knottyTom's MTB Tour Guide System
Handbuch 0.2 für Version ab 0.5.2

knottyTom@berlios.de

13. September 2006

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 0.5.1 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 0.5.1

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, ...

1.5 Ist-Zustand der Version 0.5.2

1.6 Weitere Informationsquellen

Für jegliches Feedback bin ich natürlich dankbar.

Ein zentraler Anlaufpunkt ist die Projektseite[1] des Projekts auf BerliOS. Neuigkeiten gibt es auch auf der Homepage[2] und die neuesten Versionen im Download-Bereich[3]. Man kann sich natürlich auch auf einer der Mailinglisten[4] registrieren oder die Foren[5] besuchen.

Ihr könnt mir natürlich auch ganz zwanglos eine Email schreiben. Die Adresse ist auf dem Deckblatt angegeben.

[1] $http://developer.berlios.de/projects/kttgs/$
[2] $http://kttgs.berlios.de/$
[3] $http://kttgs.berlios.de/\#download$
[4] $http://developer.berlios.de/mail/?group_id = 4839$
[5] $http://developer.berlios.de/forum/?group_id = 4839$

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[6] 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 [7] zu finden. Zum Zeitpunkt der Erstellung dieser Dokumentation ist die Version 1.6.5 die aktuellste. Das sogenannte *Binary Release* genügt für unsere Zwecke.

[6] http://java.sun.com/j2se/1.5.0/download.jsp
[7] 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://kttgs.berlios.de/#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.

Überraschung! Das war alles.

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.

Wenn ihr in dem Verzeichnis eine Datei namens `build.xml` seht, dann seid ihr vermutlich richtig.

Alle Aufrufe von `ant`, die hier verwendet werden, sind im Abschnitt 6 ab Seite 25 detailliert beschrieben.

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 muss man konfigurieren. Man kann das nun mit einem Texteditor tun: Das ist im Abschnitt 4.1.1 beschrieben.

Man kann sich das Leben aber auch leichter machen und diese Datei mithilfe der grafischen Oberfläche verändern. Das ist im Abschnitt 4.1.2 erläutert. Wenn ihr die Oberfläche verwendet, dann braucht ihr den Abschnitt über den Editor nicht zu lesen... aber vielleicht interessiert es dich doch, wie das System funktioniert.

Die Oberfläche hat den Vorteil, daß Fehler (nahezu?) ausgeschlossen sind.

4.1.1 Mit dem Editor

Die erwähnte, zentrale Steuerungsdatei heißt `./build.properties`. Bitte öffnet und editiert diese mit eurem Lieblingseditor[8].

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
```

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

4.1.2 Mit der grafischen Oberfläche

Die Oberfläche sollte intuitiv bedienbar sein. Du gibst einfach die Daten ein und drückst den Knopf CREATE. Falls irgendetwas nicht korrekt ist, dann bekommst du eine entsprechende Fehlermeldung. Wenn keine Fehlermeldung erscheint, dann klickst du auf OK und *der Käse ist gegessen*. Keine Hemmungen! Man kann nichts kaputt machen...

Wichtig! In dem Textfeld TOUR FOLDER NAME gibst du bitte den Text `MeineSuperTour` ein. Wenn du da was anderes verwendest, dann passt die nachfolgende Beschreibung nicht. Also: Mach's dir leicht und mach' was ich dir gesagt habe...

Zum Starten der Oberfläche gibst du auf der Kommandozeile den folgenden Befehl ein:

```
1 ant run.config
```

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.

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[9]. Wenn ihr jetzt 'ne Beschreibung einer fiktiven Tour seht, dann läuft das System. **Glückwunsch!**

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

4.5 Die Tour per eMail verschicken

So, die Tour[10] hätten wir nun. Aber wie kann man die Beschreibung einem Freund zukommen lassen? Auch das ist einfach, man packt alles in eine gezippte Datei und verschickt sie per eMail. Wie gehen mal wieder auf die Kommandozeile und geben den nachfolgenden Befehl ein:

```
1 ant zip-tour-html
```

Jetzt sollte im aktuellen Verzeichnis eine gezippte Datei entstanden sein. In unserem (Beispiel-)Fall heißt sie `MeineSuperTour.zip`. Sie beinhaltet alles, was man zur Betrachtung der Tour im HTML-Browser benötigt. Der Empfänger der eMail muß nur die Datei in einem neu angelegten Verzeichnis auspacken und kann sich die Tourbeschreibung angucken.

[10] Wie man die Tour erfasst wird im Abschnitt 5 ab Seite 11 beschrieben.

5 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[11].

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 ergänzt. Diese gehören natürlich ***nicht*** *in die echte Datei!*

5.1 Learning By Doing

Das, was ich nun beschreiben werde, ist etwas trocken. Ich hab' zwar versucht, es etwas aufzulockern, aber es muß ja trotzdem technisch korrekt sein und das führt zwangsläufig zu einer gewissen *Trockenheit*.

Interessanter is es sicher, die ganzen Sachen während des Lesens auch auszuprobieren. Dann ist das Ganze nicht ganz so trocken und man sieht gleich, ob man alles richtig verstanden hat.

Das ist sehr einfach mit dem System möglich: Als erstes legt man mal ein Skeleton an, wie ich es in Abschnitt 4.2 (Seite 9) beschrieben habe. Falls du das noch nicht getan hast, dann solltest du jetzt wirklich mal den kompletten Abschnitt 4 lesen.

Nun kannst du mit dem Skeleton experimentieren. Hast du die Namen aus Abschnitt 4 verwendet, dann befindet sich eine Datei names `MeineSuperTour.xml` im Verzeichnis `./xml/MeineSuperTour`. Dies ist die XML-Datei, um die sich alles dreht. Du editierst diese Datei und überprüfst die Änderungen, indem du folgendes ausführst:

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

Um die Änderungen im Browser sichtbar zu machen, musst du die Ansicht in deinem Browser aktualisieren. Welche Datei du in den Browser laden musst, ist in Abschnitt 4.4 auf Seite 9 beschrieben.

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

5.2 <tourguide>

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

```
<tourguide>
    <storage-info>
        <location />
    </storage-info>
    <general>
        ...
    <general>
    <crosspoints>
        ...
    </crosspoints>
</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...

5.3 <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:

```
<tourguide>
    <storage-info>
        <location
            country="de"
            region-or-state="Bavaria"
            city="Buttenheim"
            zip-code="96155"/>
    </storage-info>
     ...
</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>`...

5.4 <storage-info> (<authoring>)

Natürlich sollte sich auch der Autor der Tour verewigen können. Dazu dient das Tag <authoring> innerhalb von <storage-info>.

Im Attribute `author` sollte der Autor seine eMail-Adresse angeben. Dann kann man ihm bei Unklarheiten eine eMail schicken.

Man sollte auch im Attribute `creation-date` das Erstellungsdatum der Tour angeben. Es könnte ja sein, daß jemand die Tour 10 Jahre nach der Erfassung abfahren will. Dann könnte es durchaus sein, daß nicht mehr alle Angaben (auch eine 100-jährige Eiche kann mal gefällt worden sein) so stimmen. Der Leser kann sich dann darauf einstellen.

Als Datumsformat wird das Muster `YYYYMMDD` [12] verwendet. D.h., daß der 27. Januar 2006 als `creation-date="20062701"` notiert wird.

```
1 <storage-info>
2    <location ... />
3
4    <authoring
5        author="email@not-real.com"
6        creation-date="YYYYMMDD"    />
7 </storage-info>
```

5.5 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 lie-

[12] 4stelliges Jahr, 2stelliges Monat und 2stelliger Tag.

gen kommt, existiert [13] ein Unterverzeichnis namens `images`. Hier hinein werden 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:

```
1 <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.

5.6 <general>

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

```
1  <tourguide>
2      ...
3  <general
4      fitness-level="easy"
5      tech-level="medium"
6      type="round"
7      duration="01:00hrs"
8      total-uphill="600m">
9
10     <name>Nuernberger Tiergarten
11           --- Kleiner Trial inklusive 'Roller Coaster'
12     </name>
13     <distance unit="km">10</distance>
14     <desc>
15         Wir starten am Tiergarten, fahren etwas uphill durch
16         die Stromschneisse, dann den 'Mini-DH' runter...
17     </desc>
18     <reach>
19        <option name="Mit dem Auto">Auf der A3 aus Richtung
20           Wuerzburg oder Passau kommend bei Nuernberg/Moegeldorf
21           abfahren...
22        </option>
23        <option name="Per Zug">Am Nuernberg Hbf aussteigen. Dann
24           mit der Strassenbahn direkt zum Tiergarten.
25        </option>
26     </reach>
27     <maps>
28        <map>Kompass, Karte 170</map>
29     </maps>
30     <roadmap image="map.jpg"/>
31     <profile />
32 </general>
33     ...
```

[13] Wenn das das Grundgerüst via `ant gen-skeleton` angelegt wurde, dann existiert es automatisch. 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.

```
34 </tourguide>
```

5.6.1 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 für 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.
5. total-uphill: Hoehenmeter der Tour.

Werte für **fitness-level**: `easy, medium, high` oder `extrem`.

Werte für **tech-level**: `easy, medium, high` oder `extrem`.

Werte für **type**:

- round ⇒ Rundtour
- pointtopoint ⇒ Tour von A nach B
- downhill ⇒ Reiner Downhill
- trail ⇒ Schöner Singletrail

duration: Hier sollte man sowas wie `"01:30hrs"` oder `"1 Stunde, 30 Minuten"` angeben.

total-uphill: Hiermit werden die gesamten Höhenmeter der Tour angegeben. Also sowas wie `"600m"`. Vergesst die die Einheit wie z.B. `m` für Meter nicht. Es koennte sich ja auch um englische Fuß handeln.

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

5.6.2 <name>

Hier vergibst du einen kurzen, prägnanten Namen für die Tour. Dieser Text taucht zum Beispiel in Listen auf, wenn man aus mehreren Touren eine auswählen muß.

5.6.3 <distance>

Dieses Tag dient zu Angabe der Gesamtdistanz der Tour, die dein Fahrradcomputer angezeigt hat, als du sie gefahren bist. Es wird folgendermaßen verwendet:

```
<tourguide>
    ...
<general
    ...
   <distance unit="km">10</distance>
    ...
</general>
    ...
</tourguide>
```

Mit dem Attribute `unit` gibst du die Längeneinheit an (also z.B.: km, meter, Meilen). Zwischen den beiden Tags notiert man dann die Länge der Tour als Zahl. Und das war's auch schon.

5.6.4 <desc>

Zwischen den <desc>-Tags wird eine Beschreibung der Tour erfasst. Die kann durchaus etwas länger sein und soll dem Leser richtig Lust auf die Tour machen.

Die einzelnen Absätze werden durch <para> eingeschlossen. Damit kann man (im Moment nur) die Ausgabe der HTML-Datei etwas hübscher formatieren. Das Tag muss aber benutzt werden. Auch wenn nur `ein` Absatz vorliegt! Hat man das Tag vergessen, dann erscheint in der Ausgabe keinerlei Text.

5.6.5 <reach>

Irgendwie muß der Biker ja auch an den Startpunkt der Tour gelangen. Dazu verwenden wir das <reach>-Tag. Man kann mehrere Verkehrsmittel angeben. Ein Beispiel köennte wie folgt aussehen:

```
<tourguide>
    ...
<general
    ...
   <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>
    ...
</general>
    ...
</tourguide>
```

Innerhalb des <reach>-Tags kann man beliebig oft das Tag <option> verwenden. Das <option>-Tag dient dazu, die verschiedenen Verkehrsmittel bzw. Anreisewege zu beschreiben. Das Attribute `name` verwendet man dazu, den Namen des Verkehrsmittel zu notieren. Innerhalb des <option>-Tag erfolgt dann die Erklärung zur Anreise.

5.6.6 <maps>

Zum Abradeln der Tour wird man sicher 'ne Landkarte des Gebiets gebrauchen können. Auch diese Information sollten wir dem Leser mitgeben. Innerhalt des <maps>-Tags kannst du mit dem <map>-Tag beliebig viele Karten angeben:

```
<tourguide>
    ...
<general>
    ....
    <maps>
        <map>Kompass, Karte 170</map>
        <map>Andere Firma, Andere Karte 21</map>
         ...
    </maps>
    ...
</general>
    ...
</tourguide>
```

5.6.7 <roadmaps>

Du hast auch eine digitalisierte Karte der Tour (mit eingezeichnetem Weg der Tour)? Na, dann sollten wir die auch anzeigen. Dazu genügt es, den Namen des Bildes im `image` Attribute des <roadmap>-Tags anzugeben:

```
<tourguide>
    ...
<general>
    ...
    <roadmap image="map.jpg"/>
    ...
</general>
    ...
</tourguide>
```

Aber Achtung! Kartenmaterial ist urheberlich geschützt. Du solltest (ohne Genehmigung) nicht einfach eine Karte einscannen und deiner Tour beilegen. Sofern du die Karte (wie auch immer) selbst gezeichnet oder eine Genehmigung hast, dann kannst du sie ohne Bedenken verwenden.

5.6.8 <profile/>

Das ist ein seltsames Tag: Man muss es nämlich einfach nur hinschreiben... und man kann es auch einfach weglassen. Aber dann fehlt eine wichtige Information,

die noch dazu automatisch generiert werden wuerde: Das grafische Profil der Tour.

Da das Tag allein für sich steht sollte man den Slash/Schrägstrich (`.../>`) vor der schließenden Klammer nicht vergessen!

5.7 <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">
     <desc><para>Ein nordwest Punkt.</para></desc>
     <profile-desc>
        Die Stromschneisse... und los
     </profile-desc>
   </crosspoint>

   <track-info pavement="road">
      <anecdote distance="gesamter Weg" image="drop.jpg">
          Wenn's gar nicht mehr geht, dann einfach ab in
          den Wald. Der ist nicht dicht... man kann da
          auch gut schieben (und wird nicht gesehen ;-).
      </anecdote>
   </track-info>

   <crosspoint distance="200" elevation="310"
               direction="none"
               latitude="000" longitude="000">
     <desc><para>Back At Home!</para></desc>
   </crosspoint>
</crosspoints>
</tourguide>
```

WTF[14]? 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>

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

Anm. d. A.: Dem häufigen Leser technischer Dokumente wird nun auffallen — und ihn vielleicht verwirren —, daß ich die Einschachtelungstiefe des XML-Dokuments auch durch die Struktur dieses Dokuments wiedergegeben habe. Dies wird nun bewußt durchbrochen. Es ist der schiere Umfung und die Strukturtiefe der nachfolgenden zwei Abschnitte, die dieses Vorgehen vernünftig erscheinen lassen.

5.8 Entfernungen

Folgt man 'ner fremden Tour, dann ist es natürlich wichtig zu wissen, wo auf der Strecke man gerade ist. Viele Touren geben dazu die Positionen der Kreuzungspunkte in der totalen Distanz zum Startpunkt an. Das funktioniert natürlich theoretisch auch: In einer perfekten Welt.

Aber in einer perfekten Welt misst mein Fahrradcomputer auch perfekt. Und: Ein perfekter Fahrer verfährt sich nie.

Meine Realität sieht aber so aus: Bis zum Kreuzungspunkt 3 hat alles gepasst. Den Kreuzungspunkt 4 finde ich aber nicht. Ich bin schon 1000 Meter darueber. Also: Zurück zum Punkt 3. Ok, ich nehm' den anderen Weg. Nun bin ich also 2x1000 Meter über der Angabe in der Tourbeschreibung. Das muss ich natürlich bei allen weiteren Angaben in Betracht ziehen.

knottyTom's Tour Guide System zaehlt deswegen anders. Alle Entfernungen werden im Abstand zum letzten Kreuzungspunkt angegeben.

5.9 <crosspoint>

Das <crosspoint>-Tag dient dazu, einen Kreuzungspunkt im Verlauf der Tour zu beschreiben. Dazu verwenden wir fünf Attribute und zwei Subtags.

```
<tourguide>
...
<crosspoints>
   <crosspoint
      distance="300" elevation="600"
      direction="northwest"
      latitude="000" longitude="000"
      showInProfile="no">

     <desc><para>Ein nordwest Punkt.</para></desc>
     <profile-desc>
        Die Stromschneisse... und los.
     </profile-desc>
   </crosspoint>

   <track-info pavement="road">
      ....
   </track-info>

   <crosspoint distance="200" elevation="310"
               direction="none"
               latitude="000" longitude="000">
```

```
23        <desc><para>Back At Home!</para></desc>
24      </crosspoint>
25  </crosspoints>
26  </tourguide>
```

5.9.1 <crosspoint> Attribute

Die Attribute (und ihre Bedeutung) lauten wie folgt.

- `distance`: Das ist der Abstand zum letzten Kreuzungspunkt in **Metern**. Der erste <crosspoint> hat also immer die `distance="0"` (Null)!
- `elevation`: Höhe des Punktes in **Metern**.
- `direction`: Hiermit gibt man die Richtung an, in die es nach diesem Punkt weitergeht, mit einer von acht Himmelsrichtungen an. Mögliche Werte sind: `north, east, south, west, northeast, southeast, southwest` und `northwest`.
- `latitude`: Breitengrad (*wird bisher nicht verwendet*)
- `longitude`: Längengrad (*wird bisher nicht verwendet*)
- `showInProfile`: Dieses Attribute gibt an, ob der Kreuzungspunkt im grafischen Profil als Nummer erscheinen soll. Wird das Attribute nicht verwendet, dann erscheint der Punkt im Profile. Das ist gleichedeutend mit `showInProfile="yes"`. Liegen die die Kreuzungspunkt im grafischen Profil sehr dicht beisammen, dann sieht das nicht gut aus und ist auch nicht gut zu lesen. In diesem Fall sollte man `showInProfile="no"` angeben, dann wir die Nummer weggelassen. Einfach mal ausprobieren! ;-)

5.9.2 <desc>

Dieses Tag dient der Beschreibung des Kreuzungspunktes. Das ist, auch wenn man das nicht gleich erkennt, eines der wichtigsten Tags der ganzen Beschreibung!

An so einem Punkt muß sicher Fahrer wieder neu orientieren und überprüfen, ob er noch auf der richtigen Route ist. Gib ihm/ihr an dieser Stelle Hinweise, wie es jetzt weitergeht. Vorallem dann, wenn sich der Weg gabelt oder es irgendwo nicht leicht ersichtlich in den Wald geht. Du solltest also so Angaben machen wie:

- *Hier gabelt sich der Weg in drei Schotterwege. Wir nehmen den mittleren.*
- *Hinter der alten Eiche geht es rechts in den Waldweg.*
- *Bisher waren wir auf "gelb Kreis", jetzt fahren wir auf "rot Viereck"weiter.*

Jeden Absatz muss man wieder innerhalb von <para>...</para> schreiben. Vergisst man die <para>-Tags, dann sieht man wieder keine Textausgabe im generierten Dokument!

5.9.3 <images>

Innerhalb des <images>-Tag kann man bieliebig viel Bilder, die für den Kreuzungspunkt relevant sind, angeben.

```
<tourguide>
...
<crosspoints>
   <crosspoint...>
     <desc><para>Ein nordwest Punkt.</para></desc>
     <images>
        <image name="einBild.jpg"/>
        <image name="anderesBild.jpg"/>
         ....
     <images>
   </crosspoint>
    ....
</crosspoints>
</tourguide>
```

5.9.4 <profile-desc>

Wie ich ja schon erwähnt habe, kann das System automatisch ein grafisches Profil der Tour aus den erfassten Daten generieren. Dazu gehört auch eine Legende, die die einzelnen Punkt des Profil kommentiert.

Dazu wird normalerweise der Text aus dem <desc>-Tag verwendet. Dieser kann aber oftmals zu lang oder einfach unpassend für eine Legende sein.

Deshalb kann man mithilfe des <profile-desc>-Tags diesen Text überschreiben. Das Tag ist zwar optional, ich würde aber dennoch Gebrauch davon machen.

5.10 <track-info>

Innerhalb des <trackinfo>-Tags werden Daten erfaßt, die zwischen zwei Kreuzungspunkten von Interesse füer den Fahrer der Tour sind, bzw. sein könnten.

Dazu gehören z.B. Sachen wie: Wie schaut der Straßenbelag aus? Wo gibt's was zu futtern? Gibt's auch was zu sehen? Geschichten zur eigenen Tour.

Den Straßenbelag gibt man im Attribute `pavement` des Tags an also z.B. so `<trackinfo pavement="trail»` an. Erlaubte Werte und ihre Bedeutung sind:

- `road` ⇒ Asphaltierte Straße
- `forrest.road` ⇒ Wald- oder Feldweg
- `trail` ⇒ Schmaler Wanderweg

Das Ganze sieht also z.B. wie folgt aus:

```
<tourguide>
...
```

```
<crosspoints>
    ...
    <track-info pavement="road">
        ...
    </track-info>
    ...
</crosspoints>
</tourguide>
```

Innerhalb des <track-info>-Tags sind die Tags <eatndrink>, <poi> und <anecdote> erlaubt, aber allesamt sind optional. Sie haben alle zwei einheitliche Attribute. Sehen wir uns die als Erstes an.

5.10.1 Attribute für <eatndrink>,<poi> und <anecdote>

Wir haben die beiden Attribute:

- `distance` und
- `image` (optional)

Mit `distance` wird die Entfernung zum letzten Kreuzungspunkt angegeben. Das ist jetzt **nicht** der Kreuzungspunkt zu dem dem dieser `<track-info>`-Abschnitt gehört, sondern der **vor** diesem! Genaueres dazu findest du in Absschnitt 5.8 auf Seite 19.

`image` dient dazu ein Bild anzugeben. Es gilt alles das, was ich in Abschnitt 5.5 auf Seite 13 zu Bildern geschrieben habe.

5.10.2 <eatndrink>

Die Leute die deine Tour abradeln, werden sicher Hunger und Durst bekommen. Hilf ihnen und beschreib', wo es was zu Futtern gibt. Die Beschreibung erfolgt innerhalb der Tags. Hier ein Beispiel:

```
<eatndrink distance="1200" image="dead_sow.jpg">
    "Zur␣Toten␣Sau" ist ein uriges und guenstiges fraenkisches
    Lokal. Man sollte unbedingt das "Schaeufala" probieren.
    Fuer Vegetarier... ne, fuer solche Menschen gibt es da
    aber schon gar nix. >;-)
</eatndrink>
```

5.10.3 <poi>

POI steht fuer *point of interest*, also ein 'interessanter Punkt'. Das kann nun sehr vieles sein: Ein uralte Eiche. Reste eines römischen Walls. Ein Wasserfall. Eine Felsformation.

Ich finde man sollte mit dem MTB nicht nur durch die Gegend rasen, sondern sich auch (hin und wieder) die Gegend ansehen, durch die man fährt. Wenn ich

nur möglichst schnell unterwegs sein möchte dann könnte ich mir ja auch ein Rennrad anschaffen...

Mit <poi> kann man auf solche Punkte hinweisen, z.B. so:

```
<poi distance="400" image="rocks.jpg">
    Hier befindet man sich auf der sog. "Retterner Kanzel".
    Man sollte sich wirklich die Zeit nehmen und mal fuer
    5 Minuten "ins Land" zu blicken...
</poi>
```

5.10.4 <anecdote>

Ihr habt sicher ein paar Sachen beim Erkunden und Erfassen der Tour erlebt, die der Nachwelt erhalten bleiben sollten, auch wenn sie nicht relevant für die Tour selber sind. Egal, notiert auch das im <anecdote>-Tag z.B. wie folgt:

```
<anecdote distance="640" image="broken_bike.jpg">
    Bei der Abfahrt hat es mich so zerlegt, dass ich mein
    Bike wegschmeissen konnte. Und Martin hatte nichts
    Besseres zu tun als mein Leid auch noch zu
    fotografieren... ;-)
</anecdote>
```

5.10.5 <profile-points>

Wenn man eine Tour erfasst, dann müssen die Höhenangaben an den Kreuzungspunkten nicht unbedingt das echte Profil widerspiegeln. Es kann durchaus vorkommen, daß zwischen zwei Kreuzungspunkten ein echter Hammer liegt.

Das können wir bisher aber nicht erfassen. Deswegen gibt es am Ende des `<track-info>`-Abschnitts den `<profile-points>`-Bereich. Hier können Profilpunkte notiert werden, die zwischen zwei Kreuzungspunkten auftreten.

Das sieht dann z.B. so aus:

```
<tourguide>
...
<crosspoints>
   ...
   <track-info pavement="road">
      ...
      <profile-points>
        <point distance="500" elevation="300"
               latitude="000" longitude="000"
               icp="false"/>
        <point distance="1000" elevation="150"
               latitude="000" longitude="000"
               icp="false"/>
      </profile-points>
   </track-info>
   ...
</crosspoints>
</tourguide>
```

Jeder Profilpunkt wird mit einem `<point>`-Tag erfasst. Die Bedeutung der Attribute `distance, elevation, latitude` und `longitude` entspricht der eines `<crosspoint>`-Tags (siehe dazu auf den Punkt 5.9.1 auf Seite 20). Neu ist eigentlich nur das Attribute `icp`. Es muss immer vorhanden sein und immer den Wert `false` haben. Das ist seltsam. Stimmt! Das ist im Moment aber aus technischen Gründen nötig.

Mit `distance` wird die Entfernung zum letzten Kreuzungspunkt angegeben. Das ist jetzt **nicht** der Kreuzungspunkt zu dem dem dieser `<track-info>`-Abschnitt gehört, sondern der **vor** diesem! Genaueres dazu findest du in Absschnitt 5.8 auf Seite 19.

5.11 Keine Tags mehr!

Das war heftig, oder? Aber ich hab' eine positive Überraschung: Das waren alle Tags. Jetzt solltest du alles kennen was zur Erfassung einer Tour notwendig ist.

Falls du dennoch Probleme haben solltest, dann schick' mir halt mal eine eMail an die Adresse, die auf dem Deckblatt angegeben ist. Vielleicht kann ich dir ne Lösung schicken und diese Dokumentation verbessern.

Falls du es geschafft hast, eine Tour zu erfassen, dann kannst du die zip-Datei der Tour vielleicht auch irgendwo im Internet hochladen. Ich versuche einen Index aller Touren aufzubauen, die mit KTTGS erfassst worden sind. Schick' mir aber bitte nur den Link/URL zu der zip-Datei, sonst läuft womöglich meine Mailbox über.

6 Aufrufen verschiedener Funktionen über *Ant*

6.1 Neue Tour vorbereiten

Nutzung der grafischen Oberflaeche: lstsetnumbers=left,numberstyle=,stepnumber=1

```
1 ant run.config
```

6.2 Grundgerüst/Skeleton anlegen

```
1 ant gen-skeleton
```

6.3 Web-Seite generieren

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

6.4 Tour für eMail-Versendung vorbereiten

```
1 ant zip-tour-html
```

7 Tips und Tricks

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

7.1 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

7.2 Formulare verwenden (TODO)

7.3 Tourerfassung mit MP3-Player (TODO)