MASTER

Informationsmanagement und Computersicherheit

Mietwagenservice APP

Web-Service SINT-Projekt: MIC 2A/SS 2014

Diepold Markus (ic13m032)
Edelhofer Markus (ic13m005)
Makas Christian (ic13m035)

Aufgabenstellung

Web Service 1: Autoverleih (https://gate.edelhofer.org/sint/api)

Entwickeln Sie ein REST-Style Autoverleih Web Service: Das Buchen von Autos in verschiedenen Währungen soll möglich sein. Für die Währungsumrechnung soll folgendes Web Service verwendet werden:

Web Service 2: Währungsrechner: (http://remote.makas.at/ConversionService.asmx)

Entwickeln Sie ein SOAP/WSDL basierendes Währungsrechner Web Service. Dieses soll im Hintergrund die Daten von der Europäischen Zentralbank integrieren: http://www.ecb.europa.eu/stats/eurofxref/eurofxref-daily.xml Das Web Service soll auch sogenannte Cross-Rates berechnen und diese zur Verfügung stellen. Z.B. kann man aus den EUR/USD und EUR/GBP Kursen den USD/GBP Kurs berechnen.

Web Service 3: Google Maps API Web Services: (http://www.diepold.net/sint)

Verwenden Sie eines der Google Maps API Web Services: https://developers.google.com/maps/documentation/webservices/. Die integrierte Applikation soll die oben genannten Web Services benutzen. Es kann sich dabei um eine Web-Applikation oder eine Android oder iOS App handeln.

Verwendete Technologien

Web Service 1: Autoverleih

- Server
  - Scientific Linux release 6.5, Python 2.6.6, web.py 0.37, PySimpleSOAP 1.10, pysqlite 2.6.3, Apache 2.2.15, mod_wsgi-3.2-3, SQLite 3.6.20
- Entwicklungsumgebung
  - Fedora release 20, Python 2.7.5, web.py 0.37, PySimpleSOAP 1.10, pysqlite 2.6.3, SQLite 3.8.4.2
- Tools
  - gedit 3.10, SoapUI 5.0.0, Sqliteman 1.2.2, json.parser.online.fr

Web Service 2: Währungsrechner

- Server
  - Windows Server 2012 R2 Standard, IIS 8.5
- Entwicklungsumgebung
  - Visual Studio 2012/2013, Windows 8 Professional, IIS 8.5

Web Service 3: Google Maps API Web Services

- Server
  - Debian 7 (Codename Wheezy), Apache 2.2.15, PHP 5.4,
- Entwicklungsumgebung
  - Windows 8 Ultimate, XAMPP Control 3.2.1, Apache 2.7.4, PHP 5.5.9
- Tools
  - Notepad++ 6.3.2, jQuery 1.4.2, CSS3Template.co.uk, Google Web API, Github Code Repository, json.parser.online.fr

Programmbeschreibung

Web Service 1: Autoverleih

Das REST-WebService ist in Python (2.7 bzw. 2.6) geschrieben. Als Framework wird Web.py mit einer SQLite Datei als Datenbank verwendet. Durch Web.py lässt sich das WebService lokal direkt starten und besitzt auch einen Debug-Modus. Für den Produktions-Server wird ein Apache-httpd vorgeschalten und mittels WSGI verbunden. In der Datenbank gibt es fünf Tabellen:

- Table cars Bestandsliste der Fahrzeuge
- Table countries Tabelle mit allen Europäischen Ländern, Länderkürzel und Währungen
- Table customers Kundenliste
- Table locations Liste der Verleihstandorte
- Table rental Verwaltung der verliehenen Fahrzeuge

Alle Informationen werden mittels JSON ausgetauscht.

Programms Beschreibung (Klassen)

URL (Klasse)	Beschreibung
`/ (index)`	Ausgabe der Schnittstellen Übersichtsdokumentation als HTML
`/cars (show_cars)`	Mittels GET werden alle Vorhanden Fahrzeuge, sowie ihre Standort und ihre Unmittelbare Verfügbarkeit angezeigt. Dei der Standort Ausgabe wird der Name der Stadt aus der Tabelle 'locations' gelesen, alle weiteren Informationen kommen aus der Tabelle 'cars'.
`/cars/(.*) (show_car)`	Mittels GET können die Fahrzeuge auch einzeln ausgelesen werden.
`/rent (rent)`	Mittels GET werden alle Leihen angezeigt (Tabelle 'rental'). Die Personen Informationen kommen dabei aus der Tabelle ‚customers', die der Fahrzeuge aus 'cars'. Bei Aufruf dieser Klasse wir am Beginn die Methode 'set_rent_state' (siehe weiter unten), aufgerufen um einen aktuellen Status der Leihe zu Gewährleisten.
`/customers (customers)`	Mittels GET werden alle Kunden ausgegeben, mit PUT und POST kann ein neuer Kunde hinzugefügt werden.
`/customers/(.*)/rent (show_customers_rent)`	Mittels GET werden alle Leihen eines Kunden angezeigt.
`/customers/(.*) /customer)`	Mittels GET werden die Informationen über einen einzelnen Kunden ausgegeben, mit PUT und POST können die Informationen des Kunden geändert und mittels DELETE kann der Kunde gelöscht werden.
`/countries (show_countries)`	Diese Funktion gibt alle Europäischen Länder, sowie den dazugehörigen UN/LOCODE und die Währungen aus.
`/locations (show_locations)`	Mittels GET werden alle Standorte an denen ein Fahrzeug ausgeliehen und zurückgegeben werden kann ausgegeben.
`/locations/(.*) (show_locations_cars)`	Mittels GET werden alle Fahrzeuge die an dem jeweiligen Standort zur Verfügung stehen ausgegeben.

Programms Beschreibung (Methoden)

URL (Klasse)	Beschreibung
`cus_out(output)`	Diese Methode wird verwendet um den db-Output auf ein JSON Syntax umzuschreiben.
`soap_req(sC,sP,tC)`	Diese Methode ruft das SOAP-Service zur Währungsumrechnung auf. Im Fehlerfall werden der Betrag und Währung wieder in EUR zurückgegeben.
`/cars/(.*) (show_car)`	Diese Methode überprüft die 'rental' Tabelle ob ein Fahrzeug im Moment verfügbar ist und passt diese Information bei dem jeweiligen Fahrzeug an.

Web Service 2: Währungsrechner

Das WebService besteht aus einer Methode „`ConvertPrice`", welche als Eingabe das Kürzel der `Quellwährung (string)`, den `QuellPreis (double)` und das Kürzel der `Zielwährung (string)` erwartet.

Es wird das XML mit den Umrechnungswerten (http://www.ecb.europa.eu/stats/eurofxref/eurofxref-daily.xml) eingelesen und in eine Tabelle mit Währungskürzel und Umrechnungswert geschrieben. Im nächsten Schritt werden die zu den angegebenen Quell- und Zielwährungen die Umrechnungswerte aus der Tabelle ausgelesen und der entsprechende Zielwährungswert berechnet, welcher dann an das aufrufende WebService zurück gegeben wird.

Web Service 3: Google Maps API Web Services

Die Web-Service APP basiert auf einem freien CSS3/HTML5 Template und wurde vorwiegend in PHP geschrieben. Die Menüfunktion basiert auf CSS und jQuery. Für die Google-API wurde JavaScript verwendet.

Die Google API wurde folgendermaßen implementiert:

- Zuerst werden für jeden Standort die Informationen Ort, Postleitzahl und Straße an das Google Geocode-Service übermittelt.
- Als Antwort halten wir die Longitude/Latitude Informationen des jeweiligen Standortes.
- Diese Daten nehmen wir nun im PHP Script entgegen, und erzeugen via `json_encode(array)` ein Array welches wir an JavaScript übergeben.
- Im Javascript erzeugen wir mit `JSON.parse(my_json)` ein Array (von Java Objekten), welches wir über die Methode `getArray(object)` in ein Datenarray umwandeln.
- Dieses Datenarray übermitteln wir dann an die zweite Google API mit der wir alle Standorte in die Übersichtskarte der Google MAP einzeichnen.

Die APP beinhaltet folgende Webservice-Funktionen:

Webservice-Methoden	Beschreibung
`my_curl_PUT($data_string, $url)`	Die Methode wird verwendet um ein JSON an eine URL per PUT zu übermitteln. Der Rückgabewert der Funktion ist die http Rückmeldung.
`my_curl_POST($data_string, $url)`	Die Methode wird verwendet um ein JSON an eine URL per POST zu übermitteln. Der Rückgabewert der Funktion ist die http Rückmeldung.
`my_curl_DELETE($path)`	Die Methode wird verwendet um ein DELETE an eine URL zu übermitteln. Der Rückgabewert der Funktion ist die http Rückmeldung.

Für die APP wurden zusätzlich die Stylesheets angepasst und folgende Funktionalitäten implementiert:

PHP-File	Beschreibung
index.html	Startseite der APP
`standorte.php`	Listet alle Standorte und gibt zusätzlich die Geodaten aus und zeichnet die Übersichtskarte mit Pins aller Standorte.
`autos.php`	Listet alle Autos mitsamt den aktuellen Stati der Leihe und dem aktuell zugewiesenen Standort. Der Basispreis wird in der Übersicht immer in EUR ausgegeben.
`mitglieder.php`	Listet alle Mitglieder mit Adressdaten und erlaubt das Hinzufügen von neuen Benutzern. Jeder Benutzer hat einen direktlink auf seine Detaildaten.
`mitglieder_detail.php`	Listet die Detailinformationen des Benutzers. Erlaubt das ändern und das Löschen vom Benutzer. Die Buchungsübersicht zeigt alle gebuchten Fahrzeuge des Benutzers in der aktuellen Landeswährung.
`mitglieder_rent.php`	Erlaub das Erstellen von neuen Buchungen und zeigt wieder die Buchungsübersicht der schon vorhanden.
`buchungen.php`	Listet alle aktuellen Buchungen mit dem Status. Die Ausgabe der Währung erfolgt hier immer in EUR.
`contact.php`	Kontakformular für Wünsche, Anregungen.
`Weitere Links:`	• Dokumentation der APP • Schemadiagramm der APP • REST/SOAP Cars Rental (https://gate.edelhofer.org/sint/api) • SOAP Currency Converter (http://remote.makas.at/ConversionService.asmx) • Quellcode auf GitHub (https://github.com/vandenbergen/sint)

Lessons Learned

Web Service 1: Autoverleih

- Die Wahl des richtigen Python / REST Framework muss gut überlegt bzw. überprüft werden. Es wurde beim Goolge-Reasearch darauf geachtet ob es eine ausreichende Community gibt, welche das Framework verwendet.
- Die Schnittstellen-Doku sollte über den REST-Server ausgegeben werden, damit man schnell sieht welche Funktionen bereits implementiert sind.
- db-Struktur muss gut überlegt werden (Und sollte nicht so wie uns unserem Fall schon existieren, da sie mehrfach geändert wurde).
- Beim Aufrufen des REST-WebService, bzw. beim Testen der Aufrufe/Verbindung ist eine Einsicht in die (in diesem Fall) Apache Logs unerlässlich, um schnell festzustellen was am WebService ankommt. Dieses Problem wurde per Telefonkonferenz gelöst.

Web Service 2: Währungsrechner

Es mussten aus der Web.conf, welche das VisualStudio automatisch erstellt die Zeilen:

„<section name="scriptResourceHandler" type="System.Web.Configuration.ScriptingScriptResourceHandlerSection, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35" requirePermission="false" allowDefinition="MachineToApplication"/> <section name="jsonSerialization" type="System.Web.Configuration.ScriptingJsonSerializationSection, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35" requirePermission="false" allowDefinition="Everywhere"/> <section name="profileService" type="System.Web.Configuration.ScriptingProfileServiceSection, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35" requirePermission="false" allowDefinition="MachineToApplication"/> <section name="authenticationService" type="System.Web.Configuration.ScriptingAuthenticationServiceSection, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35" requirePermission="false" allowDefinition="MachineToApplication"/> <section name="roleService" type="System.Web.Configuration.ScriptingRoleServiceSection, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35" requirePermission="false" allowDefinition="MachineToApplication"/>“

gelöscht werden, da sie schon in der Standard-Config des IIS vorhanden waren und das Web-Service daher Fehlermeldungen generierte.

Web Service 3: Google Maps API Web Services

- Bei der APP wurde bewusst auf Frameworks verzichtet. Daher war das konvertieren der übermittelten JSON-Daten ein guter Mix aus Try&Error und Internet-Recherche. Leider funktionieren viele Codebeispiele aus dem Netz nicht korrekt.
- Die Schnittstelle PHP/JavaScript für die Google-API war nicht ganz so trivial zu lösen.
- Im Zuge der APP-Entwicklung haben wir erst festgelegt welche Daten wir in der Datenbank brauchen und wie unsere JSON-Files aussehen werden. Skype, Telefon, GitHub haben sich hier als helfend herauskristallisiert.
- Die DELETE Message wurde durch den Apache Server selbst von extern blockiert. Hier hilft nur die Analyse der Logfiles und Verbindungsdaten.