mail-Text und stellen daraus gefundene Informationen zusammen.
3. Nachbereitung: Die Ergebnisse der verschiedenen Agenten werden bewertet, gefiltert und zusammengefasst.

Die nach dem letzten Schritt verfügbaren Daten können schließlich für die Präsentation im User Interface benutzt werden.

4 Umsetzung

4.1 Architektur

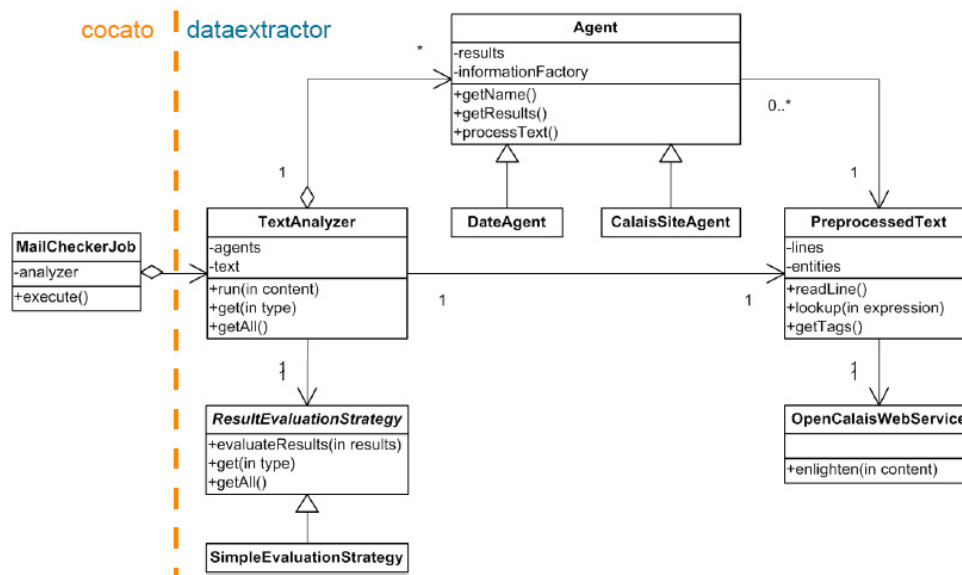


Abb 3: Die Architektur im Klassendiagramm

Das Projekt gliedert sich in drei Pakete, die das Analysieren der E-Mails implementieren. Das Paket *dataextractor* enthält die Verwaltungsfunktionen für die Agenten und erleichtert den Zugriff auf diese. Die Agenten selbst befinden sich im Paket *agents*, die Nachbereitung der Agentenergebnisse ist im Paket *evaluation* enthalten. Diese Pakete wurden komplett in Java implementiert, was insbesondere das separate Testen der Funktionalität

ten erleichtert. Dank der guten Integration von Groovy und Java ist es jedoch ohne weiteres möglich, die Pakete in Cocato zu verwenden.

Cocato benutzt die Klassen dieser Pakete, um eintreffende E-Mails zu analysieren.

Der Zugriff erfolgt dabei über die Klasse *TextAnalyzer*, die die Schnittstelle zum Portal darstellt (Fassade-Pattern [2]). Durch Aufruf der Methode *run()* wird die Analyse gestartet, die Ergebnisse der einzelnen Kategorien lassen sich mit Hilfe der Methoden *get()* und *getAll()* ermitteln.

Beim Aufruf von *run()* wird intern zunächst ein Objekt vom Typ *PreprocessedText* erzeugt, das die E-Mail für die Agenten vorbereitet. Dazu gehört insbesondere das Aufteilen in einzelne Zeilen und das Finden spezieller Schlüsselwörter. Die von *PreprocessedText* bereitgestellten Methoden, wie z. B. *lookup* zum Nachschlagen von Schlüsselwörtern, können später von allen Agenten beliebig genutzt werden.

Zum Ermitteln der Schlüsselwörter wird der OpenCalais-Webservice benutzt. Der Zugriff darauf ist in der gleichnamigen Klasse implementiert.

Wurde der Text erfolgreich vorbereitet, wird er an eine Liste mit Agenten übergeben. Die Agenten durchsuchen den Text jeweils nach Informationen und speichern diese zusammen mit zusätzlichen Daten, wie dem Informationstyp und der Position im Text.

Nachdem die Analyse durch die Agenten abgeschlossen ist, müssen die einzelnen Ergebnisse bewertet und zusammengestellt werden. Dies kann von Klassen erledigt werden, die die Schnittstelle *ResultEvaluationStrategy* implementieren. Hier können beispielsweise doppelt gefundenen Informationen entfernt und die Ergebnisse bewertet werden.

4.2 Integration in das Portal

Zugriff auf die gefundenen Informationen

Wenn Agenten bei der Analyse der E-Mails Informationen finden, so müssen diese auch in Cocato verfügbar sein und insbesondere in der Datenbank gespeichert werden. Um dies zu erreichen, bot sich der Einsatz des Factory-Patterns [2] an:

Das Paket *dataextractor* arbeitet intern mit dem Interface *ExtractedInformation*, das alle notwendigen Methoden zum Zugriff auf die gespeicherten Informationen beinhaltet. Mit Hilfe der *ExtractedInformation.Factory* können entsprechende Objekte erzeugt werden.

Das konkrete Factoryobjekt bekommt der *TextAnalyzer* dabei vom Portal geliefert, sodass es die Implementierung der Factory und der *ExtractedInformations* nicht kennen muss.

Auf diese Weise können *EmailInformations* in Cocato Teil des Grails Domainmodel sein und die Schnittstelle *ExtractedInformation* implementieren. Dies ermöglicht eine leichte Verwaltung der Informationen und garantiert gleichzeitig einen einfachen Zugriff auf diese aus dem *dataextractor*-Paket.

Weiterhin hat diese Vorgehensweise den Vorteil, dass das *dataextractor*-Paket die konkreten Implementierungen der Informationen nicht kennen muss. Somit bleibt das Paket wieder verwendbar und kann zum Beispiel auch für andere Textanalysen eingesetzt werden.

Anpassungen am Portal

Um die gewünschte Funktionalität und insbesondere das User-Interface zu realisieren, waren einige Änderungen an Cocato erforderlich:

- *MailCheckerJob*: Nach dem Abholen einer E-Mail muss diese zunächst auf relevante Daten untersucht werden. Dank der einfachen Schnittstelle des *dataextractor*-Pakets, kann dies durch Einfügen weniger Zeilen realisiert werden.
- *Controller*: Als wesentlicher Punkt sollte der Ablauf beim Erstellen einer Konferenz verbessert werden. Hierzu musste im *ConferenceController* die Aktion *createConference* überarbeitet werden. Es werden dabei die vorgeschlagenen Werte für die einzelnen Felder ermittelt und an den View weitergeleitet.

Weiterhin mussten noch zwei neue Aktionen hinzugefügt werden, die das Bestätigen erstellter Konferenzen ermöglichen. Zum einen war dies mit *unconfirmedCfPs* die Liste aller unbestätigten CfPs, zum anderen die Aktion *confirmConference*, die eine erstellte Konferenz bestätigt.

- *Views*: Für diese neuen Aktionen mussten jeweils auch entsprechende Views angelegt werden (*unconfirmedCfPs* und *confirmConference*). Weiterhin wurde das User Interface zum Erstellen von Konferenzen erheblich überarbeitet. Dabei werden dem Nutzer in der Startansicht übersichtlich alle gefundenen Informationen präsentiert. Entscheidet sich der Nutzer einen Wert zu überarbeiten, so wird mit einem Klick auf die Information mittels Ajax ein Menü mit den Vorschlägen des Systems aufgeklappt (Aktion *editStringItem/editDateItem*). Aus diesen Vorschlägen kann der Nutzer einen auswählen oder den Text per Hand ändern.

Bei Datumsangaben wird zusätzlich ein Kalender angezeigt, der die Auswahl des korrekten Datums vereinfacht. Durch einen Klick auf „OK“ wird die gewählte Information übernommen (Aktion *updateStringItem/editDateItem*). Mit einem Klick auf „Konferenz erstellen“ werden die Daten schließlich in der Datenbank gespeichert.

Implementiert wird ein wichtiger Teil dieses User Interfaces in der *CocatoTagLib*. Diese enthält zwei neue Tags *dateList* und *stringList* für die aufklappbaren Menüs, die einfach in das bestehende Formular eingebunden werden können. So kann das gleiche Verhalten an vielen unterschiedlichen Stellen im Formular benutzt werden, ohne doppelten Code zu schreiben.

5 Fazit

Wir konnten einen wesentlichen Teil unserer gesetzten Ziele erreichen. Eingehende E-Mails werden nun automatisch analysiert, wobei besonders bei Datumsangaben, URLs und E-Mail-Adressen eine hohe Trefferquote erzielt wird. Durch die flexible Architektur ist es jederzeit möglich, beliebige Agenten auszutauschen und weiter zu verbessern.

Einige Datentypen wie der Veranstaltungstitel konnten aus Zeitgründen noch nicht präzise genug ermittelt werden, die zuständigen Agenten müssen noch überarbeitet werden.

Unser User Interface präsentiert dem Nutzer die gefundenen Informationen übersichtlich und ermöglicht eine einfache Fehlerkorrektur. Durch das Vier-Augen-Prinzip wird garantiert, dass zwei Nutzer eine Konferenz bestätigt haben, bevor sie gültig ist.

Insgesamt haben uns die Funktionen von Grails, insbesondere der Zugriff auf die Datenbank mittels GORM, viele Routineaufgaben erleichtert. Andererseits haben unerklärliche Fehlermeldungen und die langsamen Reaktionen der Software unsere Arbeit erschwert. Für uns hat es sich als gute Entscheidung erwiesen, das Paket *dataextractor* in Java zu implementieren. Auf diese Weise war es uns möglich, die Agenten einzeln und getrennt vom Portal zu testen.

6 Quellen

1. Ullenboom, C.: Java ist auch eine Insel: Programmieren mit der Java Standard Edition Version 6 . Galileo Press, Bonn (2007)
2. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design patterns: Elements of reusable object-oriented software. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1995)
3. Rocher, G. K.: The Definitive Guide to Grails. Apress, Berkeley, USA (2006)
4. OpenCalais Webservice: www.opencalais.com