Die approbierte Originalversion dieser Diplom-/Masterarbeit ist in der Hauptbibliothek der Technischen Universität Wien aufgestellt und zugänglich.
http://www.ub.tuwien.ac.at

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology.
http://www.ub.tuwien.ac.at/eng

Unterschrift des Betreuers

TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

Diplomarbeit

Haskell OpenLayers Wrapper

Ausgeführt am Departement für Geodäsie und Geoinformation

Forschungsgruppe Geoinformation

der Technischen Universität Wien

unter der Anleitung von **Prof. Dr. Andrew Frank** und
Mag. Jürgen Hahn als verantwortlich
mitwirkenden Universitätsassistenten

durch

Thomas Edelmann

Datum

Unterschrift (Student)

Chuck Norris doesn't use strongly-typed languages.

He uses strong languages.

---

http://codesqueeze.com/the-ultimate-top-25-chuck-norris-the-programmer-jokes/

Erklärung zur Verfassung dieser Arbeit

Thomas Edelmann
Marktgasse 6/44
1090 Wien
tomnobleman@gmail.com

Hiermit erkläre ich, Thomas Edelmann, dass ich diese Arbeit selbstständig verfasst und das Programm selbstständig entwickelt habe ohne Benutzung anderer als der angegeben Hilfsmittel und Quellen. Ich habe jene Stellen der Arbeit, die anderen Werken entnommen sind, unter Angabe ihrer Quelle kenntlich gemacht.

Wien, 2015

Danksagung

Ich möchte mich hiermit herzlichst bei Prof. Andrew Frank und Jürgen Hahn für die Unterstützung und Bereitstellung dieses interessanten Themas bedanken.

Vielen Dank auch an meine Eltern und ihre Partner ohne die dieses Studium in vielerlei Hinsicht nicht möglich gewesen wäre. Danke für die Geduld.

Besonderen Dank geht auch an Maria und David, Muci und Andreas samt Familie, Andrea und Gerhard.

Danke an alle Organisationen und Personen, welche Open Source Software oder freie Software zur Verfügung stellen.

Für Emilija,

Worte können nicht annähernd die Besonderheit meines kleinen Mädchen ausdrücken.

Dieses Buch ist dir gewidmet.

Anmerkungen

A)
Bei allen Bezeichnungen, die auf Personen bezogen sind, betrifft die gewählte Formulierung beide Geschlechter, auch wenn aus Gründen der leichteren Lesbarkeit nur eine Geschlechtsform gewählt wurde.

B)
Der praktische Teil der Arbeit umfasst eine Software. Der Quellcode dazu ist am Ende der Arbeit angeführt. Die digitale Version steht auf einer öffentlichen Online-Plattform zum Download zur Verfügung (siehe Kapitel 1.6, Seite 4).

C)
Für die globale Integration des Projektes erfolgen die Dokumentation und Kommentare im Quellcode in englischer Sprache.

D)
Webseiten, die in Fußnoten vorkommen, waren im Jänner 2015 online verfügbar.

E)
Dieses Dokument wurde mit LaTeX erstellt.

F)
Die Zitierung der Literaturquellen erfolgt durch ein Kürzel der Autoren und Jahreszahl in eckigen Klammern. Bei Quellen aus dem Web ohne Angabe der Jahreszahl.

Zusammenfassung

Die Grundlage zur Erarbeitung dieses Themas liefert die vielschichtige Entwicklung von Webtechnologien und in weiterer Folge die steigende Bedeutung und Notwendigkeit der Visualisierung von Geodaten auf Weboberflächen. Eine vermehrte Verwendung von digitalen Karten im Web ist beispielsweise auf den Kontaktseiten von Websiten zu erkennen. Weiters wurde es durch die Ausbreitung von Smartphones mit inkludiertem Standortempfänger für die Benutzer einfacher, sich mit dem Thema digitaler Karten und standortbezogener Dienste auseinanderzusetzen. Diesem Trend folgte in den 2000er Jahren die Entwicklung von mehreren Webdiensten für Geoinformationssysteme (GIS), wie z. B. OpenLayers. Solche Dienste ermöglichen, in wenigen einfachen Schritten Geodaten auf Webseiten einzubinden.

Um im Web Kartendienste implementieren zu können, werden hauptsächlich Programmbibliotheken und Programmierschnittstellen (APIs) in JavaScript zur Verfügung gestellt. JavaScript ist eine bekannte Skriptsprache und moderne Webtechnologie, welche seit 1995 stetig weiterentwickelt wird. Jedoch sind es viele Webentwickler leid mit JavaScript zu arbeiten und dies wird besonders von der Haskell-Community als **The JavaScript Problem** [Com] publiziert und erläutert. Zur Umgehung der Entwicklung mit JavaScript existieren bereits einige funktionierende Lösungsansätze. Dabei wird JavaScript Code anhand einer anderen Programmiersprache transkompiliert.

Haskell ist eine rein funktionale Programmiersprache und liefert mit dem Paket FAY [BDJP] eine durchwegs praktikable Lösung des ‚JavaScript Problems'. Somit stellt sich die Frage, ob mit Haskell das Erstellen einer Web-GIS-Anwendung mit den Funktionen von OpenLayers möglich ist.

Das Produkt wird **Haskell OpenLayers Wrapper** (kurz: **olwrapper**) bezeichnet. Der olwrapper ermöglicht eine einfache Handhabung als Webframework und API während der Entwicklung für digitale Kartenanwendungen.

Abstract

This work is based on the multifaceted development of web technologies and the growing significance and necessity of visualization of spatial data on web interfaces. The increased use of digital maps on the web can for example be seen at the use of contact pages of company websites. Furthermore, the spread of mobile devices with GPS receivers made it easier for users to deal with the issue of digital maps and location-based services. This trend was followed in the 2000s due to the developement of several web services to achieve geographic information systems, such as OpenLayers. These kinds of web services allow a simple integration of spatial data on individual websites in just a few steps.

In order to implement web map services, mostly program libraries and application programming interfaces in JavaScript are provided. JavaScript is a regularly used scripting language and modern web technology, which is constantly being developed since 1995. However, there are many web developers weary of working with JavaScript. This change in attitute is published and explained by the Haskell community as **The JavaScript Problem** [Com]. Convenient alternatives to JavaScript are already in existence. In order to avoid using JacaScript different solutions transcompile the JavaScript code based on another programming language.

Haskell is a purely functional programming language. The package FAY [BDJP] for example offers a practical solution of 'The JavaScript Problem'. This development raises the question of wether or not it is possible to create a web GIS application with the functions of OpenLayers in Haskell.

The final product is called **Haskell OpenLayers Wrapper** (short: **olwrapper**). The olwrapper is easy to use as a web framework and an API during the development of a web map.

Inhaltsverzeichnis

1 Einleitung

1.1 Allgemeines

Diese Arbeit befasst sich im Allgemeinen mit folgenden zwei Begriffen:

1. World Wide Web (kurz: Web)
2. Geodaten

Aus diesen beiden lassen sich alle weiteren Begriffe für diese Arbeit ableiten. Zum einen entwickelten sich aus dem Web mehrere verschiedene Methoden, Techniken und Verfahren, zum anderen beruhen Geoinformationsysteme (GIS) auf dem Vorhandensein der entsprechenden Geodaten. Aus der Verknüpfung der beiden Begriffe resultiert die Entwicklung von Online-Geoinformationsysteme (Web-GIS).

Das Online-Geodatenmanagement kann folglich der Geoinformatik zugeordnet werden. Diese interdisziplinäre Wissenschaft verknüpft Informatik und Geowissenschaften.

1.2 Fragestellung

Kann mit **Haskell** eine Programmierschnittstelle erstellt werden, mit welcher es möglich ist, die JavaScript Bibliothek von **OpenLayers** bedienen zu können?

1.3 Methodik

Um die Funktionen einer OpenLayers JavaScript Bibliothek in der Programmiersprache Haskell zu transkompilieren wurde ein sukzessiver Ansatz für die Entwicklung des Programms verwendet. Im ersten Schritt wurden vorhandene Werkzeuge untersucht und kleinere ‚Programmschnipsel' entworfen, um die notwendigen Funktionen testen zu können. Somit konnten die zu verwendenden Programmteile

verifiziert und der Nutzen für den **Haskell OpenLayers Wrapper** (olwrapper) bestätigt werden. Der geläufige Einstieg, mit einem kleinen Hello-World-Programm eine neue Programmiersprache zu erlernen um schnell die verwendete Syntax zu verstehen, wurde dafür übernommen und adaptiert. Mit dem marginalen Unterschied nicht den Text "Hello World" darzustellen, sondern eine einfache Basiskarte anzeigen zu lassen (Hello-Map-Programm). Nach erfolgreicher Darstellung der Karte mittels der verifizierten Werkzeuge wurden anschließend die Funktionen von OpenLayers in kleinen Schritten dem **olwrapper** hinzugefügen.

Das Transkompilieren der gesamten OpenLayers JavaScript Bibliothek würde den Rahmen dieser Arbeit übersteigen. Deshalb wurden für diese Version zwei verschiedene Szenarien gewählt, deren Funktion für neue Kartenanwendungen möglich sein sollen. Ein Grundgerüst, auf welchem die beiden Szenarien aufbauen, wurde dazu als erstes entwickelt.

Funktionen des Grundgerüsts:

- Karte auf Webseite einbinden
- Basiskarte wechseln
- Interaktionen der Ansicht (zoomen, verschieben)
- Darstellung und Manipulation von Layern (Größe, Farbe, Deckkraft)
- Verknüpfung von HTML-Elementen mit OpenLayers Funktionen
- Koordinatentransformation für verschiedene Projektionen

Szenario 1 - öffentlichen Verkehr anzeigen : Verschiedene Layer sollen auf der Karte ein- und ausgeschalten werden können.

Szenario 2 - Kundenstandorte erzeugen: Mittels Texteingabe von Koordinatenpaaren sollen Punkte auf dem angegebenen Standort als Layer eingefügt werden.

1.4 Details zur Entwicklungsumgebung

Die Entwicklung des **Haskell OpenLayers Wrapper** erfolgt aus technischen Gründen mit dem Betriebssystem Linux. Eine damit erstellte Webanwendung ist

jedoch plattformunabhängig und kann von jedem Browser, welcher JavaScript unterstützt, ausgeführt werden.

Für eine funktionierende Entwicklungsumgebung wurden die Haskell Plattform und die Programme in Tabelle 1 installiert. Eine ausführliche Beschreibung zur Installation und weiteren Parametern sind ab Kapitel 4.2 angeführt.

1	2	3
Texteditor Geany 1.23.1  `http://geany.org`	Versionsverwaltung Git 1.9.1  `http://git-scm.com`	Browser Firefox 34.0  `www.mozilla.org`

Tabelle 1: Art, Name, verwendete Version, Logos und Websiten der verwendeten Entwicklungssoftware

Die notwendigen Programmiersprachen, um das Projekt zu realisieren sind:

1. Haskell
2. JavaScript

Zusätzlich werden für die Webentwicklung die Auszeichnungssprache HTML und die Gestaltungssprache CSS verwendet. Zum effizienten Arbeiten, für die Installation und zur Problemlösung, sind weiters Kenntnisse in der Shellprogrammierung notwendig.

1.5 Aufbau der Arbeit

Kapitel 2 gibt einen Überblick über vorhandene Kartendienste für Web-GIS-Anwendungen und weitere Informationen zum Stand der Technik.

Darauf folgend beschäftigt sich Kapitel 3 mit den betreffenden Notwendigkeiten zur Erstellung des olwrapper in Bezug auf die Programmierung.

In Kapitel 4 wird die entwickelte Software beschrieben. Dort sind u. a. die Details zur Anwendung und zu einzelnen Funktionen aufgelistet. Weiters wird die Installation, Verwendung und Problembehandlung beschrieben.

Kapitel 5 beinhaltet Ideen zu weiteren Entwicklungsschritten und es wird abschließend über die bisherige Entwicklung resümiert.

Noch vor dem dokumentierten Quellcode als Anhang befinden sich ein Glossar sowie die Verzeichnisse zu Tabellen, Abbildungen und Literatur.

1.6 Haskell OpenLayers Wrapper Link

Die in dieser Arbeit entwickelte Version 0.4.1 des Haskell OpenLayers Wrapper ist unter folgenden Links auf unbestimmte Zeit abrufbar:

- `https://hackage.haskell.org/package/olwrapper`
- `https://github.com/olwrapper/olwrapper`

2 Online-Kartendienste

2.1 Allgemeines

Es wird im folgenden Kapitel auf Online-Kartendienste für die Organisation und Präsentation von Geodaten eingegangen. Funktionen wie Routenplaner, Navigation, Geocodierung und dergleichen werden nicht behandelt, stehen aber zusätzlich bei einigen Diensten zur Verfügung; diese sind somit einem Geoinformationssystem zuzuordnen. Geoinformationssysteme sind Informationssysteme zur Erfassung, Bearbeitung, Organisation, Analyse und Präsentation räumlicher Daten. [VMG+01]

Bei den Anbietern von Geodaten und Programmierschnittstellen (API) handelt es sich meist um weltweit agierende Organisationen. Die API dient Entwicklern von Webseiten zur Integration von interaktiven Karten sowie weiteren kontextbezogenen Funktionen. Das vektor- oder rasterbasierte Kartenmaterial wird dabei anhand von Webservices generiert. Es existieren neben den globalen Anbietern von Geodaten jedoch noch weitere lokale Organisationen, deren Daten ebenso eingebunden werden können. In Österreich beispielsweise haben sich die Bundesländer zusammengeschlossen und bieten mit der sogenannten "Basemap"[1] freie Geodaten für das Bundesgebiet an.

Hauptsächlich werden die Online-Kartendienste von populären Betreibern bestimmter Internet-Suchmaschinen [JA10] angeboten (Bing, Google, Yandex). Durch die vorhandene Infrastruktur bieten diese Firmen, neben der eigenständigen Webseite zur Betrachtung der Geodaten, auch eine entsprechende Programmierschnittstelle für Webanwendungen an (siehe Kapitel 2.4).

Es existieren jedoch noch weitere Organisationen, die sich von den Suchmaschinenanbietern abgrenzen. Dazu gehören als bekannteste Vertreter der ‚Platzhirsch' MapQuest, Here des Telekommunikationskonzerns Nokia, sowie das für diese Arbeit relevante Projekt **OpenLayers**. Die Projekte OpenStreetMap[2] und Wikimapia[3]

[1] `http://www.basemap.at/`
[2] `http://www.openstreetmap.org`
[3] `http://wikimapia.org`

haben sich darauf spezialisiert, im Sinne eines geographischen Wiki, Geodaten zu sammeln und bereitzustellen. Die gesammelten Geodaten von OpenStreetMap können z. B. wiederum von den Online-Kartendiensten MapQuest und OpenLayers verwendet werden.

Zusätzlich sei an dieser Stelle noch die planetarische Web-GIS-Anwendung "NASA World Wind" erwähnt: Sie ist eine Open Source Software, mit welcher Satelliten- und Luftbilder der Planeten Erde, Mond, Mars, Venus und Jupiter angezeigt werden können.

2.2 Web Map Service

Der OpenGIS Web Map Service Interface Standard (WMS) ist eine Schnittstelle mit einer vom Open Geospatial Consortium (OGC) verfassten Schnittstellenspezifikation zum Anfordern rasterbasierter Auszüge von Geodaten über das Web. [ZL05] Eine WMS-Abfrage besteht aus einem oder mehreren Layern des gewünschten Anbieters sowie der Ausdehnung des gewählten Bereichs. Als Antwort werden korrespondierende Kacheln (Tiles) zurückgesendet und anschließend im entsprechenden Programm angezeigt. Die angeforderten Layer können sowohl im Vektor- als auch im Rasterformat vorliegen. Weitere Spezifikationen des OGC sind u. a. das Web Feature Service (WFS) oder Web Coverage Service (WCS), mit denen entweder Vektordaten bzw. raum- und/oder zeit-variierende Phänomene dargestellt werden können.[4]

2.3 Darstellung und Lizenzbestimmungen

Im Allgemeinen können Nutzer bei der Visualisierung der Geodaten zwischen folgenden zwei 2D-Ansichten wählen:

- Karte
- Bild (Satelliten- oder Luftbild)

Die Tabelle 2 zeigt die beiden Arten von Ansichten für den Raum Wien.

[4] `http://www.opengeospatial.org/standards/`

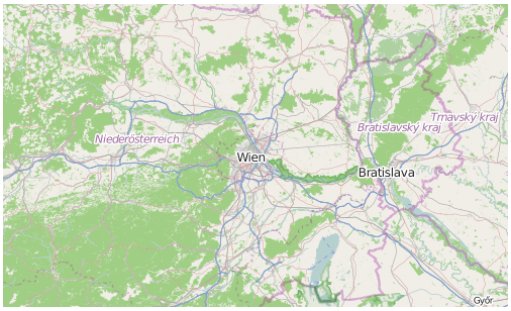
Karte	Bild
	
Quelle: OpenStreetMap	Quelle: MapQuest

Tabelle 2: klassische 2D-Kartenansichten im Web
(Zoomlevel: 8 \ Zentrum: [1825152.8,6142621.0])

Weitere Möglichkeiten der Darstellung sind:

- Hybrid: Kombination aus Bild- und Kartenmaterial
- 3D-Ansichten
- Vogelperspektive
- Photos aus der Straßenperspektive (z. B. Google's StreetView)

Zur rechtlichen Vollständigkeit werden anschließend noch einige Bemerkungen zu den **Lizenzbestimmungen** der verwendeten Geodaten erwähnt. Es gibt hierbei zwei zu differenzierende Bereiche: Einerseits die Daten, welche einer freien Lizenz oder Open Source Lizenz unterliegen und meist von einer freien Organisation oder Community bereitgestellt oder mittels Crowdsourcing gesammelt werden; andererseits kommerzielle Firmen (wie z. B. Bing oder Google), welche auch kostenlose Geodaten zur Verfügung stellen, jedoch die Weitergabe und Verwendung eigenen Lizenzbestimmungen unterliegen und somit eingeschränkte Nutzungsmöglichkeiten nach sich ziehen. Zudem handelt es sich bei den angebotenen Programmierschnittstellen um proprietäre Software und der Quellcode ist weder öffentlich zugänglich noch erweiterbar (vgl. Kapitel 2.4).

2.4 Anbieter

Diesem Kapitel abschließend folgt eine alphabetisch sortierte Liste von Anbietern bekannter APIs zur Erstellung von Web-GIS-Anwendungen.

Angegeben werden darin die aktuelle Version sowie Website zur aktuellen Anwendung und JavaScript Programmierschnittstelle. Es werden auch Dienste und zusätzliche APIs in anderen Programmiersprachen und Versionen angeboten, die nicht zwingend plattformunabhängig sind.

Zur privaten oder gewerblichen Nutzung kann bei manchen Diensten eine Registrierung notwendig sein. Dadurch erhält der Entwickler eine eindeutige Identifikationsnummer (z. B. AppKey), die angegeben werden muss um Funktionen freizuschalten. Grundsätzlich ist für Privatpersonen die Verwendung kostenlos und erst bei gewerblicher Nutzung entstehen zusätzliche Kosten für die Bereitstellung der Geodaten. Eine Entwicklung mit OpenLayers als Open Source Software unter der Verwendung der Geodaten von OpenStreetMap, welche einer freien Lizenz unterliegen, ist kostenlos und eine Entwicklung somit ohne AppKey möglich. Die Jahreszahl in der Liste gibt an, wann der Dienst veröffentlicht wurde. [map] [Dun]

- Bing (2010)
  Bing Maps AJAX Version 7
  AppKey erforderlich
  Anwendung: `http://www.bing.com/maps/`
  Entwicklerseite:
  `http://www.microsoft.com/maps/choose-your-bing-maps-API.aspx`[5]

- Google (2005)
  Google Maps JavaScript API Version 3
  AppKey nicht zwingend erforderlich
  `http://maps.google.at/`
  `https://developers.google.com/maps/`

- Here (seit 2012, früher: Ovi Maps 2007-2011, Nokia Maps 2011-2012)
  Here Maps JavaScript API Version 3.0
  AppKey erforderlich
  `https://www.here.com/`
  `https://developer.here.com/`

[5] Anmerkung: diverse Versionen für verschiedene Windows-Plattformen verfügbar

- MapQuest (1996)
  JavaScript Maps API Version 7.2
  AppKey erforderlich
  `http://www.mapquest.com/`
  `http://developer.mapquest.com/`

- OpenLayers (2006)
  OpenLayers API Version 3.1.1
  AppKey nicht erforderlich
  `http://openlayers.org/`
  `http://openlayers.org/en/v3.1.1/apidoc/`

- Yandex.Maps (2004)
  Yandex Maps JavaScript API Version 2.0
  AppKey nicht zwingend erforderlich
  `http://maps.yandex.com/`
  `http://api.yandex.com.tr/maps/doc/jsapi/`

3 Programmierung/Entwicklungswerkzeuge

3.1 Haskell

`https://www.haskell.org/`

Abbildung 1: Website und Logo von Haskell

Haskell ist eine rein funktionale Programmiersprache, wodurch Programme als mathematische Funktionen aufgefasst werden. Sie wurde nach dem Mathematiker Haskell Brooks Curry benannt und existiert seit 1990. Sie wird ständig weiterentwickelt und ist in der aktuellen Version Haskell 2010 verfügbar. Haskell basiert auf dem Lambda-Kalkül, weshalb auch der griechische Buchstabe Lambda als Logo verwendet wird. [vEU10]

Zu ihren besonderen Eigenschaften zählen:

- rein funktional:
  Funktionen geben nur Werte zurück, ändern aber nicht den Zustand eines Programms. Es gibt keine Operationen die einen Variablenwert verändern und deshalb gibt es auch keine Unterscheidung zwischen Variablen und Konstanten.

- keine imperative Programmierung:
  Einerseits besteht durch Monaden die Möglichkeit, Ein- und Ausgabeoperationen und zustandsabhängige Berechnungen wie Zufallsgeneratoren rein funktional zu behandeln, andererseits wird u. a. durch syntaktischen Zucker das Aussehen von imperativer Programmierung imitiert, wie z. B. mit der `do`-Notation.

- nicht strikt (lazy evaluation):
  Ein auszuwertender Ausdruck wird nur so weit berechnet, wie es gerade benötigt wird. Es werden nur Ausdrücke ausgewertet, die für die Berechnung des Ergebnisses gebraucht werden.

- stark typisiert:
  Es wird streng zwischen Wahrheitswerten, Zeichen, ganzen Zahlen, Gleitkommazahlen und Funktionen von und zu verschiedenen Typen unterschieden.
- Modulsystem:
  Ein Haskell Programm besteht aus einer Sammlung von Modulen. Um Module nutzen zu können, werden sie mithilfe des `import`-Befehls importiert.

Anmerkungen zur Syntax von Haskell: Die Syntax von Haskell unterscheidet zwischen Groß- und Kleinschreibung. Mit einem Großbuchstaben beginnende Bezeichner definieren Typ- und Wertkonstruktoren, wobei solche mit Kleinbuchstaben für Typvariablen, Funktionen und Parameter stehen. Einzeilige Kommentare beginnen mit zwei Bindestrichen `--`, mehrzeilige Kommentare werden in `{-` und `-}` eingeschlossen. [Jon02]

Für die zeilenweise Einrückung im Quellcode sollen keine Tabulatoren sondern Leerzeichen verwendet werden. Eine Einrückung durch Leerzeichen dient bei Haskell nicht nur der besseren Lesbarkeit, sondern dadurch wird auch die Bedeutung verändert. Zudem kann bei richtiger Anwendung die Klammersetzung teilweise ersetzt werden.

3.1.1 Haskell Plattform

Die Haskell Plattform ist ein Standardset von Paketen, Werkzeugen und weiteren Sammlungen für eine funktionierende und robuste Haskell Entwicklungsumgebung. [CPJS08] Sie steht für die Betriebssysteme Linux, Mac und Windows als Archiv zum Download zur Verfügung.

Das Archiv ist nach folgender Struktur zusammengestellt: [has]

- Compiler
- Kernpakete
- zusätzliche Plattformpakete
- Programme und Werkzeuge

In dem Archiv sind neben dem Compiler zwei wichtige Pakete für die Entwicklung von neuen Paketen inkludiert: Zum einen Cabal als Paketmanager und zum anderen Haddock zur Erstellung von Dokumentationen für ganze Pakete, auf welche anschließend genauer eingegangen wird.

3.1.2 Cabal

The Common Architecture for Building Applications and Libraries (Cabal) dient zur standardisierten Definition und Erstellung von Haskell Paketen. Ein Cabal-Paket beinhaltet im Allgemeinen eine `<Paketname>.cabal` Datei. Sie dient u. a. zur Beschreibung des Pakets und zur Angabe von Abhängigkeiten zu anderen Paketen. Der Vorteil dieses Systems liegt in der Vermeidung von komplizierten betriebssystemspezifischen Abhängigkeiten sowie der Förderung von wiederverwendbaren Code durch solche Pakete. [CPJS08]

Cabal-Install

Cabal-Install dient zur Bereitstellung und Aktualisierung von Cabal. Mit folgendem Befehl wird Cabal aktualisiert:

```
cabal update
cabal install cabal-install
```

Mit dem Befehl “update” wird die lokal gespeicherte Paketdatenbank mit der Online-Datenbank von Hackage verglichen und bei Bedarf aktualisiert.

Die aktuell installierte Version von Cabal wird durch folgende Eingabe im Terminal ausgegeben:

```
cabal --version
```

Ausgabe bei installierter Version 1.22.0.0:

```
cabal-install version 1.22.0.0
using version 1.22.0.0 of the Cabal library
```

3.1.3 Haddock

In den einzelnen Modulen eines Haskell Pakets werden mittels Kommentaren die Funktionen, Datentypen und weitere Parameter beschrieben. Durch das zusätzliche Anordnen von Sonderzeichen in den Kommentaren wandelt Haddock die Texte anschließend in das gewünschte Format um und erstellt somit eine ausgestaltete Dokumentation. Mit Haddock können z. B. HTML-Dateien erzeugt und in weiterer Folge auf Hackage (siehe Kapitel 3.2.2) hochgeladen werden. Gleiche Definitionen in verschiedenen Modulen und Paketen können somit auf Hackage online miteinander verlinkt werden.

Mit folgenden Befehlen wird die Dokumentation der Haskell Plattform und Haddock aktualisiert:

```
sudo apt-get install haskell-platform-doc
cabal update
cabal install haddock
```

Eine erfolgreiche Installation von Haddock liefert folgende Ausgabe:

```
Installing executable(s) in /home/user/.cabal/bin
Registering haddock-2.13.2.1...
Installed haddock-2.13.2.1
```

Mit Cabal wird eine HTML-Dokumentation wie folgt erstellt:

```
cabal haddock --html
```

Eine ausführliche Dokumentation sowie Anleitung zur Benutzung von Haddock wird online im “Haddock User Guide” angeboten. [MW] Auch ein Export im LaTeX Format oder für Hoogle ist mit diesem Programm möglich.

3.2 Softwareverwaltung

Software wird in Haskell überwiegend auf folgenden Onlineplattformen hochgeladen und verwaltet:

- Hackage (nur für in Haskell entwickelte Pakete) [hac]
- GitHub (für verschiedene Programmiersprachen) [gitb]

Hackage fungiert zur Bereitstellung der Paketverwaltung im Rahmen der Entwicklung mit Haskell und in weiterer Folge Cabal. Es ist die zentrale Datenbank für die Haskell Pakete. GitHub dient als zusätzliche Ablage des Quellcodes und der Dokumentation der Entwicklungsschritte. Das Hochladen der Software auf mehreren Plattformen ist nicht zwingend erforderlich.

3.2.1 GitHub

GitHub ist ein webbasierter Hosting-Dienst für Softwareprojekte. [Stü] Er bietet die Möglichkeit Softwareprojekte zu erstellen und zu verwalten. Einem Benutzer ist es möglich mehrere Projekte anzulegen, wobei nur das erste Projekt kostenlos hochgeladen werden kann. Alternative Dienste sind beispielsweise SourceForge oder Bitbucket. Anschließend werden die beiden Begriffe Git und Hub näher erläutert.

Softwareverwaltung - Git

Git ist eine freie Software und ein dezentrales Versionsverwaltungssystem für Softwareprojekte. [gita] Andere bekannte Versionsverwaltungssysteme neben Git sind u. a. Bazaar oder Mercurial.

Ein Software-Projekt wird als “repository” bezeichnet. Durch ein “commit” kann jeder Entwicklungsschritt gespeichert und das Projekt auch wieder zu jedem dieser Versionspunkte zurückgesetzt werden. Innerhalb eines Repositories können auch sogenannte “branches” erzeugt werden und dienen dem Projekt als eine Art Versionsabzweigung. Dadurch besteht die Möglichkeit das Projekt in eine andere Richtung zu entwickeln, z. B. für Testzwecke.

Webschnittstelle - Hub

Hub bedeutet in diesem Kontext, dass Git um eine Webschnittstelle, Wiki und Supportsystem erweitert wird. Durch die graphische Benutzeroberfläche ist es möglich, den Entwicklungsprozess anschaulich darzustellen. Zudem wird mittels GitHub der Funktionsumfang von Git um einige nützliche Funktionen erweitert. Plattformen wie GitHub dienen Entwicklerteams zum Projektmanagement und für die globale Integration.

3.2.2 Hackage

Hackage ist eine Onlinebibliothek zur Verwaltung von in der Programmiersprache Haskell geschriebene Programmpakete. [hac] Nach erfolgter Registrierung als Benutzer können bei entsprechender Einhaltung der Richtlinien eigene Pakete hochgeladen werden. Zum Durchsuchen, Betrachten und Verwenden der mittlerweile über 5000 (Stand 2014) hochgeladenen Pakete ist keine Anmeldung erforderlich.

Ein Paket “package” bezeichnet eine Sammlung von Haskellmodulen und besteht aus folgenden Teilen:

- Beschreibung des Funktionsumfangs
- Lizenzinformationen
- Informationen über den Autor
- herunterladbares Archiv
- Liste der Module im Paket
- optional: Dokumentation (Haddock)

Die verfügbaren Pakete auf dem Hackage Server müssen mit dem Paket- und Build-System Cabal kompatibel sein.

3.3 JavaScript

3.3.1 JavaScript

JavaScript ist plattformunabhängig und eine im Allgemeinen clientseitige, objektorientierte Skriptsprache um **dynamisches** HTML entwickeln zu können. JavaScript wird von der European Computer Manufacturers Association (ECMA) standardisiert, von allen modernen Webbrowsern unterstützt und ist folglich weit verbreitet. [Fla98] Durch JavaScript ist eine Erweiterung von HTML möglich, um beispielsweise auf Benutzereingaben zu reagieren, Inhalte zu verändern oder nachladen zu können.

3.3.2 The JavaScript Problem

Das JavaScript Problem besteht aus zwei Argumenten:

1. **Es ist mühsam mit JavaScript zu entwickeln:**

   > „The depths to which JavaScript sucks are well-documented and well-understood. Its main faults are: lack of module system, weak-typing, verbose function syntax, late binding, which has led to the creation of various static analysis tools to alleviate this language flaw, but with limited success (there is even a static type checker), finicky equality/automatic conversion, this behaviour, and lack of static types.“ [Com]

2. **JavaScript wird benötigt:**

> „Using it for what it is good for, i.e. providing a platform for browser development, but not using the language per se, is therefore desirable, and many are working to achieve this, in varying forms. There are various ways to do it, but we ought to opt for compiling an existing language, Haskell, to JavaScript, because we do not have time to learn or teach other people a new language, garner a new library set and a new type checker and all that Haskell implementations provide.“ [Com]

Zur Lösung dieses Problems und damit zur Vermeidung von JavaScript wurden bereits zwei populäre alternative Programmiersprachen entwickelt:

- CoffeScript
- TypeScript

Für die Verarbeitung von JavaScript Code in der Programmiersprache Haskell existieren ebenfalls bereits mehrere Pakete zur Lösung des JavaScript Problems. Einige davon sind noch nicht sehr weit entwickelt. Da für diese Arbeit ein robustes Werkzeug gesucht wurde, fiel die Wahl für den **Haskell Open Layers Wrapper** auf das Projekt bzw. Paket **Fay**. Zu diesem Projekt wurden zudem bereits zusätzliche Pakete entwickelt, welche für die Webentwicklung sehr nützlich sind (siehe Kapitel 3.6).

3.3.3 JQuery

JQuery ist eine schnelle und kleine Open Source JavaScript Bibliothek mit Funktionen zur Navigation und Manipulation von DOM-Elementen. Mit ihr ist ein einfacherer Zugriff auf das DOM möglich. [jqu] Die erste Version wurde 2006 veröffentlicht und im Jahr 2013 erschien die Version 2.0. Wegen Kompatibilitätsgründen wird die Version 1 noch weiterentwickelt. Für die Entwicklung des olwrapper wurde die Version 2.1.3 verwendet. JQuery besteht aus einer JavaScript Datei, die in das HTML-Dokument eingebunden werden muss.

3.4 OpenLayers

OpenLayers [Fou] ist eine Programmierschnittstelle (API) und JavaScript Bibliothek um Geodaten in Webseiten einzubinden. Eine Entwicklung solcher Webseiten erfolgt auf der Seite des Client und ist unabhängig vom Server. [JA10]

Allgemeiner Aufbau von OpenLayers:

- Map
  - Ansicht (view)
    Zur Bestimmung der visuellen Parameter
  - ein oder mehrere Layer
    Container, der aus den Quellen (siehe unten) bezogenen Daten
  - Bedienelemente (controls)
  - Interaktionen
- Projektion

Die Online-Dokumentation von OpenLayers ist ausführlich beschrieben und nützlich. Zusätzlich zur Dokumentation der Programmierschnittstelle gibt es eine umfangreiche Beispielsammlung auf der Website[6], welche dabei hilft, die Funktionen der OpenLayers Programmbibliothek besser kennenzulernen und zu verstehen. Als Grundversion von OpenLayers wurde am 29. August 2014 die Version 3.0.0 veröffentlicht (OpenLayers 3) und das dazugehörige Repository wurde bereits im August 2012 erstellt[7]. Mittlerweile wurde am 23. Dezember 2014 die Version 3.1.1 veröffentlicht, welche auch in dieser Arbeit verwendet wird. [ope] Zuvor wurde OpenLayers in der Version 2 entwickelt, welche weiterhin verwendet werden kann. Auf die Verwendung sollte wegen der komplett überarbeiteten Neuentwicklung verzichtet werden.[8]

[6] `http://openlayers.org/en/v3.1.1/examples/`
[7] `https://github.com/openlayers/ol3`
[8] `http://openlayers.org/two/`

Mögliche Quellen und Formate bei OpenLayers 3 sind:

- Kachel
- Bilder
- Vektorformate
- Json
  - GeoJson
  - TopoJson
- Text
  - IGC - flight recording file *.igc
  - Polyline
  - WKT - well known text
- XML
  - GMLBase
  - GPX
  - KML
  - OSMXML
  - WFS
  - WMSGetFeatureInfo

Koordinaten und Ausdehnungen müssen bei OpenLayers in Mercator-Projektion (EPSG 3857 - Spherical Mercator) angegeben werden. Liegen Standortdaten in anderen Projektionen vor, müssen diese vor dem Einbinden transformiert werden.

3.5 Snap

Snap ist ein in Haskell geschriebenes Webframework für Unix-Systeme.

Einige Vorteile von Snap sind: [CB11]

- Hohe Leistungsfähigkeit
- Hohe Design-Standards
- Einfachheit und Benutzerfreundlichkeit
- sehr gute Dokumentation
- Robustheit und hohe Testabdeckung

Besondere Merkmale von Snap sind: [CB11]

- schnelle HTTP-Server Bibliothek
- sinnvolle und saubere Monade für die Webprogrammierung
- Aufbau modularer Webanwendungen mittels **Snaplets**
- HTML-basierte Vorlagenerstellung zur Erzeugung von Webseiten

3.5.1 Snaplets

Snaplets sind zusammensetzbare Webanwendungen und sie ermöglichen, einen in sich geschlossenen Teil einer Funktionalität zu entwickeln. Durch Zusammenführen der Snaplets entstehen größere Anwendungen. [sna]

Neben den von den Machern von Snap entwickelten Snaplets **Heist**, Sessions und Auth existieren weitere von unabhängigen Personen entwickelte Snaplets. Einige ausgewählte Snaplets werden auch auf der Projekthomepage von Snap veröffentlicht und gewartet. [sna]

3.5.2 Heist

Heist ist eine HTML-basierte Vorlagen-Engine, um Haskell Funktionen in HTML-Tags einbinden zu können. Heist basiert auf der Idee des Lift Web Framework[9] und kann auch als eigenständige Programmbibliothek verwendet werden, ist dementsprechend ohne Snap frei nutzbar.

Wichtige Funktionen von Heist sind u. a. : [BC]

- Designerfreundliche HTML5 (oder XML) Syntax
- Codeänderungen können sichtbar gemacht werden ohne Haskell-Code neu kompilieren zu müssen
- durch Vorlagen werden redundante Codezeilen vermieden
- Trennung von Präsentation und Code

3.6 Fay

Fay ist eine Haskell Programmbibliothek und JavaScript Alternative bei der Entwicklung von Webseiten. [Cal] Eine mittels Fay entwickelte Applikation wird in weiterer Folge in JavaScript transkompiliert. Dadurch ist es möglich in Haskell definierte Funktionen in Webseiten einzubinden. Die von Fay kompilierte JavaScript Funktionsdatei muss dazu in der HTML-Datei angegeben werden. [BDJP]

Möglich wird das Transkompilieren durch ein **foreign function interface** (FFI). FFI bezeichnet die Schnittstelle, um Funktionen einer Programmiersprache A in der Programmiersprache B zu übersetzen.

[9] `http://liftweb.net/`

Beispiele für nützliche Fay Pakete bzw. Module:

- fay-text: [SB]
  Datentyp Text zur Entwicklung von Fay als JavaScript-String und Text in Haskell.

- fay-jquery: [BVD]
  Verbindung von JQuery und Fay.

- snaplet-fay: [Ber]
  Integration von Fay in Snap. Der besondere Vorteil liegt dabei in der Möglichkeit, während der Entwicklung die Anwendung on-the-fly neukompilieren zu können.

4 Haskell Open Layers Wrapper

4.1 Allgemeines

Der Haskell OpenLayers Wrapper wurde als Webframework mithilfe der Pakete Snap und Fay entwickelt. Ein bedeutender Vorteil ist dabei, dass jede Änderung während der Entwicklung sofort (on-the-fly) im Webbrowser angezeigt werden kann und keine erneute Kompilierung des gesamten olwrapper Pakets notwendig ist. Die Weiterentwicklung von Fay als Snaplet (snaplet-fay) dient dem olwrapper als Werkzeug zum Transkompilieren der OpenLayers Funktionen. Mit Heist als weiteres Snaplet ist eine sehr effektive Trennung von HTML Präsentation (Front-End) und der Programmierung mit Haskell (Back-End) möglich.

Am Ende des Kapitels helfen Szenarien beim Einstieg in das Programm (ähnlich einem Tutorial).

4.2 Vorbereitung der Entwicklungsumgebung

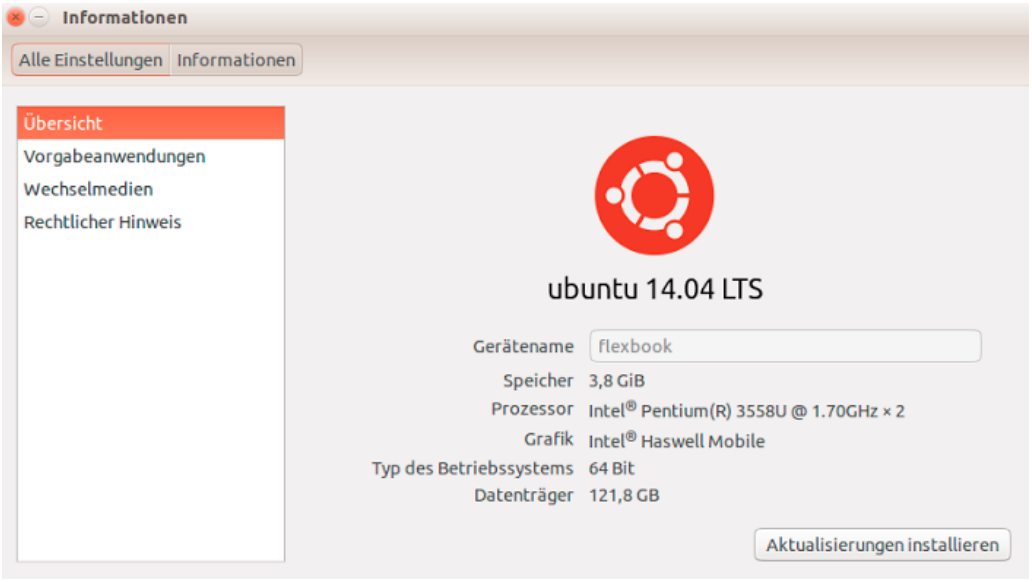
Entwickelt wurde der Haskell OpenLayers Wrapper zur Gänze mit dem Betriebssystem Ubuntu (einen Überblick über die verwendete Entwicklungsumgebung zeigt Abbildung 2 auf S. 26). Als Grundlage dient dem olwrapper die **Haskell Plattform**. Für verschiedene Ubuntu Versionen stehen verifizierte Pakete online zum Download zur Verfügung.[10]

Die Installation der Haskell Plattform und der dazugehörigen Dokumentationsdatenbank erfolgt im Terminal folgendermaßen:

```
sudo apt-get install haskell-platform                1
sudo apt-get install haskell-platform-doc            2
```

[10] z.B. 14.04 Codename Trusty Tahr,
`http://packages.ubuntu.com/trusty/haskell-platform`

Lenovo Flex 2-14



Cabal v1.22.0.0

Haddock v2.13.2.1

Abbildung 2: verwendete Entwicklungsumgebung

Nach der Installation der Haskell Plattform muss zur Umgebungsvariablen PATH von Ubuntu der Installationsordner von Cabal hinzugefügt werden. Dieser ist in der Regel `$HOME/.cabal/bin/`. Im Terminal wird das Hinzufügen durch folgende Befehle dauerhaft im Betriebssystem gespeichert:

```
PATH=\$HOME/.cabal/bin/:\$PATH
export PATH
```

Mithilfe des nun vorhanden Pakets Cabal sollten anschließend folgende Befehle ausgeführt werden, um das System auf den neuesten Stand zu bringen.

```
cabal update
cabal install cabal-install
```

Weitere Entwicklungsprogramme, wie Geany, Git und Firefox (vgl. Tabelle 1 auf S. 3), werden mit folgenden Befehlen installiert:

```
sudo apt-get install geany
sudo apt-get install git
sudo apt-get install firefox
```

4.3 Paket erstellen

Bei der Erstellung eines Haskell Pakets für Hackage müssen die in Kapitel 3.2.2 aufgezählten Regeln eingehalten werden. In erster Linie werden dazu eine Setup- und eine Konfigurationsdatei für Cabal benötigt. Anschließend kann zum Erstellen des Pakets das Programm Cabal verwendet werden.

Folgende Befehle können zur Erstellung eines Pakets im Terminal im Grundverzeichnis des olwrapper ausgeführt werden:

```
cabal configure
cabal sdist
```

Dadurch wird im Verzeichnis "./olwrapper/dist/" das Paket als Archiv bereitgestellt. Des Weiteren wird von Cabal beim Erstellen geprüft, ob das Paket beim Hochladen zu Hackage angenommen werden konnte. Informationen dazu schreibt Cabal beim Erstellen des Pakets in den Terminal.

4.4 Installation des Webframework

Existiert der olwrapper bereits als geziptes Archiv (weil es z. B. von Hackage oder GitHub heruntergeladen wurde), muss dieses zuerst in einen Ordner nach Wahl entpackt werden. Danach muss mit einem Terminal in diesen Ordner und in weiterer Folge in den Ordner "olwrapper" gewechselt werden, damit im Anschluss mit Cabal das Paket installiert werden kann. Cabal erhält aufgrund der Konfigurationsdatei "olwrapper.cabal" die notwendigen Informationen und leitet die Installation ein. Eine ausführbare olwrapper-Datei wird durch die Cabal-Installation in den

Installationsordner von Cabal kopiert und ist somit aufgrund der PATH Variable mit dem Betriebssystem verlinkt.

Der Befehl zum Installieren muss im Terminal im Grundverzeichnis des olwrapper ausgeführt werden:

```
cabal install -f development
```

Der optionale Parameter `-f` gibt den Entwicklungsmodus an; es wird empfohlen diesen auch zu verwenden. Als Flag muss "development" gesetzt sein, damit eine dynamische Entwicklung ohne kompilieren des gesamten Pakets möglich wird. Ohne Angabe des Flag ist eine Entwicklung on-the-fly nicht möglich. Nach der Installation werden Dateien im Verzeichnis "dist" angelegt. Wenn dieses Verzeichnis nicht existiert, wird es automatisch erzeugt.

In "./olwrapper/dist/build/olwrapper/" und im Installationsverzeichnis von Cabal befindet sich nach der Installation jeweils eine ausführbare olwrapper Datei. Da der Pfad zu Cabal der PATH-Variable hinzugefügt wurde, wird es möglich das Webframework aus dem Projektverzeichnis zu starten (siehe Kapitel 4.5.1).

Ausgabe im Terminal am Ende der Installation:

```
...
Linking dist/build/olwrapper/olwrapper ...
Installing executable(s) in /home/user/.cabal/bin
Installed olwrapper-0.x.x
```

4.5 Verwendung des Webframework

4.5.1 Starten

Im Terminal wird der olwrapper in seinem Verzeichnis mit folgendem Befehl gestartet:

```
olwrapper -p 8000
```

Der optionale Parameter `-p` gibt den zu verwendeten Port für das Webframework an. Danach erscheint im Terminal folgende Zeile mit Angabe der URL, auf der die entwickelte Web-GIS-Anwendung erreichbar ist:

```
Listening on http://0.0.0.0:8000/
```
1

In der Fay-Konfigurationsdatei "devel.cfg" existiert zudem ein Parameter (verbose), der die Anzahl der zu generierenden Informationszeilen im Terminal regelt. Bei jedem Laden der Webseite im Browser werden zusätzliche Informationen im Terminal angezeigt (siehe Abbildung 3 auf S. 30). Nach dem Laden des olwrapper werden die Snaplets Heist und Fay initialisiert. Nun entsteht auch die transkompilierte Datei, welche anschließend weiterverwendet werden kann.

Die fertige Webanwendung besteht aus mindestens einer HTML-Datei und aus den drei JavaScript-Bibliotheken, welche local oder per Online-Ressource in die HTML-Datei eingebunden werden müssen:

- index.tpl - HTML-Datei
- ol.js - OpenLayers JavaScript Bibliothek
- jquery.js - JQuery JavaScript Bibliothek
- Index.js - olwrapper JavaScript Bibliothek

Eine Beispiel für eine mit olwrapper entwickelte Webseite zeigt Abbildung 4 auf S. 31 (Detail dazu sind im Tutorial auf S. 41 angeführt).

4.5.2 Beenden

Der olwrapper als Webframework wird im Terminal, in dem er ausgeführt wird, entweder mit der Tastenkombination Strg + c oder durch Schließen des Terminalfensters beendet.

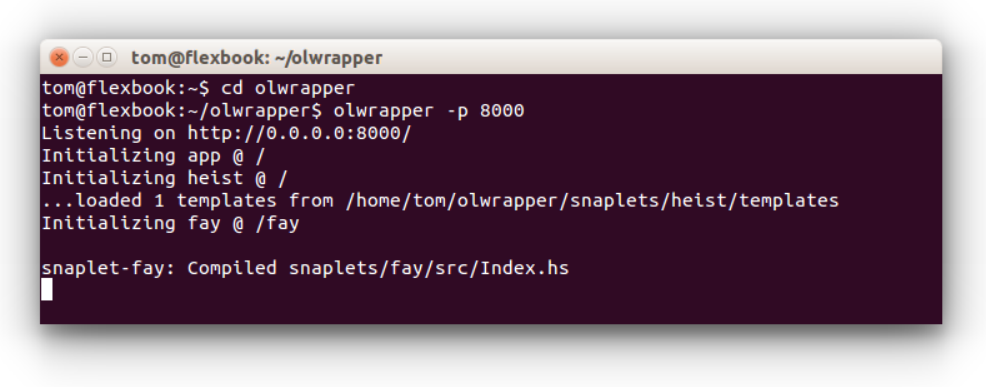


Abbildung 3: Terminalinformationen bei geladenem olwrapper

4.5.3 Bereinigen

Bei Veränderung bestimmter Dateien ist eine Neuinstallation des olwrapper erforderlich. Dies ist vor allem bei der Konfigurationsdatei ("olwrapper.cabal") der Fall. Um diese Änderungen wirksam zu machen, sollten zuvor bereits kompilierte Dateien des olwrapper bereinigt werden. Beim Bereinigen wird das "dist"-Verzeichnis entfernt. Die ausführbare Datei olwrapper im Installationsverzeichnis von Cabal wird dabei nicht gelöscht, jedoch bei jeder Installation überschrieben.

Befehl für den Terminal zum Bereinigen des olwrapper (der Befehl muss im Verzeichnis des olwrapper ausgeführt werden):

```
cabal clean
```
1

4.6 Bekannte Fehlermeldungen

4.6.1 Port besetzt

Der gewünschte Port für den olwrapper ist besetzt und es kann keine neue Verbindung aufgebaut werden.

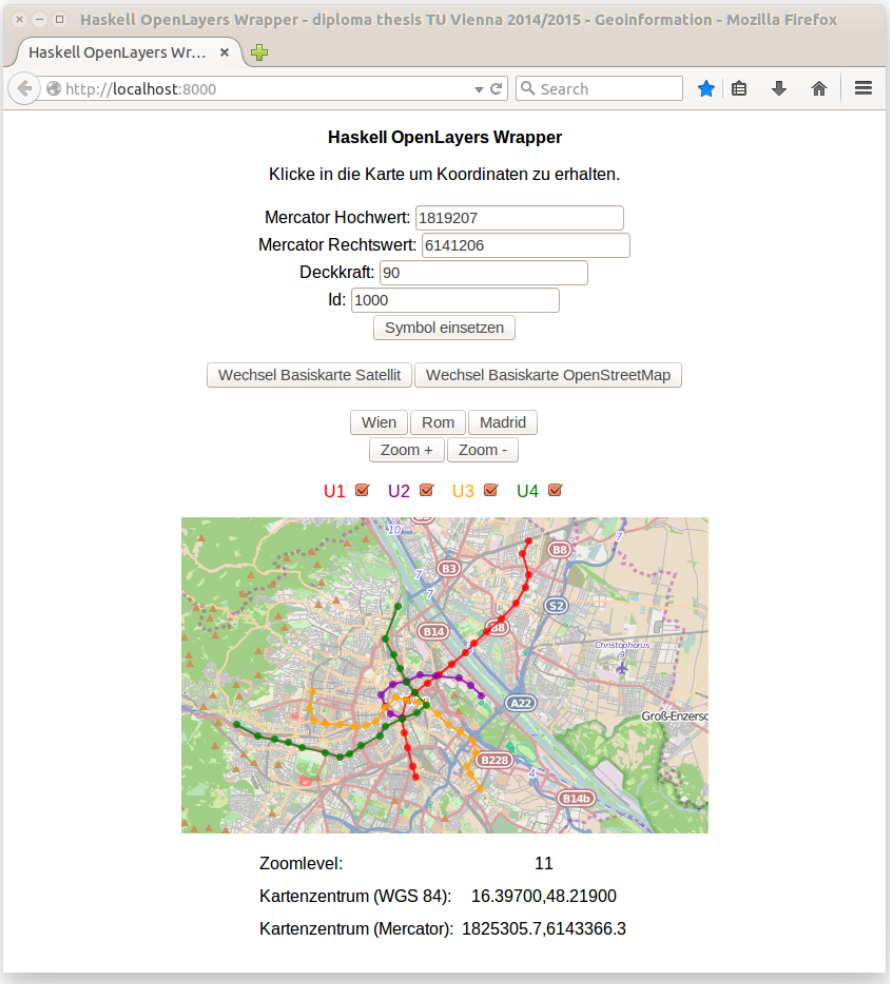


Abbildung 4: Browseransicht für die geladene Webanwendung mittels olwrapper

Dies äußert sich im Terminal unter folgender Ausgabe (z. B. bei Port 8000):

```
Listening on http://0.0.0.0:8000/
Error on startup:
bind: resource busy (Address already in use)
Shutting down...
```

Lösung: Der laufende Prozess auf dem gesuchten Port muss beendet werden. Jedoch sollte der Prozess nur beendet werden, wenn dieser bekannt ist und es sicher ist, ihn zu beenden. Ansonsten sollte ein anderer freier Port beim Starten des

olwrapper benutzt werden.

Befehle zum Suchen und Beenden der besetzen Verbindung am Port (z. B. für Port 8000):

```
lsof -i \:8000                                    1
kill -9 PID                                       2
kill -9 Prozessname                               3
lsof -i :8000                                     4
```

Zeile 1: Anzeige der ID des Prozesses (PID)
Zeile 2: PID einsetzen
Zeile 3: optional
Zeile 4: sollte nun im Terminal nichts ausgeben

4.6.2 Fehler in der Produktion

Probleme und sonstige Fehler, welche während der Entwicklung der Webanwendung auftreten, werden entweder im Terminal, in dem der olwrapper gestartet wurde, oder in der Konsole des Webbrowser angezeigt. Im Browser Firefox kann die Konsole mit der Taste F12 angezeigt werden.

Bei Fehlern in der Haskellprogrammierung kann die Verknüpfung zur transkompilierten JavaScript Datei nicht erstellt werden. Das Snaplet Heist funktioniert in diesem Fall jedoch weiterhin. Folglich werden im Browser nur Inhalte der HTML-Vorlage von Heist angezeigt und ein Zugriff auf die Funktionen bzw. Objekte der Haskell-Programmierung sind nicht möglich.

4.6.3 Fehlende Installation

Nach dem Bereinigen muss vor dem erneuten Starten des olwrapper zuerst eine Neuinstallation durchgeführt werden. Wird nach dem Bereinigen der olwrapper nicht neu installiert, stattdessen der olwrapper gestartet, erscheint im Browser folgende Fehlermeldung:

```
A web handler threw an exception. Details:
GHC error:

can't find a package database at
dist/package.conf.inplace
```

Sowie die Ausgabe im Terminal:

```
Listening on http://0.0.0.0:8000/
olwrapper: GHC error:

can't find a package database at
dist/package.conf.inplace
```

Lösung: olwrapper beenden und zuvor Installation durchführen

4.7 Ordnerstruktur

Die Abbildung 5 zeigt die Ordnerstruktur des olwrapper.

```
./olwrapper/
            LICENSE
            olwrapper.cabal
            README.md
            Setup.hs
            log/
                access.log
                error.log
            snaplets/
                     fay/
                         devel.cfg
                         js/
                            Index.js
                         src/
                             Index.hs
                     heist/
                          templates/
                                    index.tpl
            src/
                Application.hs
                Main.hs
                Site.hs
            static/
            web/
                OlApp.hs
                Tutorial/
                         OlApp.hs
                         Traffic.hs
            wrapper/
                    OpenLayers.hs
                    OpenLayers/
                               Func.hs
                               Html.hs
                               Internal.hs
                               Types.hs
```

Abbildung 5: Ordnerstruktur des olwrapper

4.8 Erläuterungen zur Ordnerstruktur

Im folgenden Kapitel werden die Verzeichnisse und Dateien des olwrapper in Bezug auf ihre Funktionsweise und Art beschrieben.

LICENSE: Diese Datei beinhaltet die Lizenzinformationen zum Projekt und ist ebenso als Parameter in der Konfigurationsdatei olwrapper.cabal eingetragen. Als Lizenz wird die **BSD3** (Berkeley Software Distribution 3-Clause License) verwendet.

README.md: Mithilfe der Auszeichnungssprache Markdown[11] wird diese Datei dazu verwendet, für die Softwareverwaltungsplattform GitHub eine ausgestaltete Projektseite mit den wichtigen Informationen anbieten zu können.

log-Verzeichnis: Im Verzeichnis "log/" werden in der Regel zwei Dateien automatisch erzeugt, die die Zugriffe und sonstige Meldungen des Webframework dokumentieren. Wenn das Verzeichnis nicht existiert, sollte es vor dem Starten des Webframework erzeugt werden.

Anmerkung: Die folgende Dateien und Verzeichnisse dürfen für die Entwicklung der Web-GIS-Anwendung geändert werden.

static-Verzeichnis: Im Verzeichnis "static/" können Dateien für den statischen Zugriff der Webseite hinzugefügt und in weiterer Folge in die Heist-Vorlagen eingebunden werden. Das können u. a. CSS-Dateien oder auch diverse andere JavaScript-Bibliotheken sein.

src-Verzeichnis: Die Webframework Dateien von Snap (Application.hs, Main.hs und Site.hs) wurden dahingehend verändert, sodass die Snaplets **auth** und **session** entfernt wurden und zusätzlich Fay als Snaplet hinzugefügt wurde. Dazu wurden die Codezeilen des Projekts snaplet-fay übernommen.

snaplets/fay/devel.cfg - Fay Konfiguration: In dieser von **Fay** entwickelten Konfigurationsdatei werden die Parameter für Fay festgelegt. Änderungen in dieser Datei werden erst nach einem Neustart[12] des olwrapper wirksam.

[11] `https://daringfireball.net/projects/markdown/`
[12] Anmerkung: ohne einem Bereinigen

Die Parameter in der Konfigurationsdatei von Fay sind:

- compileMode (Development/Production):
  - Development:
    Bei jedem Ladevorgang der Webseite im Browser wird die Datei Index.js neu kompiliert.
  - Production:
    Für die statische Ausführung empfohlen, wenn auch olwrapper nicht mehr im Entwicklungsmodus installiert wurde[13].
    Alle Haskell Module in
    ./olwrapper/snaplets/fay/src/ werden in
    ./olwrapper/snaplets/fay/js transkompiliert.

- verbose (on/off):
  Gibt an, ob ausführlichere Informationen von Fay im Terminal ausgegeben werden sollen.

- prettyPrint (on/off):
  js-beautify für eine verschönerte Formatierung der erzeugten JavaScript-Dateien verwenden.

- includeDirs (wrapper/src,snaplets/fay/ol):
  Verzeichnisliste mit Modulen durch Beistriche getrennt. Dadurch werden die Module in den angegebenen Verzeichnissen mit Fay verknüpft. Das Verzeichnis snaplets/fay/src wird von Fay automatisch auf Module untersucht und muss nicht angegeben werden.

- packages (fay-text,fay-jquery):
  Zu importierende Fay Pakete. Die Liste wird durch Beistriche getrennt.

snaplets/heist/templates/index.tpl: Es handelt sich hierbei um eine Heist-konforme Vorlagendatei im HTML-Format, welche als Startdatei für die Entwicklung der Webseite dient. Die Datei kann beliebig erweitert werden und folgt den Richtlinien für die Entwicklung mit Heist [BC].

[13] Anmerkung: cabal install -f development

Sie beinhaltet in dieser Version zudem für den olwrapper das HTML-Element zur Verknüpfung mit der Kartenanwendung:

```
<div id="map"></div>
```
1

Mit dem Parameter “id” des div-Elements wird die **targetId** des Moduls OlApp.hs verknüpft. Das Vorlagensystem von Heist kann auch dazu verwendet werden, um die Kartenanwendung in eine andere Vorlagendatei auszulagern. Die Funktionen von Heist werden durch den olwrapper nicht eingeschränkt.

Es müssen für den vollen Funktionsumfang des olwrapper folgende JavaScript Bibliotheken eingebunden werden:

- JQuery
- JavaScript OpenLayers 3 Bibliothek
- Index.js

Das Einbinden in dieser HTML-Datei kann über das **static-Verzeichnis** oder Webseiten erfolgen, mit Ausnahme der Index.js, welche immer ins Verzeichnis “./olwrapper/snaplets/fay/js” kompiliert wird und mit folgender Codezeile eingebunden werden muss:

```
<script src="/fay/Index.js"/>
```
1

web/OlApp.hs: In dieser Datei wird die Kartenanwendung entwickelt und die Verknüpfung zum HTML-Element hergestellt. Sie hängt über die Funktion **designMap** und dem Parameter **targetId** mit dem Modul OpenLayers.hs und der Heist-Vorlage index.tpl zusammen. Für größere Projekte ist es von Vorteil Deklarationen in zusätzlichen Haskell-Modulen auszulagern (siehe Tutorial bzw. Szenarien auf S. 42 und 43 sowie im Modul Traffic.hs).

web/Tutorial/: Details zu den Inhalten der Tutorialdateien sind im Kapitel 4.9 ab S. 41 angeführt.

snaplets/fay/js/Index.js: Hierbei handelt sich um eine automatisch generierte JavaScript-Bibliothek, in der die Haskell-Funktionen des olwrapper aufgrund der Datei Index.hs transkompiliert werden, so dass eine funktionierende Webanwendung möglich wird. Die Datei wird bei jedem Kompilieren überschrieben und muss in die passende HTML-Datei eingebunden werden. Wie oft die Datei neu kompiliert wird, kann in der Fay-Konfigurationsdatei devel.cfg mit dem Paramter "compileMode" angegeben werden [Development, Production].

snaplets/fay/src/Index.hs: Als Vorlage diente bei der Entwicklung dieses Moduls die Index.hs des Pakets snaplet-fay und wurde für den olwrapper adaptiert. In dieser Datei können auch Webanwendungen unabhängig von der Entwicklung für die Kartenanwendung erstellt werden. Sie verknüpft die Anweisungen in OlApp.hs mit dem Modul OpenLayers.hs über die Funktion **olwrapperLoad**.

olwrapper.cabal - Cabal Konfiguration: Diese Datei ist für die Zuordnung des olwrapper als Cabal-Paket notwendig. Dadurch wird es möglich den olwrapper mit Cabal installieren und auf Hackage hochladen zu können. Sie beinhaltet u. a. Informationen über das Projekt, Verfasser, Lizenzinformationen, Versionsnummer, Angaben der zu verwendeten Ordnerstruktur sowie Angabe der Abhängigkeiten zu anderen Haskell-Paketen und/oder Haskell-Modulen. Der Inhalt wurde überwiegend aus dem Paket snaplet-fay übernommen. Soll die zu entwickelnde Anwendung um weitere Pakete erweitert werden, ist die Angabe der zusätzlichen Paketabhängigkeiten in dieser Datei erforderlich.

Sie beinhaltet folgende Sektionen:

- Informationen zum Paket (u.a. Name, Version, Beschreibung, Lizenz)
- Angaben zur Ordner- und Dateistruktur
- Angaben für Installationsparamter (Flags)
- Definition des Pakets (library)
- Definition des Webframework (executable)
- Link zur Softwareverwaltung (z. B. auf GitHub)

Um den Funktionsumfang von einzelnen Haskell Paketen und Modulen zu erweitern, können weitere Parameter hinzugefügt werden.

Setup.hs: Neben der Cabal-Konfigurationsdatei dient diese Setupdatei zur Erstellung eines Hackage-konformen Pakets (siehe Kapitel 4.3).

Wichtig: Die folgenden Dateien sollen nur bei der Weiterentwicklung des Haskell OpenLayers Wrapper und besonders in Bezug auf die Erweiterung der OpenLayers JavaScript-Bibliothek verändert werden.

wrapper/OpenLayers.hs: Das Modul OpenLayers.hs ist eine Art ‚Mutterschiff' für die Funktionsweise des Haskell OpenLayers Wrapper. Darin wird das Kartenobjekt initialisiert und sie dient als Verbindung zwischen OpenLayers, Snap sowie den Snaplets Heist und Fay.

Wichtige Funktionen sind:

- Definition der Funktion **olwrapperLoad** für das Modul Index.hs
- Setzen der **targetId** (definiert in OlApp.hs und index.tpl)
- Ausführung der in OlApp.hs definierten Funktion **designMap**

wrapper/OpenLayers.Func.hs: Als ‚Hauptschlagader' des olwrapper kann diese Datei bezeichnet werden. In ihr sind alle die OpenLayers JavaScript-Bibliothek betreffenden Funktionen definiert. Mithilfe von Fay und dem Modul FFI von Fay werden die JavaScript-Funktionen in Haskell transformiert und in weiterer Folge bei Ausführung in die Index.js Datei transkompiliert.

Zur Übersichtlichkeit sind die Funktionen in dieser Datei in folgende Bereiche unterteilt:

- new ... Objekte erstellen
- add ... Objekte hinzufügen
- remove ... Objekte entfernen
- change ... Objekte ändern
- set ... Parameter von Objekten setzen
- get ... Parameter von Objekten erhalten

- transform ... Koordinaten transformieren
- other ... zusätzliche Hilfsfunktionen

wrapper/OpenLayers.Html.hs: Dieses Modul beinhaltet alle Funktionen in Bezug auf die HTML-Programmierung. Also all jenen Funktionen, welche nicht der Transformation für die OpenLayers JavaScript Bibliothek entspringen und hauptsächlich für die Verknüpfung der olwrapper-Funktionen mit den HTML-Elementen notwendig sind. Durch solch eine Funktion wird es beispielsweise möglich eine Checkbox zu erzeugen und in weiterer Folge mit einem Kartenobjekt zu verknüpfen.

wrapper/OpenLayers.Internal.hs: Hier sollen Funktionen definiert werden, welche für alle Module zur Verfügung stehen sollten.

wrapper/OpenLayers.Types.hs: Zur Trennung der Funktionen der OpenLayers Bibliothek sind in dieser Datei die Definitionen für zusätzliche Datentypen bzw. Typkonstruktoren ausgelagert. Weiters ist die Funktion zur Auswahl der Basiskarte (MapSource) in dieser Datei definiert.

Die implementierten Basiskarten sind:

- Sat ... MapQuest Bild
- Osm ... MapQuest OpenStreetMap
- Hyb ... MapQuest Hybrid
- OSM ... OpenStreetMap

4.9 Implementierung der Szenarien und Tutorial

4.9.1 Allgemeines

Als Grundlage für das initialisierende Hello-Map-Programm[14] beinhalten die Dateien "OlApp.hs" und "index.tpl" den minimalen Code wie in Abbildung 6 und 7 dargestellt.

```
module OlApp where

import            OpenLayers.Func
import            OpenLayers.Types
import            OpenLayers.Internal

targetId = "map"
designMap = OpenLayers.Internal.void $ do
    addBaseLayer Osm
```

Abbildung 6: Beispiel für einen Minimalcode von OlApp.hs

Hinweise zu Abbildung 6: Die targetId muss mit der "id" in "index.tpl" übereinstimmen und es muss die Funktion "designMap" vorhanden sein. Geodaten werden mit einem Basislayer (`addBaseLayer`) sichtbar.

```
<!DOCTYPE html>
<html>
<head>
  <script src="http://openlayers.org/en/v3.1.1/build/ol.js"/>
  <script src="/static/jquery.min.2.1.3.js"/>
  <script src="/fay/Index.js"/>
</head>
<body>Hello Map
  <div id="map"/>
</body>
</html>
```

Abbildung 7: Beispiel für einen Minimalcode von index.tpl

Hinweise zu Abbildung 7: Es ist auch möglich die JavaScript-Bibliotheken von OpenLayers und JQuery lokal oder von anderen Quellen einzubinden. Wichtig ist, dass eine aktuelle Version bereitgestellt wird. In der fertig entwickelten Webanwendung besteht auch die Möglichkeit, die kompilierte "Index.js" aus anderen Quellen

[14] Anmerkung: Als Anlehnung an Hello-World-Programm

importieren zu lassen.

Um anschließend die geforderten Szenarien zu implementieren, wird der minimale Code der beiden Dateien “OlApp.hs” und “index.tpl” folgendermaßen erweitert.

4.9.2 Szenario 1 - öffentlichen Verkehr anzeigen

- HTML erweitern - index.tpl:
  - Neue Tabelle als Container für die neuen Elemente definieren:

```
<table>
  <tr>
    <td id="table1"/>
  </tr>
</table>
```

- Variablen deklarieren - OlApp.hs:
  - ID für das neue HTML-Element

```
visiblecheckboxU1 = "opacitycheckboxU1"
```

  - z. B. U-Bahn-Stationen definieren und verbinden - `u1` (siehe ./olwrapper/web/Tutorial/Traffic.hs [15]):

- Neuen Layer hinzufügen und Index merken - OlApp.hs:
  - Basiskarte = Index 0, nächster neuer Layer = Index 0+1

```
addStyledFeatures u1 defaultopacity
```

- Neue Checkbox erstellen und mit dem neuen Layer verknüpfen - OlApp.hs:
  - Checkbox in Tabelle erstellen:

```
addCheckbox visiblecheckboxU1 "table1" "U1"
```

  - Checkbox und Layer per Index verknüpfen:

```
addOlDomInput visiblecheckboxU1 "checked"
    "visible" (getLayerByIndex 1) -- Index 1
```

Abbildung 8 zeigt einen Ausschnitt aus dem Browser für das erste Szenario.

[15] Quelle der Koordinaten der U-Bahn-Stationen: `http://www.pocketnavigation.de`

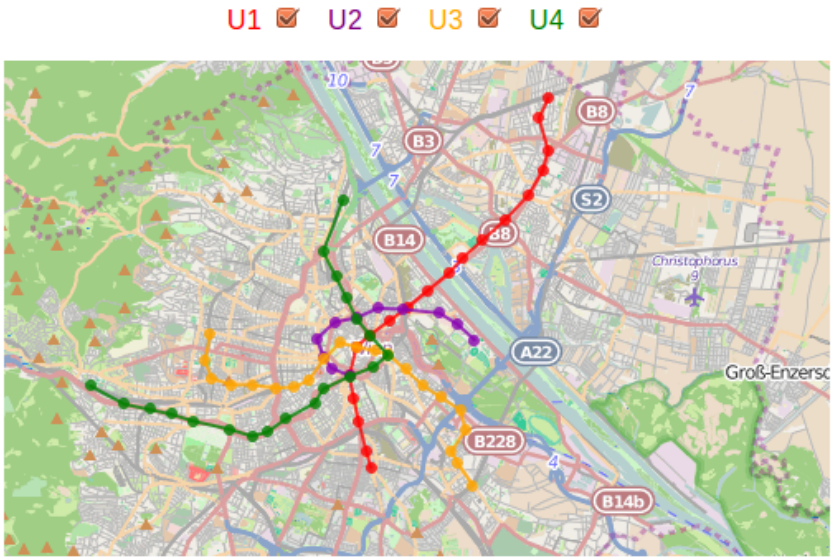


Abbildung 8: eingeschaltene U-Bahn-Linien und dazugehörige Checkboxen

4.9.3 Szenario 2 - Kundenstandorte erzeugen

- HTML erweitern - index.tpl:
  - Neuen Container für die neuen Elemente definieren:

```
<div id="mapdesc"></div>
```

- Variablen deklarieren - OlApp.hs:
  - Eigenen Punktstil für die neuen Punkte definieren:

```
mypointstyle = GeoPointStyle 4
    "green" "black" 2
```

  - IDs für die neuen HTML-Elemente:

```
descId          = "mapdesc"
formId          = "forminput"
hochinputId     = "xinput"
rechtsinputId   = "yinput"
opacityinputId = "opacinput"
idinputId       = "idinputId"
```

- Funktionen ausführen - OlApp.hs:
  - Form in den Container einfügen

```
addForm descId formId
```

  - Textfelder für die Koordinateneingabe, Deckkraft und ID erstellen

```
addInput "Mercator Hochwert:"
    formId hochinputId     "1819207"
addInput "Mercator Rechtswert:"
    formId rechtsinputId   "6141206"
addInput "Deckkraft:"
    formId opacityinputId  "90"
addInput "Id:"
    formId idinputId       "1000"
```

  - Schaltfläche hinzufügen, die den neuen Punkt als Layer importieren soll

```
addButton "Symbol einsetzen" descId
    (addPointFromLabels hochinputId
        rechtsinputId opacityinputId
        idinputId mypointstyle)
```

Abbildung 9 zeigt einen Ausschnitt aus dem Browser für das zweite Szenario.

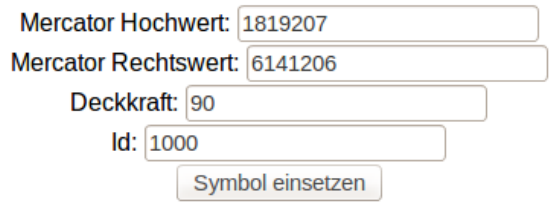


Abbildung 9: mit Vorgabewerten gefüllte Textfelder sowie Schaltfläche zum Einfügen eines Punkts

4.9.4 Tutorial

Im Verzeichnispfad ".olwrapper/web/Tutorial/" befinden sich zwei Dateien, welche als Beispiel für eine entwickelte Web-GIS-Anwendung dienen. Die Anwendung kombiniert die beiden Szenarien und zusätzlich werden die Parameter Zentrum und Zoomlevel der Karte am unteren Rand angezeigt und nach Änderung der Kartenansicht automatisch aktualisiert (vgl. Abbildung 4). Als weitere Funktion

können in der Karte Koordinaten abgegriffen und in einem Pop-up-Fenster angezeigt werden. Zusätzliche Schaltflächen verdeutlichen die Funktionsweise zwischen HTML-Elementen und der Karte.

5 Conclusio

5.1 Allgemeines

OpenLayers bietet mit der JavaScript-Programmbibliothek ein sehr umfangreiches Werkzeug zum Erzeugen von Web-GIS-Anwendungen. In dieser Arbeit konnte nur ein kleiner Teil des Ganzen für Haskell entwickelt werden. Der Grundstein für eine Fortführung zu einer Bibliothek mit größerem Funktionsumfang ist gelegt.

Das JavaScript-Problem kann für OpenLayers mit dem olwrapper umgangen werden. Als Vergleich zwischen einer mit auschließlich JavaScript und einer identen mit Haskell entwickelten Kartenanwendungen zeigen die Quellcodeunterschiede auf den Seiten 49 f. auf. Implementiert wurde dafür das Minimalbeispiel zum Anzeigen einer Hello-Map-Webseite.

Der Fokus lag in dieser Arbeit auf der Frage, ob und wie es möglich ist, die OpenLayers Funktionen mit Haskell bedienen zu können. Da dies positiv beantwortet werden konnte, wäre eine Forschung in Hinblick auf Performance und eine Weiterentwicklung wünschenswert.

In dieser Version dient der olwrapper als reine Vorlage und es kann durch Herunterladen und Entpacken des Pakets schnell eine eigene Kartenanwendung erstellt werden. Ziel ist es jedoch ein Paket anzubieten, welches ein Initialisierungsfunktion enthält, welche selbstständig die notwendigen Dateien und Ordnerstruktur in einem neuen Projektverzeichnis anlegt. Im Paket snaplet-fay steht z. B. die Funktion “init” zur Verfügung. Eine ähnliche Funktionsweise sollte angedacht werden.

Weiters ist es meiner Meinung nach interessant zu untersuchen, wie sich der Unterschied einer olwrapper-Anwendung und einer identen OpenLayers-JavaScript-Anwendung in Hinblick auf die Ladezeiten auswirkt. Im Tutorial besitzt die transkompilierte JavaScript-Bibliothek Index.js beispielsweise 7000 Zeilen Quellcode und dies führt noch zu keinen merkbaren Geschwindigkeitseinbußen beim Betrieb der Webseite. Ein Test mit einer umfangreicheren Kartenanwendung könnte Aufschluß über die Performance des olwrapper geben.

Ein weiterer Aspekt für die Zukunft ist sicherlich, eine effizientere Entwicklungsumgebung für die Entwicklung von Webanwendungen mit Haskell bereitzustellen.

Durch Integration der Entwicklungswerkzeuge des Browsers kann ein großer Nutzen während der Programmierung vorhergesagt werden. Für die Entwicklungsumgebung “Eclipse” existiert für Haskell bereits ein Integrationswerkzeug[16]. Es war mir jedoch nicht möglich, ein funktionierendes Eclipse-Projekt zum Laufen zu bringen, welches mir subjektiv mehr Vorteile als mit der Entwicklung mit Geany verschafft hätte. Ein großes Hindernis bei der verwendeten Entwicklungsumgebung war das Auffinden von Fehlern aufgrund des Fehlens eines Debuggers. So war es notwendig, die Informationen zu den Fehlern gleichzeitig im Browser und Terminal zu vergleichen und zu analysieren.

Bislang können mit dem olwrapper bis zu vier verschiedene Basiskarten sowie Layer als Punkte und Linien eingefügt werden. OpenLayers bietet eine Vielzahl an Möglichkeiten, Vektor- und Rasterformate in die Kartenanwendung zu integrieren. Daher wäre eine Weiterentwicklung des olwrapper, in Hinblick auf die Unterstützung mehrerer Formate, förderlich.

[16] `http://eclipsefp.github.io/`

5.2 Vergleich JavaScript - Haskell

In diesem Vergleich anhand der Hello-Map-Webseite ist die Trennung zwischen der Präsentation und Kartendefinition bei der Lösung mit Haskell besonders gut erkennbar. Zusätzlich anzumerken ist die Reduktion der Klammersetzungen im Vergleich zu JavaScript.

Das Verhältnis der Codezeilen zur Definition der Karte ist zwischen den beiden Varianten 3 zu 12. Selbstverständlich könnte auch der ganze Code in einer Zeile stehen, dieses Argument soll jedoch aufgrund gewisser Designrichtlinien beim Programmieren hier nicht greifen.

Auf der anderen Seite ist es aber mit JavaScript schneller möglich das Hello-Map-Programm zu entwickeln. In diesem Vergleich wäre die Entwicklung mit JavaScript die ‚Gewinnerin'.

	Klammern	Zeilen	Entwicklungszeit
Haskell	1	3 (Z: 7-9)	o
JavaScript	10	12 (Z: 10-21)	+

Tabelle 3: Vergleich Haskell-JavaScript im Hello-Map-Programm

Klammern ersetzt mittels Funktion:

```
($) :: (a -> b) -> a -> b
f $ x = f x
```

5.2.1 Haskell

Programm

```
module OlApp where

import            OpenLayers.Func
import            OpenLayers.Types
import            OpenLayers.Internal

targetId = "map"
designMap = OpenLayers.Internal.void $ do
    addBaseLayer Osm
```

HTML-Vorlage

```
<!DOCTYPE html>
<html>
<head>
  <script src="http://openlayers.org/en/v3.1.1/build/ol.js"/>
  <script src="/static/jquery.min.2.1.3.js"/>
  <script src="/fay/Index.js"/>
</head>
<body>Hello Map
  <div id="map"/>
</body>
</html>
```

5.2.2 JavaScript

HTML-Datei

```
<!DOCTYPE html>
<html>
  <head>
    <script src="http://openlayers.org/en/v3.1.1/build/ol.js"
            type="text/javascript"></script>
  </head>
  <body>Hello Map
    <div id="map"/>
    <script type="text/javascript">
      var map = new ol.Map({
        target: 'map',
        layers: [
          new ol.layer.Tile({
            source: new ol.source.MapQuest({layer: 'osm'})
          })
        ],
        view: new ol.View({
            center: [0,0],
            zoom:       3,
        })
      });
    </script>
  </body>
</html>
```

AJAX Asynchronous JavaScript und XML ist eine Technik zur asynchronen Datenübertragung zwischen Client und Server. Sie ermöglicht den Austausch von Teilen einer Webseite, so dass dabei nicht die ganze Webseite neu geladen und aufgebaut werden muss.

Auszeichnungssprache Maschinenlesbare Sprache für die Gliederung und Formatierung von Texten und anderen Daten.

Bezeichner Ein eindeutiger Identifikator für ein Objekt.

Compiler Programm, welches ein anderes Programm, das in einer bestimmten Programmiersprache geschrieben ist, zur Ausführung am Computer übersetzt.

Copyleft Die Veränderung des Werkes ist nur dann erlaubt, wenn alle Änderungen unter den gleichen Lizenzbedingungen weitergegeben werden. Im Gegensatz zu copyright wird das Recht zum Kopieren eingeräumt.

Crowdsourcing Auslagerung einzelner Aufgaben oder Projekten an eine Gruppe freiwilliger Nutzer über das Internet. Das Paradebeispiele hierfür ist Wikipedia.

CSS (cascading style sheet) Gestaltungssprache für elektronische Dokumente, um die Darstellungsvorgaben von den Inhalten zu trennen. Sie kann innerhalb des Dokuments sowie in einer separaten Datei definiert werden.

Debian `http://www.debian.org/` Gemeinschaftlich entwickeltes, freies Betriebssystem mit Linux Kernel.

Document Object Model (DOM) Vom W3C definierte Schnittstellenbeschreibung um einen einheitlichen Zugriff auf HTML- oder XML-Dokumente zu ermöglichen.

Engine In der Informationstechnologie ein eigenständiger Teil eines Computerprogramms zur Berechnung komplexer Strukturen. Sie läuft meist selbstständig im Hintergrund.

EPSG-Code Weltweit eindeutiger 4- bis 5-stellige Code für Koordinatenreferenzsysteme oder Transformationen.

Extensible Markup Language (XML) Allgemeine Auszeichnungssprache zur strukturierten Darstellung von Daten in Textform. Es können beliebig viele Dokumenttypen und in Folge beliebig viele Strukturelemente für jeden Dokumenttyp festgelegt werden.

Flag Statusindikator oder binäre Variable, welche gesetzt, gelöscht oder ausgelesen werden kann.

Free Software Foundation (FSF) `http://www.fsf.org/`
Stiftung um freie Software zu fördern.

Freie Software `https://www.gnu.org/` Software, die die Freiheit und Gemeinschaft der Nutzer respektiert. Ganz allgemein bedeutet das, dass Nutzer die Freiheit haben Software auszuführen, zu kopieren, zu verbreiten, zu untersuchen, zu ändern und zu verbessern.

Funktionale Programmierung Verfeinerung des deklarativen Programmierparadigma, bei dem Programme ausschließlich aus Funktionen bestehen und keine veränderliche Variablen existieren.

GeoJson `http://geojson.org/` Offenes Datenformat für geographische Daten und Informationen unter Verwendung der JavaScript Object Notation.

GIS Dateiformat Standardisierte Datenformate, welche geographische Informationen beinhalten. Bekannte GIS-Datenformate sind u. a. ECW, GeoTIFF, Shapefile, DXF, GeoJson, GML)

Glasgow Haskell Compiler (GHC) Ein quelloffener Compiler für die Programmiersprache Haskell.

Hypertext Markup Language (HTML) Zusammen mit DOM und CSS einer der Kernsprachen des World Wide Web. Eine textbasierte Auszeichnungssprache zur Strukturierung von digitalen Inhalten in elektronischen Dokumenten.

Internet Engineering Task Force (IETF) `https://www.ietf.org/`
Eine Organisation, die sich mit der technischen Weiterentwicklung des Internets befasst, um dessen Funktionsweise zu verbessern.

JavaScript Object Notation (Json) Textbasiertes Datenformat zum Datenaustausch zwischen Anwendungen in Verbindung mit JavaScript oder als Ersatz für XML um beispielsweise Speicherplatz zu sparen.

Monade Ein abstrakter Datentyp. Dient der funktionalen Programmierung als Hilfsmittel zur Verkettung von Operationen und als Wrapper zu anderen Datentypen. Besteht aus den Komponenten Typkonstruktor, Einheitsfunktion (return) sowie mindestens einer weiteren Operation (bind).

Open Data Commons (ODC) `http://opendatacommons.org/`
Ein Projekt der Open Knowledge Foundation (OKF, Stiftung für offenes Wissen) um einen rechtlichen Rahmen für freie Daten zu schaffen und Lizenzen für freie Datenbanken zu pflegen.

Open Database License (ODbL) Freie Datenbanklizenz des ODC mit starkem Copyleft, welche u. a. von OpenStreetMap (OSM, siehe Kapitel GIS) verwendet wird.

Open Geospatial Consortium (OGC) Eine gemeinnützige Organisation für die Entwicklung von Standards für die raumbezogene Informationsverwaltung zur Gewährleistung von Interoperabilität.

Open Source Initiative (OSI) `http://opensource.org/`
Eine Organisation zur Förderung von Open Source Software.

Open Source Software Software, welche unter den Bedingungen der OSI lizenziert wurde. Der Programmcode steht als Quelltext zur Verfügung, die Software darf beliebig oft kopiert, verbreitet, genutzt, verändert und in veränderter Form weiterverwendet werden.

plattformunabhängig Ein Programm ist plattformunabhängig, wenn es auf diversen Plattformen ausgeführt werden kann. Webanwendungen sind häufig auf verschiedenen Browsern und auf verschiedenen Plattformen lauffähig.

Programmbibliothek Sammlung von Unterprogrammen, Routinen oder Funktionen. Sie bildet keine eigenständig lauffähige Einheit.

Programmierparadigma Grundlegendes Konzept einer Programmiersprache. Es wird zwischen folgenden zwei wichtigen Konzepten unterschieden: imperative und deklarative Programmiersprachen.

Bei imperativen Programmiersprachen (e.g. Fortran, Java, C, Pascal) besteht das Programm aus einzelnen Anweisungen, die nacheinander abgearbeitet werden und den Ablauf beschreiben, "was wie zu tun ist".
Deklarative Programmiersprachen (e.g. Haskell, Prolog, Lisp) richten ihr Augenmerk auf das "was", das Programm tun soll und nicht wie, wodurch die Beschreibung des Problems in den Vordergrund rückt.

Programmierschnittstelle (application programming interface)
Eine Schnittstelle zur Anwendungsprogrammierung und zur Anbindung an andere Programme. Inkludiert meist auch die Dokumentation der Schnittstellenfunktionen.

Syntaktischer Zucker Alternative Schreibweisen zur Vereinfachung und besseren Lesbarkeit des Quelltextes bei Programmiersprachen.

transkompilieren Beim Transkompilieren ist es möglich, das Programm in einer anderen Programmiersprache zu schreiben als das Endprodukt. Der Transcompiler übersetzt den Quellcode in die andere Programmiersprache.

Webframework (Web Application Framework) Eine Software zur Entwicklung von Webseiten oder Webservices. Mittels vordefinierten und vorgefertigten Klassen können schnell lauffähige Webanwendungen entwickelt werden.

Website vs. Webseite Als Website wird der gesamte Webauftritt bezeichnet mit eindeutigen Namen (Domain) wie z.B. www.test.at. Die Webseiten sind die einzelnen Seiten bzw. HTML-Dokumente auf der Website.

Webtechnologie Implementierte Methoden und Verfahren, die für die Entwicklung und Anwendung von Systemen, die im World Wide Web genutzt werden, die Grundlage bilden (z.B. HTML, CSS, JavaScript).

Wiki Ein System für Websiten, mit welchem unbegrenzt Personen ein bestimmtes Ziel durch Sammlung und Bereitstellung von Informationen oder Wissen erreichen können.

World Wide Web Consortium (W3C) `http://www.w3.org/`
Gremium zur Standardisierung der Techniken im Wold Wide Web.

Wrapper Schnittstelle um Systeme miteinander zu verbinden bzw. zu verknüpfen. Ein Software-Wrapper beinhaltet Software, die im neuen System funktionsfähig ist.

Zur Erstellung des Glossar wurden folgende Websiten (besucht: Dezember 2015) verwendet:

- `http://www.itwissen.info/`
- `http://wirtschaftslexikon.gabler.de/`
- `http://www.wikipedia.org/`

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur

[BC] Doug Beardsley and Gregory Collins. Heist Haskell Paket. `http://hackage.haskell.org/package/heist`. besucht: 20. Jänner 2015.

[BDJP] Adam Bergmark, Chris Done, David Johnson, and Conrad Parker. Fay GitHub Wiki. `https://github.com/faylang/fay/wiki`. besucht: 20. Jänner 2015.

[Ber] Adam Bergmark. Snaplet-Fay Haskell Paket. `https://hackage.haskell.org/package/snaplet-fay`. besucht: 20. Jänner 2015.

[BVD] Adam Bergmark, Brian Victor, and Chris Done. Fay-JQuery Haskell Paket. `http://hackage.haskell.org/package/fay-jquery`. besucht: 20. Jänner 2015.

[Cal] Paul Callaghan. Webprogrammierung mit Haskell. `https://pragprog.com/magazines/2012-12/web-programming-in-haskell`. besucht: 20. Jänner 2015.

[CB11] Gregory Collins and Doug Beardsley. The Snap Framework: A Web Toolkit for Haskell. *Internet Computing, IEEE*, 15(1):84–87, 2011.

[Com] HaskellWiki Community. The Javascript Problem. `https://www.haskell.org/haskellwiki/The_JavaScript_Problem`. besucht: 20. Jänner 2015.

[CPJS08] Duncan Coutts, Isaac Potoczny-Jones, and Don Stewart. Haskell: Batteries Included. In *Proceedings of the First ACM SIGPLAN Symposium on Haskell*, Haskell '08, pages 125–126, New York, NY, USA, 2008. ACM.

[Dun] Geoff Duncan. Is Russia's Yandex beating Google at its own game? `http://www.digitaltrends.com/mobile/is-russias-yandex-beating-google-at-its-own-gam/`. besucht: 20. Jänner 2015.

[Fla98] David Flanagan. *JavaScript: The Definitive Guide*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 3rd edition, 1998.

[Fou] Open Source Geospatial Foundation. Openlayers Webpräsenz. `http://openlayers.org/`. besucht: 20. Jänner 2015.

[gita] Git User's Manual. `https://www.kernel.org/pub/software/scm/git/docs/user-manual.html`. besucht: 20. Jänner 2015.

[gitb] GitHub. `https://github.com/`. besucht: 20. Jänner 2015.

[hac] Hackage. `https://hackage.haskell.org/`. besucht: 20. Jänner 2015.

[has] The Haskell Platform. `https://www.haskell.org/platform/`. besucht: 20. Jänner 2015.

[JA10] Marc Jansen and Till Adams. *OpenLayers: Webentwicklung mit dynamischen Karten und Geodaten*. Professional reference. Open Source Press, 2010.

[Jon02] Simon Peyton Jones. *Haskell 98 Language and Libraries: The Revised Report*. `http://haskell.org/`, 2002.

[jqu] JQuery. `http://jquery.com/`. besucht: 20. Jänner 2015.

[map] `http://www.maps.net/`. besucht: 20. Jänner 2015.

[MW] Simon Marlow and David Wearn. Haddock User Guide. `https://www.haskell.org/haddock/doc/html/index.html`. besucht: 20. Jänner 2015.

[ope] Openlayers GitHub Releases. `https://github.com/openlayers/ol3/releases`. besucht: 20. Jänner 2015.

[SB] Michael Snoyman and Adam Bergmark. Fay-Text Haskell Paket. `https://hackage.haskell.org/package/fay-text`. besucht: 20. Jänner 2015.

[sna] What Are Snaplets? `http://snapframework.com/snaplets`. besucht: 20. Jänner 2015.

[Stü] Moritz Stückler. Was ist eigentlich dieses GitHub? `http://t3n.de/news/eigentlich-github-472886/`. besucht: 20. Jänner 2015.

[vEU10] Jan van Eijck and Christina Unger. *Computational Semantics with Functional Programming*. Cambridge University Press, New York, NY, USA, 1st edition, 2010.

[VMG+01] K. Virrantaus, J. Markkula, A. Garmash, V. Terziyan, J. Veijalainen, A. Katanosov, and H. Tirri. Developing gis-supported location-based services. In *Web Information Systems Engineering, 2001. Proceedings of the Second International Conference on*, volume 2, pages 66–75, 2001.

[ZL05] Chuanrong Zhang and Weidong Li. The roles of web feature and web map services in real-time geospatial data sharing for time-critical applications. *Cartography and Geographic Information Science*, 32(4):269–283, 2005.

ANHANG

Anmerkungen zur Lizenz:

Die für die Erstellung des **Haskell Open Layers Wrapper** besonders wichtigen und verwendeten Pakete und Projekte sollen hier nochmal mit ihren entsprechenden Lizenzarten (Lizenztextevorlagen siehe Fußnoten) angeführt werden:

- JQuery - MIT
- Fay-Text - MIT
- OpenLayers - BSD2
- Snaplet-Fay - BSD3
- Fay-JQuery - BSD3

Anmerkungen zu den Kommentaren:

Alle Kommentare in den verschiedenen Dateitypen werden im Anhang rot dargestellt. Haskell-Module beinhalten Kommentare im Sinne der Haddock Richtlinien zur Erstellung der Haddock Dokumentation.

In der Datei "snaplets/fay/devel.cfg" enthaltene Kommentare stammen von den Entwicklern des Pakets Snaplet-Fay und wurden aus diesem Paket übernommen.

Die HTML-Datei "snaplets/heist/templates/index.tpl" gliedert sich in zwei Abschnitte. Im oberen Teil ist der Minimalcode (siehe Abbildung 7) für ein Hello-Map-Programm auskommentiert, darunter befindet sich der Code, damit das Tutorial funktioniert.

MIT: `http://opensource.org/licenses/mit-license.html`
BSD2: `http://opensource.org/licenses/BSD-2-Clause`
BSD3: `http://opensource.org/licenses/BSD-3-Clause`

olwrapper.cabal

```
Name:           olwrapper                                                                                  1
Version:        0.4.1                                                                                      2
Synopsis:       An OpenLayers JavaScript Wrapper and Webframework with snaplet-fay                         3
Description:    Haskell OpenLayers Wrapper                                                                 4
                .                                                                                          5
                * the project has development status                                                       6
                .                                                                                          7
                * the library is also the tutorial                                                         8
                .                                                                                          9
                * for a full documentation please read my diploma thesis (TU Vienna) - fully available in summer 2015 10
Author:         Thomas Edelmann                                                                            11
Maintainer:     tomnobleman@gmail.com                                                                      12
Stability:      Experimental                                                                               13
Category:       Web, Map, OpenLayers, Webframework, snaplet-fay                                            14
Build-Type:     Simple                                                                                     15
Cabal-version:  >=1.12                                                                                     16
license:        GPL                                                                                        17
license-file:   LICENSE                                                                                    18

extra-source-files:                                                                                        20
    LICENSE                                                                                                21
    README.md                                                                                              22
    snaplets/fay/devel.cfg                                                                                 23
    snaplets/heist/templates/index.tpl                                                                     24
    static/jquery.min.2.1.3.js                                                                             25
    static/ol.3.1.1.js                                                                                     26
    snaplets/fay/src/Index.hs                                                                              27
    src/Application.hs                                                                                     28
    src/Main.hs                                                                                            29
    src/Site.hs                                                                                            30

Flag development                                                                                           32
    Description:  Whether to build the server in development (interpreted) mode                            33
    Default:      False                                                                                    34

library                                                                                                    36
    hs-source-dirs:      wrapper, web                                                                      37
    default-language:    Haskell2010                                                                       38
    exposed-modules:                                                                                       39
        OpenLayers                                                                                         40
        OpenLayers.Func                                                                                    41
        OpenLayers.Html                                                                                    42
        OpenLayers.Internal                                                                                43
        OpenLayers.Types                                                                                   44
        OlApp                                                                                              45
    other-modules:                                                                                         46
        Tutorial.OlApp                                                                                     47
        Tutorial.Traffic                                                                                   48
    other-extensions:                                                                                      49
    build-depends:                                                                                         50
        base                      >= 4 && < 4.8,                                                           51
        bytestring                >= 0.10      ,                                                           52
        lens                      == 4.4.0.2   ,                                                           53
        mtl                       >= 2         ,                                                           54
        text                      >= 0.11      ,                                                           55
        fay                       == 0.21.2    ,                                                           56
        fay-text                  == 0.3.2     ,                                                           57
        fay-jquery                == 0.6.0.2   ,                                                           58
        snap                      == 0.13.3.1  ,                                                           59
        snap-core                 == 0.9.6.3   ,                                                           60
        snap-server               == 0.9.4.5   ,                                                           61
        snaplet-fay               >= 0.3       ,                                                           62
        snap-loader-static        == 0.9.0.2   ,                                                           63
        snap-loader-dynamic       == 0.10.0.2                                                              64

executable olwrapper                                                                                       66
```

```
    hs-source-dirs:      src, snaplets/fay/src
    main-is:             Main.hs
    default-language:    Haskell2010
    build-depends:
        base                     ,
        bytestring               ,
        lens                     ,
        mtl                      ,
        text                     ,
        fay                      ,
        fay-text                 ,
        fay-jquery               ,
        snap                     ,
        snap-core                ,
        snap-server              ,
        snaplet-fay              ,
        snap-loader-static

    if flag(development)
        build-depends:  snap-loader-dynamic == 0.10.0.2
        cpp-options:    -DDEVELOPMENT
        ghc-options:    -threaded -w -Wall
    else
        if impl(ghc >= 6.12.0)
            ghc-options:
                   -threaded -Wall -fwarn-tabs -funbox-strict-fields
                   -fno-warn-orphans -fno-warn-unused-do-bind
        else
            ghc-options:
                   -threaded -Wall -fwarn-tabs -funbox-strict-fields
                   -fno-warn-orphans

source-repository head
    Type:       git
    Location:   https://github.com/olwrapper/olwrapper.git
```

LICENSE

```
Copyright (c) 2015, Thomas Edelmann
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
    * Redistributions of source code must retain the above copyright
    notice, this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright
    notice, this list of conditions and the following disclaimer in the
    documentation and/or other materials provided with the distribution.
    * Neither the name of the <organization> nor the
    names of its contributors may be used to endorse or promote products
    derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL <COPYRIGHT HOLDER> BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

README.md

```
Haskell OpenLayers Wrapper
==========================

the project has development status

the library is also the tutorial

for a full documentation please read my diploma thesis 2015 (TU Vienna)
[Geoinformation](http://www.geoinfo.tuwien.ac.at/)
```

Setup.hs

```
    import Distribution.Simple

    main = defaultMain
```

snaplets/fay/src/Index.hs

```
{-|
Module      :  Snaplet Fay Module
Description :  module fay handle olwrapper application onload of a website
-}
module Index where

import           OpenLayers

main :: Fay ()
main = olwrapperAddOnLoad olwrapperLoad
```

snaplets/fay/devel.cfg

```
# "Development": Compile files on every request.
# "Production":  Compile all files in snaplets/src/fay when the
#                application is started and then serve them statically.
# Both methods will write files to disk for debugging purposes.
# Default is "Development".
compileMode = "Development"
# Print more or less information about what the snaplet is doing.
# Default is on.
verbose = on
# Run js-beautify on the results if it is available in the PATH.
# If it isn't available the compiled file will be used as is.
# Default is on.
prettyPrint = on
# A comma-separated list of directories to look for fay imports in.
# This is useful if you want to share modules between Fay and Snap.
# Paths are relative to the root of the snap application.
# snaplets/fay/src will always be checked for imports.
# Default is "" meaning no additional directories will be checked.
includeDirs = "wrapper,web"
# A comma separated list of Fay package names.
# This gives the same result as using includeDirs,
# except Fay figures out the paths for you.
# Default is "" meaning no other Fay packages will be used.
packages = "fay-text,fay-jquery"
```

snaplets/heist/templates/index.tpl

```
<!DOCTYPE html> <!-- minimum code
<html>
    <head>
        <script src="http://openlayers.org/en/v3.1.1/build/ol.js"/>
        <script src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"/>
        <script src="/fay/Index.js"/>
    </head>
    <body>Hello Map
        <div id="map"/>
    </body>
</html>
--> <!-- tutorial -->
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
        <title>Haskell OpenLayers Wrapper - diploma thesis TU Vienna 2015 - Geoinformation</title>
        <script src="http://openlayers.org/en/v3.1.1/build/ol.js"/>
        <script src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"/>
        <script src="/fay/Index.js"/>
        <style type="text/css">
            * { font-family: Arial, Verdana, Helvetica, sans-serif; }
            #map {width: 500px; height: 300px; }
        </style>
    </head>
    <body>
        <center>
            <p><b>Haskell OpenLayers Wrapper</b></p>
            <div id="mapdesc"></div>
            <table style="padding: 12px;">
                <tr>
                    <td id="table1" style="color: red;    padding-right: 12px"/>
                    <td id="table2" style="color: purple; padding-right: 12px"/>
                    <td id="table3" style="color: orange; padding-right: 12px"/>
                    <td id="table4" style="color: green"/>
                </tr>
            </table>
            <div id="map"></div>
            <table style="padding: 12px;">
                <tr>
                    <td>Zoomlevel:</td>
                    <td>
                        <div id="zoomlabel"     align="center" style="padding: 4px"></div>
                    </td>
                </tr>
                <tr>
                    <td>Kartenzentrum (WGS 84):</td>
                    <td>
                        <div id="wgslabel"      align="center" style="padding: 4px"></div>
                    </td>
                </tr>
                <tr>
                    <td>Kartenzentrum (Mercator):</td>
                    <td>
                        <div id="mercatorlabel" align="center" style="padding: 4px"></div>
                    </td>
                </tr>
            </table>
        </center>
    </body>
</html>
```

src/Application.hs

```
{-# LANGUAGE NoImplicitPrelude #-}
{-# LANGUAGE TemplateHaskell   #-}

module Application where

import          Control.Lens
import          Snap.Snaplet
import          Snap.Snaplet.Heist
import          Snap.Snaplet.Fay

data App = App {
       _heist :: Snaplet (Heist App)
     , _fay   :: Snaplet Fay
       }

makeLenses ''App

instance HasHeist App where heistLens = subSnaplet heist

type AppHandler = Handler App App
```

src/Main.hs

```
{-# LANGUAGE CPP               #-}
{-# LANGUAGE TemplateHaskell   #-}

module Main where

import          Control.Exception (SomeException, try)
import          Data.Text
import          Snap.Http.Server
import          Snap.Snaplet
import          Snap.Snaplet.Config
import          Snap.Core
import          System.IO
import          Site

#ifdef DEVELOPMENT
import           Snap.Loader.Dynamic
#else
import           Snap.Loader.Static
#endif

getConf :: IO (Config Snap AppConfig)
getConf = commandLineAppConfig defaultConfig

getActions :: Config Snap AppConfig -> IO (Snap (), IO ())
getActions conf = do
    (msgs, site, cleanup) <- runSnaplet (appEnvironment =<< getOther conf) app
    hPutStrLn stderr $ unpack msgs
    return (site, cleanup)

main :: IO ()
main = do
    (conf, site, cleanup) <- $(loadSnapTH [| getConf |] 'getActions ["snaplets/heist/templates"])
    _ <- try $ httpServe conf site :: IO (Either SomeException ())
    cleanup
```

src/Site.hs

```
{-# LANGUAGE OverloadedStrings #-}
{-# LANGUAGE PackageImports #-}

module Site ( app ) where

import          Data.ByteString     (ByteString)
import          Snap.Snaplet
import          Snap.Snaplet.Fay
import          Snap.Snaplet.Heist
import          Snap.Util.FileServe
import          Application

routes :: [(ByteString, AppHandler ())]
routes = [
           ("/fay",    with fay fayServe)
         , ("/static", serveDirectory "static")
         ]

app :: SnapletInit App App
app = makeSnaplet "app" "" Nothing $ do
    h <- nestSnaplet "" heist ( heistInit "templates" )
    f <- nestSnaplet "fay" fay initFay
    addRoutes routes
    return $ App h f
```

web/OlApp.hs

```
{-|
Module      :  OpenLayers Wrapper Webdevelopment
Description :  to create a web application
-}
module OlApp where

import           Prelude hiding (void)
import           OpenLayers.Func
import           OpenLayers.Types
import           OpenLayers.Internal
import           Tutorial.OlApp (designTutorialMap)
import           Fay.FFI

-- | ID to combine the HTML-Element map with the OpenLayers object
targetId = "map"
-- | definition of the behaviour for OpenLayers
designMap :: Fay ()
designMap = void $ do
    addBaseLayer OSM
    designTutorialMap
```

web/Tutorial/OlApp.hs

```
module Tutorial.OlApp where

import           Prelude hiding (void)
import           OpenLayers.Func
import           OpenLayers.Types
import           OpenLayers.Html
import           OpenLayers.Internal
import           Tutorial.Traffic
import           Fay.FFI

zoomlevel = 11
defaultopacity = Opacity 80

coor1 =  Coordinate  16.369   48.196 wgs84proj
coor2 =  Coordinate  16.370   48.195 wgs84proj
coor3 =  Coordinate  16.371   48.194 wgs84proj
vienna = Coordinate  16.397   48.219 wgs84proj
rome   = Coordinate  12.5     41.9   wgs84proj
bern   = Coordinate   7.4458  46.95  wgs84proj
madrid = Coordinate (-3.6833) 40.4   wgs84proj

-- Styles
mypointstyle = GeoPointStyle 4 "green" "black" 2
mylinestyle  = GeoLineStyle  "green" 3

-- Features
mypoint = GeoPoint vienna                301 mypointstyle
myline  = GeoLine  [coor1, coor2, coor3] 101 mylinestyle

-- IDs for the html-tags

descId = "mapdesc"
formId = "forminput"
hochinputId = "xinput"
rechtsinputId = "yinput"
opacityinputId = "opacinput"
idinputId = "idinputId"

zoomlabel = "zoomlabel"
wgslabel = "wgslabel"
mercatorlabel = "mercatorlabel"
visiblecheckboxU1 = "opacitycheckboxU1"
visiblecheckboxU2 = "opacitycheckboxU2"
visiblecheckboxU3 = "opacitycheckboxU3"
visiblecheckboxU4 = "opacitycheckboxU4"

designTutorialMap :: Fay ()
designTutorialMap = void $ do
    -- baselayer loaded before
    -- init center and zoom
    setCenter vienna
    setZoom zoomlevel
    --  ADD LAYERS
    --  LAYER Index 1      --  add only one feature
    addStyledFeature myline  $ Opacity 0    --  hiding with Opacity 0
    --  LAYER Index 2      --  add only one feature
    addStyledFeature mypoint $ Opacity 0    --  hiding with Opacity 0
    --  LAYER Index 3      --  or add more with a list
    addStyledFeatures u1 defaultopacity
    --  LAYER Index 4
    addStyledFeatures u2 defaultopacity
    --  LAYER Index 5
    addStyledFeatures u3 defaultopacity
    --  LAYER Index 6
    addStyledFeatures u4 defaultopacity
```

```
--                                                                                                                    66
--  CLICK EVENT                                                                                                       67
addSingleClickEventAlertCoo "EPSG:4326"                                                                               68
addDiv descId "div0" "Klicke in die Karte, um Koordinaten zu erhalten."                                               69
addBreakline descId                                                                                                   70
--                                                                                                                    71
--  ADD INPUT POINT                                                                                                   72
----  create an input form to be able to type a coordinate to insert a pointfeature                                   73
addForm descId formId                                                                                                 74
addInput "Mercator Hochwert: " formId hochinputId "1819207"                                                           75
addInput "Mercator Rechtswert: " formId rechtsinputId "6141206"                                                       76
addInput "Deckkraft: " formId opacityinputId "90"                                                                     77
addInput "Id: " formId idinputId "1000"                                                                               78
----  add the button to insert the pointfeature                                                                       79
addButton "Symbol einsetzen" descId $ addPointFromLabels hochinputId rechtsinputId opacityinputId idinputId           80
     mypointstyle
addBreakline descId                                                                                                   81
--                                                                                                                    82
--  CHANGE BASELAYER                                                                                                  83
----  change your baselayer at Index 0                                                                                84
addBreakline descId                                                                                                   85
addButton "Wechsel Basiskarte Satellit" descId $ changeBaseLayer Sat                                                  86
addButton "Wechsel Basiskarte OpenStreetMap" descId $ changeBaseLayer OSM                                             87
addBreakline descId                                                                                                   88
--                                                                                                                    89
--  BUTTONS WITH DIFFERENT METHODS                                                                                    90
----  add buttons to navigate to specific location and zoomlevel                                                      91
addBreakline descId                                                                                                   92
addButton "Wien"   descId $ setCenter vienna                                                                          93
addButton "Rom"    descId $ setCenterZoom rome zoomlevel                                                              94
addButton "Madrid" descId $ setCenterZoom madrid zoomlevel                                                            95
addBreakline descId                                                                                                   96
--  add button for other tools                                                                                        97
addButton "Zoom +" descId $ zoomIn  1                                                                                 98
addButton "Zoom -" descId $ zoomOut 1                                                                                 99
addBreakline descId                                                                                                   100
--                                                                                                                    101
--  BINDINGS                                                                                                          102
----  create some checkbox for binding                                                                                103
addCheckbox visiblecheckboxU1 "table1" "U1 "                                                                          104
addCheckbox visiblecheckboxU2 "table2" "U2 "                                                                          105
addCheckbox visiblecheckboxU3 "table3" "U3 "                                                                          106
addCheckbox visiblecheckboxU4 "table4" "U4 "                                                                          107
----  bind the html-element checkbox with a layer                                                                     108
addOlDomInput visiblecheckboxU1 "checked" "visible" $ getLayerByIndex 3                                               109
addOlDomInput visiblecheckboxU2 "checked" "visible" $ getLayerByIndex 4                                               110
addOlDomInput visiblecheckboxU3 "checked" "visible" $ getLayerByIndex 5                                               111
addOlDomInput visiblecheckboxU4 "checked" "visible" $ getLayerByIndex 6                                               112
--                                                                                                                    113
--  EVENT LABELS                                                                                                      114
----  add some labels to display map events                                                                           115
addElement zoomlabel     "<div>" "zl1"                                                                                116
addElement wgslabel      "<div>" "wl1"                                                                                117
addElement mercatorlabel "<div>" "ml1"                                                                                118
----  at moveend of the map add the center and zoomlevel of the map to the label                                      119
addMapWindowEvent "moveend" $ setEventToHtml zoomlabel getZoom                                                        120
addMapWindowEvent "moveend" $ setEventToHtml wgslabel $ getCenter wgs84proj 5                                         121
addMapWindowEvent "moveend" $ setEventToHtml mercatorlabel $ getCenter mercatorproj 1                                 122
--                                                                                                                    123
```

web/Tutorial/Traffic.hs

```
module Tutorial.Traffic where

import           OpenLayers.Types

uStationSize = 3
uLineSize = 2

u1Color = "red"
u1StationStyle = GeoPointStyle uStationSize u1Color u1Color 1
u1LineStyle = GeoLineStyle u1Color uLineSize
u2Color = "#8904B1"
u2StationStyle = GeoPointStyle uStationSize u2Color u2Color 1
u2LineStyle = GeoLineStyle u2Color uLineSize
u3Color = "orange"
u3StationStyle = GeoPointStyle uStationSize u3Color u3Color 1
u3LineStyle = GeoLineStyle u3Color uLineSize
u4Color = "green"
u4StationStyle = GeoPointStyle uStationSize u4Color u4Color 1
u4LineStyle = GeoLineStyle u4Color uLineSize

u4_01 = Coordinate 16.26139 48.19750 wgs84proj
u4_02 = Coordinate 16.27528 48.19250 wgs84proj
u4_03 = Coordinate 16.28639 48.19111 wgs84proj
u4_04 = Coordinate 16.29473 48.18972 wgs84proj
u4_05 = Coordinate 16.30389 48.18778 wgs84proj
u4_06 = Coordinate 16.31890 48.18555 wgs84proj
u4_07 = Coordinate 16.32834 48.18361 wgs84proj
u4_08 = Coordinate 16.33472 48.18472 wgs84proj
u4_09 = Coordinate 16.34250 48.18834 wgs84proj
u4_10 = Coordinate 16.35445 48.19278 wgs84proj
u4_11 = Coordinate 16.35806 48.19667 wgs84proj
u4_12 = Coordinate 16.36889 48.20015 wgs84proj
u4_13 = Coordinate 16.37889 48.20278 wgs84proj
u4_14 = Coordinate 16.38500 48.20612 wgs84proj
u4_15 = Coordinate 16.37750 48.21150 wgs84proj
u4_16 = Coordinate 16.37195 48.21611 wgs84proj
u4_17 = Coordinate 16.36751 48.22194 wgs84proj
u4_18 = Coordinate 16.36361 48.22778 wgs84proj
u4_19 = Coordinate 16.35806 48.23472 wgs84proj
u4_20 = Coordinate 16.36612 48.24888 wgs84proj

u4 = [
    GeoPoint u4_01 40001 u4StationStyle,
    GeoPoint u4_02 40002 u4StationStyle,
    GeoPoint u4_03 40003 u4StationStyle,
    GeoPoint u4_04 40004 u4StationStyle,
    GeoPoint u4_05 40005 u4StationStyle,
    GeoPoint u4_06 40006 u4StationStyle,
    GeoPoint u4_07 40007 u4StationStyle,
    GeoPoint u4_08 40008 u4StationStyle,
    GeoPoint u4_09 40009 u4StationStyle,
    GeoPoint u4_10 40010 u4StationStyle,
    GeoPoint u4_11 40011 u4StationStyle,
    GeoPoint u4_12 40012 u4StationStyle,
    GeoPoint u4_13 40013 u4StationStyle,
    GeoPoint u4_14 40014 u4StationStyle,
    GeoPoint u4_15 40015 u4StationStyle,
    GeoPoint u4_16 40016 u4StationStyle,
    GeoPoint u4_17 40017 u4StationStyle,
    GeoPoint u4_18 40018 u4StationStyle,
    GeoPoint u4_19 40019 u4StationStyle,
    GeoPoint u4_20 40020 u4StationStyle,
    GeoLine [u4_01, u4_02, u4_03, u4_04, u4_05, u4_06, u4_07, u4_08, u4_09, u4_10,
            u4_11, u4_12, u4_13, u4_14, u4_15, u4_16, u4_17, u4_18, u4_19, u4_20] 41001 u4LineStyle
    ]
```

```
u3_01 = Coordinate 16.42024 48.17001 wgs84proj
u3_02 = Coordinate 16.41389 48.17611 wgs84proj
u3_03 = Coordinate 16.41139 48.17944 wgs84proj
u3_04 = Coordinate 16.41722 48.18528 wgs84proj
u3_05 = Coordinate 16.41500 48.19084 wgs84proj
u3_06 = Coordinate 16.40667 48.19445 wgs84proj
u3_07 = Coordinate 16.39972 48.19750 wgs84proj
u3_08 = Coordinate 16.39223 48.20222 wgs84proj
u3_09 = Coordinate 16.38500 48.20639 wgs84proj
u3_10 = Coordinate 16.37917 48.20750 wgs84proj
u3_11 = Coordinate 16.37222 48.20833 wgs84proj
u3_12 = Coordinate 16.36500 48.20944 wgs84proj
u3_13 = Coordinate 16.35834 48.20501 wgs84proj
u3_14 = Coordinate 16.35222 48.19917 wgs84proj
u3_15 = Coordinate 16.34611 48.19722 wgs84proj
u3_16 = Coordinate 16.33861 48.19695 wgs84proj
u3_17 = Coordinate 16.32861 48.19778 wgs84proj
u3_18 = Coordinate 16.31972 48.19778 wgs84proj
u3_19 = Coordinate 16.31167 48.19972 wgs84proj
u3_20 = Coordinate 16.30890 48.20444 wgs84proj
u3_21 = Coordinate 16.31111 48.21195 wgs84proj

u3 = [
    GeoPoint u3_01 30001 u3StationStyle,
    GeoPoint u3_02 30002 u3StationStyle,
    GeoPoint u3_03 30003 u3StationStyle,
    GeoPoint u3_04 30004 u3StationStyle,
    GeoPoint u3_05 30005 u3StationStyle,
    GeoPoint u3_06 30006 u3StationStyle,
    GeoPoint u3_07 30007 u3StationStyle,
    GeoPoint u3_08 30008 u3StationStyle,
    GeoPoint u3_09 30009 u3StationStyle,
    GeoPoint u3_10 30010 u3StationStyle,
    GeoPoint u3_11 30011 u3StationStyle,
    GeoPoint u3_12 30012 u3StationStyle,
    GeoPoint u3_13 30013 u3StationStyle,
    GeoPoint u3_14 30014 u3StationStyle,
    GeoPoint u3_15 30015 u3StationStyle,
    GeoPoint u3_16 30016 u3StationStyle,
    GeoPoint u3_17 30017 u3StationStyle,
    GeoPoint u3_18 30018 u3StationStyle,
    GeoPoint u3_19 30019 u3StationStyle,
    GeoPoint u3_20 30020 u3StationStyle,
    GeoPoint u3_21 30021 u3StationStyle,
    GeoLine [u3_01, u3_02, u3_03, u3_04, u3_05, u3_06, u3_07, u3_08, u3_09, u3_10,
            u3_11, u3_12, u3_13, u3_14, u3_15, u3_16, u3_17, u3_18, u3_19, u3_20, u3_21] 31001 u3LineStyle
    ]

u2_01 = Coordinate 16.36910 48.20028 wgs84proj
u2_02 = Coordinate 16.36139 48.20251 wgs84proj
u2_03 = Coordinate 16.35833 48.20556 wgs84proj
u2_04 = Coordinate 16.35528 48.21056 wgs84proj
u2_05 = Coordinate 16.36305 48.21528 wgs84proj
u2_06 = Coordinate 16.37195 48.21639 wgs84proj
u2_07 = Coordinate 16.38074 48.21908 wgs84proj
u2_08 = Coordinate 16.39133 48.21875 wgs84proj
u2_09 = Coordinate 16.40640 48.21770 wgs84proj
u2_10 = Coordinate 16.41376 48.21466 wgs84proj
u2_11 = Coordinate 16.42095 48.21030 wgs84proj

u2 = [
    GeoPoint u2_01 20001 u2StationStyle,
    GeoPoint u2_02 20002 u2StationStyle,
    GeoPoint u2_03 20003 u2StationStyle,
    GeoPoint u2_04 20004 u2StationStyle,
    GeoPoint u2_05 20005 u2StationStyle,
    GeoPoint u2_06 20006 u2StationStyle,
    GeoPoint u2_07 20007 u2StationStyle,
    GeoPoint u2_08 20008 u2StationStyle,
```

```
    GeoPoint u2_09 20009 u2StationStyle,
    GeoPoint u2_10 20010 u2StationStyle,
    GeoPoint u2_11 20011 u2StationStyle,
    GeoLine [u2_01, u2_02, u2_03, u2_04, u2_05, u2_06, u2_07, u2_08, u2_09, u2_10,
            u2_11] 21001 u2LineStyle
    ]


u1_01 = Coordinate 16.37778 48.17500 wgs84proj
u1_02 = Coordinate 16.37612 48.17945 wgs84proj
u1_03 = Coordinate 16.37278 48.18751 wgs84proj
u1_04 = Coordinate 16.37056 48.19389 wgs84proj
u1_05 = Coordinate 16.36889 48.20028 wgs84proj
u1_06 = Coordinate 16.37194 48.20836 wgs84proj
u1_07 = Coordinate 16.37750 48.21167 wgs84proj
u1_08 = Coordinate 16.38583 48.21556 wgs84proj
u1_09 = Coordinate 16.39333 48.21944 wgs84proj
u1_10 = Coordinate 16.40139 48.22390 wgs84proj
u1_11 = Coordinate 16.41055 48.22889 wgs84proj
u1_13 = Coordinate 16.41611 48.23305 wgs84proj
u1_14 = Coordinate 16.42436 48.23811 wgs84proj
u1_15 = Coordinate 16.43390 48.24362 wgs84proj
u1_16 = Coordinate 16.44333 48.25041 wgs84proj
u1_17 = Coordinate 16.44922 48.25702 wgs84proj
u1_18 = Coordinate 16.45158 48.26277 wgs84proj
u1_19 = Coordinate 16.44773 48.27164 wgs84proj
u1_20 = Coordinate 16.45164 48.27719 wgs84proj

u1 = [
    GeoPoint u1_01 10001 u1StationStyle,
    GeoPoint u1_02 10002 u1StationStyle,
    GeoPoint u1_03 10003 u1StationStyle,
    GeoPoint u1_04 10004 u1StationStyle,
    GeoPoint u1_05 10005 u1StationStyle,
    GeoPoint u1_06 10006 u1StationStyle,
    GeoPoint u1_07 10007 u1StationStyle,
    GeoPoint u1_08 10008 u1StationStyle,
    GeoPoint u1_09 10009 u1StationStyle,
    GeoPoint u1_10 10010 u1StationStyle,
    GeoPoint u1_11 10011 u1StationStyle,
    GeoPoint u1_13 10013 u1StationStyle,
    GeoPoint u1_14 10014 u1StationStyle,
    GeoPoint u1_15 10015 u1StationStyle,
    GeoPoint u1_16 10016 u1StationStyle,
    GeoPoint u1_17 10017 u1StationStyle,
    GeoPoint u1_18 10018 u1StationStyle,
    GeoPoint u1_19 10019 u1StationStyle,
    GeoPoint u1_20 10020 u1StationStyle,
    GeoLine [u1_01, u1_02, u1_03, u1_04, u1_05, u1_06, u1_07, u1_08, u1_09, u1_10,
            u1_11, u1_13, u1_14, u1_15, u1_16, u1_17, u1_18, u1_19, u1_20] 11001 u1LineStyle
    ]
```

wrapper/OpenLayers.hs

```
{-|                                                                                                                    1
Module      :  OpenLayers Wrapper mothership                                                                           2
Description :  this module combines OpenLayers with Fay                                                                3
-}                                                                                                                     4
module OpenLayers where                                                                                                5

import           Prelude hiding (void)                                                                                 7
import           Fay.FFI                                                                                               8
import           OlApp (targetId, designMap)                                                                           9
import           OpenLayers.Internal                                                                                   10
-- | to add OpenLayers Wrapper when the page event load is registered                                                  11
olwrapperAddOnLoad :: Fay f -> Fay ()                                                                                  12
olwrapperAddOnLoad = ffi "window.addEventListener(\"load\", %1)"                                                       13
-- | to add a default OpenLayers map object with name \"olmap\"                                                       14
addDefaultMap :: Fay ()                                                                                                15
addDefaultMap = ffi "olmap = new ol.Map({renderer: 'canvas'})"                                                         16
{-|                                                                                                                    17
  Initialises an object of the OpenLayers map as HTML object.                                                         18
  The JavaScript variable name \"olc\" must be reserved for the application!                                           19
-}                                                                                                                     20
defineCode :: Fay ()                                                                                                   21
defineCode = ffi "olc = $(olmap)[0]"                                                                                   22
-- | setting the target between html and OpenLayers                                                                   23
setTarget :: String -> Fay ()                                                                                          24
setTarget = ffi "olc.setTarget(%1)"                                                                                    25
-- | setting a default view for first map appearence                                                                  26
setDefaultView :: Fay ()                                                                                               27
setDefaultView = ffi "olc.setView(new ol.View({center:[0,0],zoom:2}))"                                                 28
-- | initialises an OpenLayers view and load the definitions from the OpenLayers webapplication defined in "OlApp"   29
olwrapperLoad :: Fay ()                                                                                                30
olwrapperLoad = void $ do                                                                                              31
    addDefaultMap                                                                                                      32
    defineCode                                                                                                         33
    setTarget targetId                                                                                                 34
    setDefaultView                                                                                                     35
    designMap                                                                                                          36
```

wrapper/OpenLayers/Func.hs

```
{-|
Module      :  OpenLayersFunc
Description :  OpenLayers JavaScript and Haskell functions (using FFI)
-}
module OpenLayers.Func where

import           Prelude hiding (void)
import           JQuery
import           Fay.Text hiding (head, tail, map)
import           OpenLayers.Html
import           OpenLayers.Internal
import           OpenLayers.Types
import           Fay.FFI
-- * NEW FUNCTION
-- | new MapQuest layer
newLayerMqt :: String     -- ^ map type
            -> Object     -- ^ new layer (Tile)
newLayerMqt = ffi "new ol.layer.Tile({source: new ol.source.MapQuest({layer: %1})})"
-- | new OpenStreetMap layer
newLayerOSM :: Object     -- ^ new layer (Tile)
newLayerOSM = ffi "new ol.layer.Tile({source: new ol.source.OSM()})"
-- | new Layer as vector
newVector :: Object      -- ^ vectorsource
          -> Opacity     -- ^ opacity
          -> Fay Object  -- ^ new layer (Vector)
newVector = ffi "new ol.layer.Vector({source: %1, opacity: %2.slot1*0.01})"
-- | new 'GeoFeature' ('GeoPoint' or 'GeoLine')
newFeature :: GeoFeature -> Fay Object
newFeature f = case f of
    GeoPoint p id s -> newFeaturePoint $ transformPoint p
    GeoLine pts id s -> newFeatureLine $ transformPoints pts
    _ -> error "newStyledFeature: the GeoFeature is not implemented yet"
-- | new map source GeoJSON as LineString
newFeatureLine :: [(Double, Double)]  -- ^ input coordinates
               -> Fay Object
newFeatureLine = ffi "new ol.source.GeoJSON({object:{'type':'Feature','geometry':{'type':'LineString','coordinates':
    %1}}})"
-- | new map source GeoJSON as Point
newFeaturePoint :: (Double, Double)  -- ^ an input coordinate
                -> Fay Object
newFeaturePoint = ffi "new ol.source.GeoJSON({object:{'type':'Feature','geometry':{'type':'Point','coordinates':
    %1}}})"
-- | new styled 'GeoLine'
newLineStyle :: GeoLineStyle -> Object
newLineStyle = ffi "[new ol.style.Style({stroke: new ol.style.Stroke({color: %1.color, width: %1.width})})]"
-- | new styled 'GeoPoint'
newPointStyle :: GeoPointStyle -> Object
newPointStyle = ffi "[new ol.style.Style({image: new ol.style.Circle({radius: %1.radius, fill: new ol.style.Fill({
    color:(%1.fillcolor == 'null' ? 'rgba(0,0,0,0)' : %1.fillcolor)}), stroke: %1.outcolor == 'null' ? null : new ol
    .style.Stroke({color: %1.outcolor, width: %1.outwidth})})})]"
-- | new OpenLayers DOM binding
newOlInput :: JQuery  -- ^ element to bind to
           -> String  -- ^ case (f.e. \"checked\" by Checkbox)
           -> Object  -- ^ map object (f.e. 'getLayerByIndex 0')
           -> String  -- ^ attribute of map object to change (f.e. \"visible\")
           -> Fay ()  -- ^ return type is void
newOlInput = ffi "(new ol.dom.Input(%1[0])).bindTo(%2, %3, %4)"
-- * ADD FUNCTION
-- | add an event on \"singleclick\" to display pop-up with coordinates in mercator and custom projection
addSingleClickEventAlertCoo :: String  -- ^ EPSG-Code of custom projection (f.e. \"EPSG:4326\")
                            -> Fay ()  -- ^ return type is void
addSingleClickEventAlertCoo = ffi "olc.on('singleclick', function (evt) {alert(%1 + ': ' + ol.proj.transform([evt.
    coordinate[0], evt.coordinate[1]], 'EPSG:3857', %1) + '\\nEPSG:3857: ' + evt.coordinate)})"
-- | add a layer to the map, and remove all layers before inserting (baselayer has now index 0)
addBaseLayer :: MapSource -> Fay ()
```

```
addBaseLayer s = void $ do
    removeLayers
    addMapLayer s
-- | add a MapSource to the map
addMapLayer :: MapSource -> Fay ()
addMapLayer s
      | s == OSM                  = addLayer newLayerOSM
      | Prelude.any(s==)mapQuests = addLayer ( newLayerMqt $ showMapSource s)
      | otherwise                 = error ("wrong MapSource allowed is OSM and " ++ show mapQuests)
-- | add a layer to the map
addLayer ::  Object -> Fay ()
addLayer = ffi "olc.addLayer(%1)"
-- | add a 'GeoFeature' to a new layer to the map (first add coordinates to a new feature and then add the style and
     the id to this new feature, at least create a layer with the feature and add the layer to the map)
addStyledFeature :: GeoFeature -- ^ input ('GeoPoint' or 'GeoLine')
                 -> Opacity    -- ^ opacity for the GeoFeature
                 -> Fay ()
addStyledFeature f o = do
    feature <- newFeature f
    styleFeature feature f
    setFeatureId feature f
    vector <- newVector feature o
    addLayer vector
-- | add more than one 'GeoFeature' to a new layer (similar to the 'addStyledFeature' function)
addStyledFeatures :: [GeoFeature]  -- ^ input ('GeoPoint' and/or 'GeoLine')
                  -> Opacity       -- ^ global opacity for the GeoFeatures
                  -> Fay ()
addStyledFeatures  f o = do
    features <- mapS newFeature f
    zipWithS styleFeature features f
    zipWithS setFeatureId features f
    vectors  <- zipWithS newVector features [ o | x <- [0..(Prelude.length features)-1]]
    addLayer (head vectors)
    sources <- return $ zipWith getVectorFeatureAt ( vectors) [ 0 | x <- [0..(Prelude.length features)-1]]
    addFeatures (head vectors) (tail sources)
-- | add an array with features to a layer
addFeatures :: Object     -- ^ layer
            -> [Object]   -- ^ featurearray
            -> Fay ()
addFeatures = ffi "%1.getSource().addFeatures(%2)"
-- | add a new point feature (and a new layer) by defining from HTML elements
addPointFromLabels :: String         -- ^ id of the HTML element for the first coordinate (element need value, must
     be a double, see 'Coordinate')
                   -> String         -- ^ id of the HTML element for the seconde coordinate (element need value, must
                         be a double, see 'Coordinate')
                   -> String         -- ^ id of the HTML element for the opacity (element need value, must be an
                        integer, see 'Opacity')
                   -> String         -- ^ id of the HTML element for the feature id (element need value, must be a
                        positive integer, see 'OpenLayers.Types.id')
                   -> GeoPointStyle  -- ^ define the style of the feature
                   -> Fay ()         -- ^
addPointFromLabels xId yId oId idId s = void $ do
    xinput <- selectId xId
    xcoor <- getVal xinput
    yinput <- selectId yId
    ycoor <- getVal yinput
    o <- getInputInt oId
    i <- getInputInt idId
    addStyledFeature (GeoPoint (Coordinate (toDouble xcoor) (toDouble ycoor) (Projection "EPSG:3857")) i s) (Opacity
         o)
-- | add a map event listener (f.e. when zoom or pan)
addMapWindowEvent :: String      -- ^ the event trigger (f.e. \"moveend\")
                  -> Fay JQuery  -- ^ action on the event
                  -> Fay ()
addMapWindowEvent = ffi "olc.on(%1, %2)"
-- | add a connection between HTML and OpenLayers to manipulate layer attributes
addOlDomInput :: String   -- ^ id of the HTML element which triggers
              -> String   -- ^ HTML value to trigger (f.e. \"checked\")
              -> String   -- ^ layer attribute to manipulate (f.e. \"visible\")
              -> Object   -- ^ layer to manipulate
```

```
              -> Fay ()
addOlDomInput id typehtml value method = void $ do
    element <- selectId id
    newOlInput element typehtml method value
-- * REMOVE FUNCTION
-- | remove all layers from the map
removeLayers :: Fay ()
removeLayers = void $ do
    layers <- getLayers
    mapM removeLayer layers
-- | remove a layer object (only use with layers)
removeLayer :: a       -- ^ layer to remove
            -> Fay ()
removeLayer = ffi "olc.removeLayer(%1)"
-- * CHANGE FUNCTION
-- | zoom IN and specify levels to change
zoomIn :: Integer    -- ^ number of zoomlevels to change
       -> Fay ()
zoomIn = ffi "olc.getView().setZoom(olc.getView().getZoom()+%1)"
-- | zoom OUT and specify levels to change
zoomOut :: Integer   -- ^ number of zoomlevels to change
        -> Fay ()
zoomOut = ffi "olc.getView().setZoom(olc.getView().getZoom()-%1)"
-- | style a feature @/after/@ creating a new feature and @/before/@ adding to a new layer (this is an internal
     function for 'addStyledFeature' and 'addStyledFeatures')
styleFeature :: Object      -- ^ the previously created new feature
             -> GeoFeature  -- ^ the GeoFeature object of the previously created feature
             -> Fay ()
styleFeature object feature = case feature of
    GeoPoint p id s  -> styleFeature' object $ newPointStyle s
    GeoLine pts id s -> styleFeature' object $ newLineStyle  s
    _ -> error "styleFeature: the GeoFeature is not implemented"
-- | style a feature FFI function
styleFeature' :: Object -> Object -> Fay ()
styleFeature' = ffi "%1.getFeatures()[0].setStyle(%2)"
-- | change the baselayer at layerindex 0
changeBaseLayer :: MapSource   -- ^ new 'MapSource' for the baselayer
                -> Fay ()
changeBaseLayer s = void $ do
    layers <- getLayers
    addBaseLayer s
    mapS addLayer $ tail layers
-- * SET FUNCTION
-- | set the id of a feature, get the feature by layer and featureindex
setId ::  Object  -- ^ layer with the requested feature
      -> Integer  -- ^ new id for the feature ( > 0)
      -> Integer  -- ^ index of the feature in the layer (Layer.getFeatures()[i] , i >= 0)
      ->  Fay ()
setId = ffi "(%2 < 1 || %3 < 0) ? '' : %1.getSource().getFeatures()[%3].setId(%2)"
-- | set the id of the first feature of the vector
setFeatureId :: Object     -- ^ vector
             -> GeoFeature -- ^ input for the id
             -> Fay ()
setFeatureId = ffi "%1.getFeatures()[0].setId(%2.id)"
-- | set map center with a 'Coordinate'
setCenter :: Coordinate -> Fay ()
setCenter c = setCenter' $ transformPoint c
-- | set map center FFI function
setCenter' :: (Double, Double) -> Fay ()
setCenter' = ffi "olc.getView().setCenter(%1)"
-- | set the map center and the zoomlevel at the same time
setCenterZoom :: Coordinate  -- ^ center position
              -> Integer     -- ^ zoomlevel
              -> Fay ()      -- ^ return type is void
setCenterZoom c z = void $ do
    setCenter c
    setZoom z
-- | set the zoomlevel of the map
setZoom :: Integer -> Fay ()
setZoom = ffi "olc.getView().setZoom(%1)"
```

```
-- * GET FUNCTION
-- | get map center in requested projection with n decimal places
getCenter :: Projectionlike  -- ^ requested projection
          -> Integer         -- ^ decimal places
          -> Fay Text
getCenter proj fixed = do
     c <- getCenter'
     coordFixed (transformPointTo proj c) fixed
-- | get map center FFI function
getCenter' :: Fay (Double, Double)
getCenter' = ffi "olc.getView().getCenter()"
-- | get map zoom level
getZoom :: Fay Text
getZoom = ffi "olc.getView().getZoom()"
-- | get an array of all layers in the map
getLayers :: Fay [Object]
getLayers = ffi "olc.getLayers().getArray()"
-- | get a layer by index and return a object
getLayerByIndex :: Integer -> Object
getLayerByIndex = ffi "olc.getLayers().item(%1)"
-- | get a layer by index and return a fay object
getLayerByIndex' :: Integer -> Fay Object
getLayerByIndex' = ffi "olc.getLayers().item(%1)"
-- | get the id of a feature (<http://openlayers.org/en/v3.1.1/apidoc/ol.Feature.html OpenLayers Feature>)
getFeatureId :: Object
             -> Integer
getFeatureId = ffi "%1.getId()"
-- | get a feature from a vector at index position
getVectorFeatureAt :: Object      -- ^ Vector
                   -> Integer     -- ^ index of vector features
                   -> Object
getVectorFeatureAt = ffi "%1.getSource().getFeatures()[%2]"
-- | get the number of features in a vector
getVectorFeatureLength :: Object  -- ^ Vector
                       -> Integer -- ^ number of features in vector
getVectorFeatureLength = ffi "%1.getSource().getFeatures().length"
-- * TRANSFORM FUNCTION
-- | transform coordinates from mercator to requested projection
transformPointTo :: Projectionlike    -- ^ target projection
                 -> (Double, Double)  -- ^ input
                 -> (Double, Double)  -- ^ output
transformPointTo = ffi "ol.proj.transform(%2, 'EPSG:3857', %1.slot1)"
-- | transform 'Coordinate' to mercator (EPSG:3857)
transformPoint :: Coordinate        -- ^ input
               -> (Double, Double)  -- ^ output
transformPoint = ffi "ol.proj.transform([%1.x, %1.y], %1.from.slot1, 'EPSG:3857')"
-- | transform 'Coordinate's to mercator (EPSG:3857)
transformPoints :: [Coordinate] -> [(Double, Double)]
transformPoints c = [transformPoint x | x <- c]
-- * OTHER FUNCTION
-- | create a Text from a tuple of doubles with fixed decimal places and seperator \",\"
coordFixed ::  (Double, Double) -> Integer -> Fay Text
coordFixed = ffi "%1[0].toFixed(%2) + ',' + %1[1].toFixed(%2)"
-- | change Text to Double
toDouble :: Text -> Double
toDouble = ffi "%1"
-- | 'sequence' the 'map'-function
mapS :: (a -> Fay b) -> [a] -> Fay [b]
mapS f x = sequence (map f x)
-- | 'sequence' the 'zipWith'-function
zipWithS :: (a -> b -> Fay c) -> [a] -> [b] -> Fay [c]
zipWithS f x y = sequence $ zipWith f x y

-- zipWithS3 :: (a -> b -> c -> Fay d) -> [a] -> [b] -> [c] -> Fay [d]
-- zipWithS3 f x y z = sequence $ zipWith3 f x y z
```

wrapper/OpenLayers/Html.hs

```
{-|
Module      :  OpenLayersHtmlFunc
Description :  OpenLayers HTML and JQuery functions and constants
-}
module OpenLayers.Html where

import           Prelude hiding (void, showString)
import           JQuery
import           Fay.Text
import           Fay.FFI
import           OpenLayers.Internal
-- * CONSTANTS
-- | a \<a\> element
aElement  = "<a>"
-- | a \<br\> element
brElement = "<br>"
-- | a \<p\> element
pElement  = "<p>"
-- | a \<div\> element
divElement  = "<div>"
-- | a \<form\> element
formElement  = "<form>"
-- | a \<input\> element
buttonElement = "<input>"
-- | a \<input\> element of type text
inputTextElement = "<input type=text>"
-- | a \<input\> element of type checkbox
inputCheckboxElement = "<input type=checkbox>"
-- * FUNCTIONS
-- ** ADD FUNCTIONS
-- | add a new html element to an existing and fill the content of the new with text
addTextToHtml :: String  -- ^ text to add
              -> String  -- ^ type of the html element to add
              -> String  -- ^ id of the html element to append
              -> Fay ()
addTextToHtml content element parentId = void $ do
     c <- selectId parentId
     d <- select' element
     setHtml' content d
     appendTo c d
-- | add a new button (type=submit) to an existing html element with a text and a method
addButton :: String  -- ^ text for the button
          -> String  -- ^ id of the html element to append
          -> Fay ()  -- ^ method for the button
          -> Fay ()
addButton text parentId method = void $ do
     c <- selectId parentId
     d <- select' buttonElement
     setAttr' "value" text d
     setAttr' "type" "submit" d
     onClick' method d
     appendTo c d
-- | add a new checkbox to an existing html element with a text and a method
addCheckbox :: String  -- ^ id for the new checkbox
            -> String  -- ^ id of the html element to append
            -> String  -- ^ text for the checkbox
            -> Fay ()
addCheckbox id parentId label = void $ do
    parent  <- selectId parentId
    div     <- select' divElement
    setHtml' label div
    checkbox <- select' inputCheckboxElement
    setAttr' "id" id checkbox
    appendTo div checkbox
    appendTo parent div
```

```
-- | add a div-element with a text
addDiv :: String  -- ^ id of the html element to append
       -> String  -- ^ id for the new div-element
       -> String  -- ^ text for the div-element
       -> Fay ()
addDiv parentId divId label = void $ do
    parent  <- selectId parentId
    div    <- select' divElement
    setAttr' "id" divId div
    setHtml' label div
    appendTo parent div
-- | add a custom html element
addElement :: String  -- ^ id of the html element to append
           -> String  -- ^ element type (f. e. \"<div>\")
           -> String  -- ^ id for the new element
           -> Fay ()
addElement parentId element id = void $ do
    parent <- selectId parentId
    d <- select' element
    setAttr' "id" id d
    appendTo parent d
-- | add a new form-element
addForm :: String   -- ^ id of the html element to append
        -> String   -- ^ id for the new form-element
        -> Fay JQuery
addForm parentId formId = do
    parent   <- selectId parentId
    form     <- select' $ formElement
    setAttr' "id" formId form
    appendTo parent form
-- | add a new input element of type text
addInput :: String  -- ^ label of the element
         -> String  -- ^ id of the html element to append
         -> String  -- ^ id for the new input element
         -> String  -- ^ default value for the text field
         -> Fay ()
addInput label parentId elementId defaultValue = void $ do
    parent <- selectId parentId
    div    <- select' divElement
    setHtml' label div
    input   <- select' inputTextElement
    setAttr' "id"    elementId    input
    setAttr' "value" defaultValue input
    appendTo div input
    appendTo parent div
-- | add a new breakline
addBreakline :: String  -- ^ id of the html element to append
             -> Fay ()
addBreakline id = void $ do
    parent <- selectId id
    value <- select' brElement
    appendTo parent value
-- ** OTHER FUNCTIONS
-- | get the integer value of an element
getInputInt :: String       -- ^ id of the html element
            -> Fay Integer
getInputInt labelId = do
    idinput <- selectId labelId
    idvalue <- getVal idinput
    toInt idvalue
-- | select an element due to his id
selectId :: String      -- ^ id of the html element
         -> Fay JQuery
selectId s = select $ showString $ "#" ++ s
-- | in combination f. e. with "addMapWindowEvent" to change the html element content
setEventToHtml :: String      -- ^ id of the html element
               -> Fay Text    -- ^ method to generate text (f. e. "getZoom")
               -> Fay JQuery
setEventToHtml elementId function = do
    element <- selectId elementId
```

```
    f <- function
    setHtml f element
-- | String to Text with FFI
showString :: String -> Text
showString = ffi "%1"
-- | Double to Text with FFI
showDouble :: Double -> Text
showDouble = ffi "%1"
-- | Text to Fay Integer with FFI
toInt :: Text -> Fay Integer
toInt = ffi "%1"
-- | String to Integer with FFI
toInt' :: String -> Integer
toInt' = ffi "%1"
-- ** ADAPTED FAY.JQUERY FUNCTIONS
-- | adapted from JQuery's "select"
onClick' :: Fay ()      -- ^ method for the button
         -> JQuery      -- ^ button element
         -> Fay JQuery
onClick' = ffi "%2['click'](%1)"
-- | adapted from JQuery's "select"
select' :: String       -- ^ html element to select (see CONSTANTS)
        -> Fay JQuery
select' s = select $ showString s
-- | adapted from JQuery's "setHtml"
setHtml' :: String      -- ^ value for inner html
         -> JQuery      -- ^ html element to set
         -> Fay JQuery
setHtml' content target = setHtml (showString content) target
-- | adapted from 'Fay.JQuery.setAttr'
setAttr' :: String      -- ^ parameter to set (f.e. \"id\")
         -> String      -- ^ value for the parameter
         -> JQuery      -- ^ html element to set
         -> Fay JQuery
setAttr' att val target = setAttr (showString att) (showString val) target
```

wrapper/OpenLayers/Internal.hs

```
{-|
Module      :  OpenLayers Wrapper Internal Definitions
Description :  use this modul to for global definitions
-}
module OpenLayers.Internal where

import           Fay.FFI
-- | a functor to ignore the result of the function, such as the return value of an IO action.
void :: Fay f -> Fay ()
void f = f >> return ()
```

wrapper/OpenLayers/Types.hs

```
{-|
Module      :  OpenLayersTypes
Description :  new types, new data, global constants and mapsource for olwrapper
-}
module OpenLayers.Types where
-- * Constants
-- | The 'Projectionlike' for WGS84
wgs84proj = Projection "EPSG:4326"
-- | The 'Projectionlike' for Mercator
mercatorproj = Projection "EPSG:3857"
-- * Types
-- ** type
-- | Layers with IDs are using 'GeoId'
type GeoId = Integer
-- ** data
-- | OpenLayers Projection constructed f.e. with \'Projection \"EPSG:4326\"\'
data Projectionlike = Projection String
-- | constructor for a coordinate with x,y and projection
data Coordinate = Coordinate {
        -- | first coordinate
        x :: Double,
        -- | second coordinate
        y :: Double,
        -- | projection
        from :: Projectionlike
    }
-- | Layer Opacity from min=0 (not visible) to max=100
data Opacity = Opacity Integer
-- | defining a new feature to add to the map, possible are 'GeoPoint' and 'GeoLine'
data GeoFeature =
    -- | A GeoPoint is a styled point feature with an id
    GeoPoint {
        -- | position with a 'Coordinate'
        pt ::  Coordinate,
        -- | id for the feature with 'GeoId'
        id :: GeoId,
        -- | pointstyle with 'GeoPointStyle'
        pstyle :: GeoPointStyle
    } |
    -- | A GeoLine is a styled line feature with an id
    GeoLine {
        -- | linepoints as a  list of 'Coordinate'
        pts :: [Coordinate],
        -- haddock only need one id definition
        id :: GeoId,
        -- | linestyle with 'GeoLineStyle'
        lstyle :: GeoLineStyle
    }
-- | defining a style for a 'GeoLine' feature
data GeoLineStyle = GeoLineStyle {
        -- | line color, can be a name (red) or code (#FF0000)
        color :: String,
        -- | width of the line
        width :: Integer
    }
-- | defining a style for a circled 'GeoPoint' feature
data GeoPointStyle = GeoPointStyle {
        -- | radius of the circle
        radius :: Integer,
        -- | fill color of the circle, can be a name (red) or code (#FF0000)
        fillcolor :: String,
        -- | line color of the border of the circle, name (red) or code (#FF0000)
        outcolor :: String,
        -- | width of the border of the circle, if 0 no outcolor needed
        outwidth :: Integer
```

```
    }
-- | list of possible mapsources
data MapSource
    = Sat  -- ^ MapQuest's satellite
    | Hyb  -- ^ MapQuest's hybrid
    | Osm  -- ^ MapQuest's OpenStreetMap
    | OSM  -- ^ OpenStreetMap
    deriving (Show, Eq)
-- * List
-- | puts the sources from MapQuest in a list
mapQuests = [Osm, Sat, Hyb]
-- * Functions
-- | transform 'MapSource' to 'String' and handle wrong input
showMapSource :: MapSource -> String
showMapSource ms
    | ms == Sat = "sat"
    | ms == Hyb = "hyb"
    | ms == Osm = "osm"
    | ms == OSM = "OSM"
    | otherwise = "showMapSource not defined, check Layer"
```