knottyTom's MTB Tour Guide System
Handbuch 0.1pre2 für Version 0.5

knottyTom@berlios.de

21. November 2005

Inhaltsverzeichnis

1 Was ist/wird/kann das?

Ich fahre gerne mit meinem MTB durch die Gegend. Anfangs einfach so ins Blaue hinein, dann nahm ich schon mal 'ne Karte mit, als ich so fit war auch mal in Gegenden vorzudringen, wo ich mich nicht mehr so gut auskannte.

Irgendwann entdeckte ich dann, dass man im Buchhandel auch Bücher kaufen konnte, die komplette Touren beschrieben. "Super Sache", dachte ich mir.

Anfangs... diese Bücher sind ja nicht ganz billig. Naja, mal im Internet gucken: Vielleicht gibt es da ja auch was. Nicht schlecht getippt, da gab es wirklich schon 'ne Menge Sachen. Aber irgendwie war jede(r) Autor(in) und jede Website der Meinung, ein eigenes System schaffen zu müssen.

Alles war bissl anders formatiert, bei manchen Beschreibung fehlten mir einfach ein paar Infos zur Tour. Alles in Allem nicht gerade das, was einen Software-Entwickler vom Hocker reißt: Ich brauch' Struktur und die Möglickeit, die vorhandenen Informatioen so aufbereiten zu können, wie ich sie gerne lese (oder zumindest verschiedene wählbare Einstellungen).

Jetzt wird so manch' eine(r) sagen: "Ganz schön frech, da bekommt er die Infos für lau und beschwert sich auch noch darüber". Stimmt!

Aber frech wäre es nur dann, wenn man sich nicht daran macht, das zu verbessern, was man kritisiert. Das mache ich mit diesem Projekt.

1.1 Häufig geäußertes Mißverständnis

Natürlich macht es wenig Sinn eine Tour (doch etwas) umständlich über dieses System zu erfassen, um die erfassten Daten nur persönlich zu nutzen. Denn: Wenn man so 'ne Tour mal gefahren ist, dann kann man sie meist *blind* wieder abfahren. Genau!

Das Ziel ist vielmehr, die erfasste Tour anderen Bikern zur Verfügung zu stellen. Einmal erfasst kann man sie jedem Interessierten zum Beispiel per eMail zukommen lassen.

Das *ferne* Ziel ist es natürlich alle Touren zentral auf einem Rechner im Internet zu sammeln, damit Leute durch die Sammlung blättern, sich eine passende aussuchen können und dann alles an Infos zur Verfügung haben, was sie zum Abradeln der Tour brauchen.

Man hätte eine einheitliche Oberfläche um *seine* Tour zu finden. Hat man sie gefunden, hätte man einen einheitlichen Aufbau der Tourbeschreibung. Das ist das Ziel. Das ist meine Idee ...

1.2 Enttäuschte Hoffnungen

Ja, es tut mir leid, aber dies ist erst Version 0.5 des Projekts. Ich enttäusche euch lieber gleich, bevor ich zu viele Hoffnungen wecke.

Dieses System ist derzeit keine Software mit der man eine Tourbeschreibung mit Hilfe einer grafischen Benutzerschnittstelle erfassen kann. Es bietet auch keine

Datenbankunterstützung.

Wenn ihr nicht aus dem Unix/Linux-Bereich kommt, dann fragt ihr euch sicher was das (in das ich so viel Freizeit investiere) überhaupt soll. Immerhin kann ich sagen, dass dieses System auch unter Windows und vermutlich auch am Mac läuft. Aber wenn ihr es nicht mögt auf der Kommandozeile Befehle zu tippen, dann solltet ihr mal wieder zu einem späteren Zeitpunkt hierher zurückkommen.

Aber vielleicht lest ihr euch doch noch den nächsten Abschnitt durch ...

1.3 Ist-Zustand der Version 0.5

Ok, nach dem letzten Abschnitt mag man sich fragen: "Ja, was kann es denn überhaupt"? Gute Frage. Ich werde versuchen sie hier zu beantworten.

Ich habe erstmal versucht, eine XML-Struktur zu schaffen, die es dem Autor der Tour ermöglicht, möglichst alle Daten, die relevant sind, zu erfassen. Das sind derzeit Punkte wie:

- Es werden Daten erfasst, die dazu dienen, die Tour in einem Online-System verfügbar zu machen.
- Erfasst wird: Technisches Können, Fitness, Typ der Tour, Dauer, Gesamtlaenge.
- Daten zur Anreise
- Einbindung einer Karte als GIF, JPG oder PNG. Ausserdem Hinweise zu gedruckten Karten.
- Detaillierte Informationen zum Streckenabschnitt.
  - Höhe
  - Entfernung zum letzten Wegpunkt
  - Breitengrad
  - Längengrad
  - Staßenbelag/Wegbeschaffenheit (grafische Darstellung)
  - Interessante Punkte
  - Restaurants/Lokale/Kneipen
  - Anekdoten
  - Orientierung: Windrose zeigt den weiteren Weg
  - Orientierung: Beliebige Anzahl an Bildern beschreiben die Tour
- Und natürlich: Beschreibungen, Beschreibungen, ...

Ferner wird automatisch ein Profil (mit grafischer Darstellung des Streckenbelags) der Tour generiert. Die Ausgabe ist prinzipiell in jeder Sprache (derzeit wird aber nur Deutsch und Englisch unterstützt) möglich. Ausserdem sollte das alles auf Windows, Linux (mein Entwicklunssystem) und Mac (nicht getestet) laufen.

1.4 Pläne

Da gibt es **so** viele. Jetzt schreib' ich erstmal dieses Handbuch zu Ende. Dann gibt es da so simple Dinger wie:

- Gedruckte Formulare zur Erfassung der Tour.
- Tips und Tricks: Wie male ich 'ne Karte?
- Stylesheet für Druckformat.
- Mehrtagestouren.
- ...

Und dann so Hämmer wie:

- Grafische Benutzerschnittstelle (da läuft schon was ;-).
- GPS-Systeme: Wie kann man die anbinden?
- Wie kann man Höhenprofile einbinden, die von anderen Systemen aufgezeichnet werden.
- *totally weird*: Audio-Ausgabe der Tour via MP3-Player.
- *bit weird*: Tips und Tricks: Wie erfasse ich die Tour mit 'nem MP3-Player?

Ihr seht schon, da kommt noch einiges. Und ich hocke hier alleine rum. Also, wenn ihr Lust habt hier mitzuwerkln $\Longrightarrow$ bitte melden! Ich kann jeden brauchen: Coder, Designer, Tester, Writer, Biker, Visionär, Hardware, ...

2 Bevor es los geht

Ich schreib' ja nicht alles selber. Ein paar Sachen brauchen wir schon noch.

2.1 Java 1.5+

Das ganze System ist Java-basiert, was dem Vorteil hat, dass es auf allen Plattformen läuft, die eine *Java Virtual Machine* zur Verfügung stellen. Sollte man diese nicht besitzen, dann ist es der Zeit, diese zu installieren.

Die aktuelle Java-Version ist bei SUN[1] zu finden. Dort bitte *Java 5.0 JDK* wählen.

2.2 Jakarta ANT 1.6.2+

Wir brauchen auch den Jakarta ANT Framework, der ist beim Jakarta Projekt [2] zu finden. Zum Zeitpunkt der Erstellung dieser Dokumentation ist die Version 1.6.5 die aktuellste. Das sogenannte *Binary Release* genügt für unsere Zwecke.

[1] http://java.sun.com/j2se/1.5.0/download.jsp
[2] http://ant.apache.org/bindownload.cgi

3 Installation

Ah, sind doch noch ein paar Leute da? *Ok. Let's get into it ...*

Als erstes muessen wir mal 1(2) Datei(en) von der Projektseite runterladen. Die sind zu finden unter

http://www.web-space.tv/knottytom/index.html#download

Wir brauchen das Basissystem (base system) und die benutzten Java-Bibliotheken (needed Java libraries). Der Download ist getrennt, da die Bibliotheken recht fett werden können, sich aber über die Zeit eher wenig ändern. Es besteht also eigentlich kein Grund diese Bibliotheken mit jeder neuen Version des Basissystems immer wieder neu zu saugen!

Gut, wir haben jetzt mal beide zip-Dateien gezogen. Jetzt legen wir ein Arbeitsverzeichnis an und kopieren beide zip-Dateien da rein. Nicht nervös werden, wenn die beiden Dateien **nicht** dieselbe Versionsnummer haben. Das macht nichts!

In diesem Arbeitsverzeichnis packen wir nun beide (!) Dateien aus. Unter Erhaltung der Verzeichnisstruktur, bitte.

4 Schnellstart

OK. Ihr habt also alles installiert. Jetzt sollte man auch mal testen, ob alles funktioniert ... und natürlich auch um seine Neugierde zu befriedigen ;-)

Führt einfach die nachfolgenden Schritte aus. Falls etwas nicht funktionieren sollte, dann schreibt eine Mail an mich und beschreibt die Fehlermeldung(en). Ich hoffe aber natürlich inständig, daß alles klappt. In dem Fall könnt ihr mir auch eine Mail schreiben. Einfach um mein Ego zu stärken ...

Im folgenden muss ich mich oftmals auf Verzeichnisse beziehen. Wenn ich `./` (Punkt Slash) schreibe, dann meine ich damit **das** Verzeichnis, in dem ihr die Zip-Dateien ausgepackt habt. Alle Aktionen die nachfolgend durchgeführt werden, müssen in diesem Verzeichnis gestartet werden.

4.1 Build.properties anpassen

Wir haben eine zentrale Steuerungsdatei, die kontrolliert welche Verzeichnisstruktur angelegt, welche Sprache und welches Layout (für HTML) verwendet werden soll. Dies Datei heißt `./build.properties`. Bitte öffnet und editiert diese mit eurem Lieblingseditor[3].

Die Datei `./build.properties` sollte dann wie folgt aussehen (die Zeilennummern hab' ich nur zur Orientierung da, sie gehören **nicht** in die Datei!):

```
1 tour.name=MeineSuperTour
2 # Currently supported languages: de, en
3 lang=de
4 css=biketour.css
```

4.2 Skeleton generieren

Soweit so gut. Aber um irgendwas zu sehen brauchen wir ja 'ne Tourbeschreibung. Woher nehmen, wenn nicht zaubern? Einfach zaubern! Das System kann ein Grundgerüst (englisch: *Skeleton*) anlegen. Dazu ist nur der nachfolgende Aufruf von Ant notwendig. Ab auf die Kommandozeile und den nachfolgenden Befehl ausführen:

```
1 ant gen-skeleton
```

4.3 HTML-Ausgabe erzeugen

Das Erzeugen der HTML-Datei ist genauso einfach. Auf der Kommandozeile tippt ihr einfach diesen Befehl:

```
1 ant gen-tour-html
```

Da könnten jetzt zwei Warnungen (englisch: *warning(s)*) auftauchen. Die könnt ihr ignorieren.

[3] Du nimmst jetzt den `vi`? Cool!

4.4 HTML im Browser bewundern

Jetzt kommt der große Moment, wo der Frosch ins Wasser springt. In dem Verzeichnis `./html/MeineSuperTour` sollte sich eine Datei namens `MeineSuperTour.html` befinden.

Öffnet diese Datei (`./html/MeineSuperTour/MeineSuperTour.html`) in einem Browser eurer Wahl[4]. Wenn ihr jetzt 'ne Beschreibung einer fiktiven Tour seht, dann läuft das System. **Glückwunsch!**

[4] Getestet hab' ich nur: Firefox und Konquerer

5 Tips und Tricks

Hier werde ich eine Tricks zum Besten geben, die mir recht hilfreich erscheinen. Tut mir leid, daß hier zur Zeit noch nichts zu finden ist.

5.1 Karte erstellen (TODO)

Achtung! Landkarten unterliegen prinzipiell dem Urheberrecht. Dies gilt auch fuer Imagedateien (JPGs, GIFs, PNGs, ...) aus dem Internet. Ohne Genehmigung des Urhebers könnte ihr sie nicht einfach kopieren und unter's Volk bringen.

- Get the map
- Start Gimp
- Copy information to other layer
- Remove layer with map

5.2 Formulare verwenden (TODO)

5.3 Tourerfassung mit MP3-Player (TODO)

6 Der Aufbau der XML-Datei

Der gesamte Inhalt einer Tourbeschreibung wird in einer XML-Datei abgelegt. Ich werde in diesem Kapitel detailliert beschreiben, wie diese Datei aufgebaut ist.

Das Wissen um den Aufbau dieser Datei ist aus zwei Gründen notwendig:

1. Solange wir keine grafische Schnittstelle zur Erfassung besitzen, muß man diese Datei per Hand erfassen. Dazu muß man natürlich wissen, wie sie aufgebaut ist.
2. Auch wenn wir irgenwann eine grafische Schnittstelle haben, muß ein(e) Entwickler(in), die/der das System erweitern will, wissen, wie der Aufbau nun exakt aussieht[5].

Sieht man sich die XML-Datei das erste Mal an, dann schaut das alles recht komplex und konfus aus. Aber, ehrlich, so schwierig ist das alles gar nicht. Fangen wir mal an...

Anm.: In den XML-Beispielen sind Zeilennummern ergaenzt. Diese gehören natürlich ***nicht*** *in die echte Datei!*

6.1 <tourguide>

Die gesamte Tourbeschreibung wird in zwischen dem öffnenden und dem schließenden `<tourguide>`-Tag eingeschloßen. Das Ganze sieht also so aus:

```
1  <tourguide>
2      <storage-info>
3          <location />
4      </storage-info>
5      <general>
6          ...
7      <general>
8      <crosspoints>
9          ...
10     </crosspoints>
11 <tourguide>
```

Im Wesentlich haben wir also drei Abschnitte:

1. <storage-info>
2. <general>
3. <crosspoints>

Alle Abschnitte werden nachfolgenden genauer beleuchtet. Der Wichtigste is `<crosspoints>`, da er die eigentlich Beschreibung der Tour beinhaltet. Wozu sind aber die anderen da? Fangen wir an mit...

[5] Ok, es existiert eine dokumentierte DTD, aber das nur am Rande...

6.2 <storage-info> (<location>)

Wie schon erwähnt, erfassen wir ja unsere Tour nicht für uns selbst: Wir kennen sie ja schon. Sondern für andere.

Mal angenommen, du wohnst in Hamburg und fährst übers Wochende nach Buttenheim in Bayern (das MTB ist natürlich dabei). Am Samstag hättest du gute drei Stunde Zeit 'ne kleine Tour zu fahren. Gibt's da bekannte Touren?

Wäre jetzt nett, einfach ins Internet zu gehen. Das dir bekannte Interface zu benutzen und mal zu gucken, was da in Buttenheim so geht.

Genau dafür ist das Tag `<storage-info><location></storage-info>` vorhanden. Wie? Schauen wir uns das mal an:

```
1  <tourguide>
2      <storage-info>
3          <location
4              country="de"
5              region-or-state="Bavaria"
6              city="Buttenheim"
7              zip-code="96155"/>
8      </storage-info>
9      ...
10 <tourguide>
```

Alles klar? Alle Infos werden mithilfe der Attribute des Tags `<location>` erfaßt. Wie werden diese Attribute nun zur Suche benutzt? Sorry, das ist (noch?) nicht Aufgabe dieses Projekts. Sie sind erstmal da. Es liegt nun an demjenigen, der die Suchfunktion implementiert, wie clever die erfassten Daten genutzt werden.

Eine kurze Anmerkung zum Attribut `country`: Es sollten die sogenannten Top Level Domains (wie sie aus dem Internet bekannt sind) verwendet werden. Ein einfacheres und bekannteres System zur Angabe von Ländern existiert praktisch nicht.

Gut. Nun haben wir (mit viel Glück) drei Touren gefunden. Aber sind wir dafür auch fit genug? Dauert das nicht zu lange? Wie kommt man an den Startpunkt? Gibt's ne Karte? Ja, das braucht man alles. Deswegen gibt es das Tag `<general>`...

6.3 Ein Wort zu Bildern

Hmm, den Abschnitt `<general>` hab' ich ins nächste Kapitel verschoben. Ich muß noch schnell was zu Bildern im Allgemeinen sagen. Dauert nicht lange, ist aber wichtig für **alle** nachfolgenden Abschnitte.

Natürlich unterstützt das System die Verwendung von Bildern. Ohne Bilder wäre das Ganze wohl nur, wenn überhaupt, die Hälfte wert. Dazu müssen die Bilder aber auch gefunden werden. Aber wie?

So: In dem Verzeichnis in dem die XML-Datei, über die wir hier ja reden, zu liegen kommt, existiert [6] ein Unterverzeichnis namens `images`. Hier hinein wer-

[6] Wenn das das Grundgerüst via `ant gen-skeleton` angelegt wurde, dann existiert es au-

den alle Bilder kopiert, die wir später benötigen.

Gut, jetzt hätten wird alle Bilder in unserem Verzeichnis. Aber wie geben wir sie in der XML-Datei an? Auch das ist ziemlich einfach. Schauen wir uns mal ein Beispiel an:

```
<image name="big_drop.jpg"/>
```

Das war's auch schon. Die Bilder werden **ohne** Angabe des Verzeichnisses (`images`) durch die Notation des Bildnamens eingetragen. An welchen Stellen das nötig/möeglich ist, werden wir im weiteren Verlauf erfahren.

6.4 <general>

Ok. Nach dem Exkurs zur Angabe von Bildern, widmen wir uns jetzt dem nächsten Abschnitt der XML-Datei: `<general>`. Erstmal wieder ein Beispiel:

```
<tourguide>
    ...
<general
    fitness-level="easy"
    tech-level="medium"
    type="round"
    duration="01:00hrs">

    <name>Nuernberger Tiergarten
          --- Kleiner Trial inklusive 'Roller Coaster'
    </name>
    <distance unit="km">10</distance>
    <desc>
        Wir starten am Tiergarten, fahren etwas uphill durch
        die Stromschneisse, dann den 'Mini-DH' runter...
    </desc>
    <reach>
       <option name="Mit dem Auto">Auf der A3 aus Richtung
          Wuerzburg oder Passau kommend bei Nuernberg/Moegeldorf
          abfahren...
       </option>
       <option name="Per Zug">Am Nuernberg Hbf aussteigen. Dann
          mit der Strassenbahn direkt zum Tiergarten.
       </option>
    </reach>
    <maps>
       <map>Kompass, Karte 170</map>
    </maps>
    <roadmap image="map.jpg"/>
    <profile />
</general>
    ...
<tourguide>
```

tomatisch. Ansonsten muß es per Hand z.B. mit dem Befehl `mkdir images` unter linux-artigen Systemen bzw. `md images` auf einem Windows-Rechner angelegt werden.

6.4.1 Die Attribute von <general>

Wie man aus dem Beispiel ersieht, hat <general> vier Attribute. Diese dienen dem/r Anwender(in) dazu, zu entscheiden, ob die Tour füer ihn/sie geeignet ist.

Die Angaben hier sind nicht ganz einfach, aber wichtig. Wenn du topfit bist, die Tour aber auch für dich hart war, dann bringt das hier zum Ausdruck. Wenn der Downhill für dich nicht ganz einfach, aber du technisch nicht so perfekt bist, dann setze den `tech-level` nicht gleich auf `extrem`.

Nachfolgend die Attribute und ihre Bedeutung (was man hierfür angeben kann, wird im Anschluß erläutert):

1. fitness-level: Wie fit muss man füer die Tour sein?
2. tech-level: Wie hoch ist der technische Anspruch? Knifflige Wurzelpassagen? Extreme Downhills (mit fetten Drops)?
3. type: Rundtour, Single Trial, ...
4. duration: Dauer der Tour. Als du sie gefahren bist.

So und nun zu den restlichen Tags die innerhalb von <general> verwendet werden können oder müssen...

6.4.2 <distance>

6.4.3 <desc>

6.4.4 <reach>

6.4.5 <maps>

6.4.6 <roadmaps>

6.4.7 <profile/>

6.5 <crosspoints>

Mann-o-mann, so 'ne Menge Zeugs zu tippen. Wann kommt jetzt eigentlich die Beschreibung der Tour? Voila... jetzt!

Innerhalb des <crosspoints>-Abschnitts werden alle relevanten Wegpunkte der Tour beschrieben (und, optional, noch vieles mehr).

Hier mal ein Beispiel:

```
<tourguide>
...
<crosspoints>
   <crosspoint
       distance="300" elevation="600"
       direction="northwest"
       latitude="000" longitude="000">
```

```
8         <desc> Ein nordwest Punkt.</desc>
9         <profile-desc>
10           Die Stromschneisse... los geht's
11 ␣␣␣␣␣</profile-desc>
12 ␣␣␣</crosspoint>
13
14 ␣␣␣<track-info␣pavement="road">
15 ␣␣␣␣␣␣<anecdote␣distance="gesamter␣Weg"␣image="drop.jpg">
16 ␣␣␣␣␣␣␣␣␣␣Wenn's gar nicht mehr geht, dann einfach ab in
17             den Wald. Der ist nicht dicht... man kann da
18             auch gut schieben (und wird nicht gesehen ;-).
19         </anecdote>
20     </track-info>
21
22     <crosspoint distance="200" elevation="310"
23                 direction="none"
24                 latitude="000" longitude="000">
25       <desc>Back At Home!</desc>
26     </crosspoint>
27 <crosspoints>
28 <tourguide>
```

WTF[7]? Jaja, schaut kompliziert aus, is' aber auch bloß mit Wasser gekocht.

Gehen wir das mal analytisch an. Innerhalb von <crosspoints> scheint es ein paar andere relevante Tags zu geben. Na, also: Richtig erkannt. Und es handelt sich dabei nur um zwei

- <crosspoint> und
- <track-info>

[7] Engl.: *What the fuck*, Ausruf des Erstaunten...;-)