Webprogrammierung mit C++ - Tntnet macht es möglich

Was ist Tntnet?

- Server für C++ Webapplikationen
- Open source (LGPL)
- Unix/Linux

Eigenschaften von Tntnet

- Templatesprache zum Einbetten von C++ in HTML
- Seiten werden compiliert
- Nutzt umfangreiche C++ Bibliothek cxxtools

Warum C++

- Ausgereifte Sprache mit großem Funktionsumfang (Klassen, Templates, Destruktoren ...)
- Schnelle und kompakte Programme
- ISO-Standard (Investitionsschutz)
- Stabil (nicht jedes Jahr neue oder geänderte Features)

Technik von Tntnet

- Eigenständiger Server
- Multithreaded
- Hochoptimiert
- Skalierbar
- SSL/TLS Unterstützung mit OpenSSL oder GnuTLS
- Als Modul oder eigenständige Applikation übersetzbar

Features

- Flexibles URL mapping mit regulären Ausdrücken
- Binärdaten und andere statische Resourcen können mit compiliert werden
- Fehlerbehandlung durch Exceptions
- Flexibles Logging für Fehlersuche
- Templatesprache

Templatesprache ECPP

- Einbetten von C++ in HTML
- Präcompiler übersetzt ecpp in C++
- Alle C++ Features nutzbar
- Reichhaltige API
- „policy free“
  - Der Entwickler hat die Freiheit aber auch die Verantwortung, die Applikation so zu strukturieren, wie er es braucht

ECPP Features

- Scoped variable: application, session, request, thread scope
- Einfacher Zugriff auf Query parameter (GET, POST)
- Unterstützung für http upload, Cookies, ...
- Automatisches Übersetzen von html entities

Die wichtigsten ECPP tags

- `<$...$>` Ausgabe eines C++-Ausdrucks
- `<%cpp>...</%cpp>` C++-Verarbeitungsblock
- `<%args>...</%args>` Formular-Parameter
- `<%pre>...</%pre>` für #include-Direktiven
- `<& component >` Komponentenaufruf
- `<#...#>` Kommentar

Beispiel


```
<#
  this is a simple hello-world-application
#>
<%args>
name;             // define query-parameter
                  // this defines a variable of type std::string with
                  // the name "name"
</%args>
<html>
 <head>
  <title>Hello World-application for tntnet</title>
 </head>

 <body bgcolor="#FFFFFF">
  <img src="tntnet.jpg" align="right">

  <h1>Hello <$ name.empty() ? "World" : name $></h1>

  <form>
   What's your name?
   <input type="text" name="name" value="<$name$>"> <br>
   <input type="submit">
  </form>

 </body>
</html>

~
                                                    <2          Alles
```

Was gibt es noch?

Cxxtools

- Basisbibliothek von Tntnet
- Umfangreiche Sammlung von C++ Klassen

Eigenschaften von cxxtools

- Serialisierungsframework (Json, Xml, ...)
- RPC (xmlrpc, jsonrpc, binary)
- HTTP Client und Server
- Logging, Unittest
- Netzwerk, Threading, async I/O
- Unicode support
- Und vieles mehr ...

Beispiel: json rpc server


```
#include <iostream>
#include <cxxtools/arg.h>
#include <cxxtools/json/rpcserver.h>
#include <cxxtools/eventloop.h>

double add(double a1, double a2)
{
  return a1 + a2;
}

int main(int argc, char* argv[])
{
  try
  {
    cxxtools::Arg<std::string> ip(argc, argv, 'i');
    cxxtools::Arg<unsigned short> port(argc, argv, 'p', 7004);
    cxxtools::EventLoop loop;
    cxxtools::json::RpcServer jsonServer(loop, ip, port);

    jsonServer.registerFunction("echo", echo);
    jsonServer.registerFunction("add", add);

    loop.run();
  }
  catch (const std::exception& e)
  {
    std::cerr << e.what() << std::endl;
  }
}
```

Beispiel: json rpc client


```
#include <iostream>
#include <cxxtools/arg.h>
#include <cxxtools/remoteprocedure.h>
#include <cxxtools/json/rpcclient.h>

int main(int argc, char* argv[])
{
  try
  {
    cxxtools::Arg<std::string> ip(argc, argv, 'i');
    cxxtools::Arg<unsigned short> port(argc, argv, 'p', 7004);

    cxxtools::json::RpcClient client(ip, port);
    cxxtools::RemoteProcedure<double, double, double> add(client, "add");

    std::cout << add(17, 4) << std::endl;
  }
  catch (const std::exception& e)
  {
    std::cerr << e.what() << std::endl;
  }
}
```

Tntdb

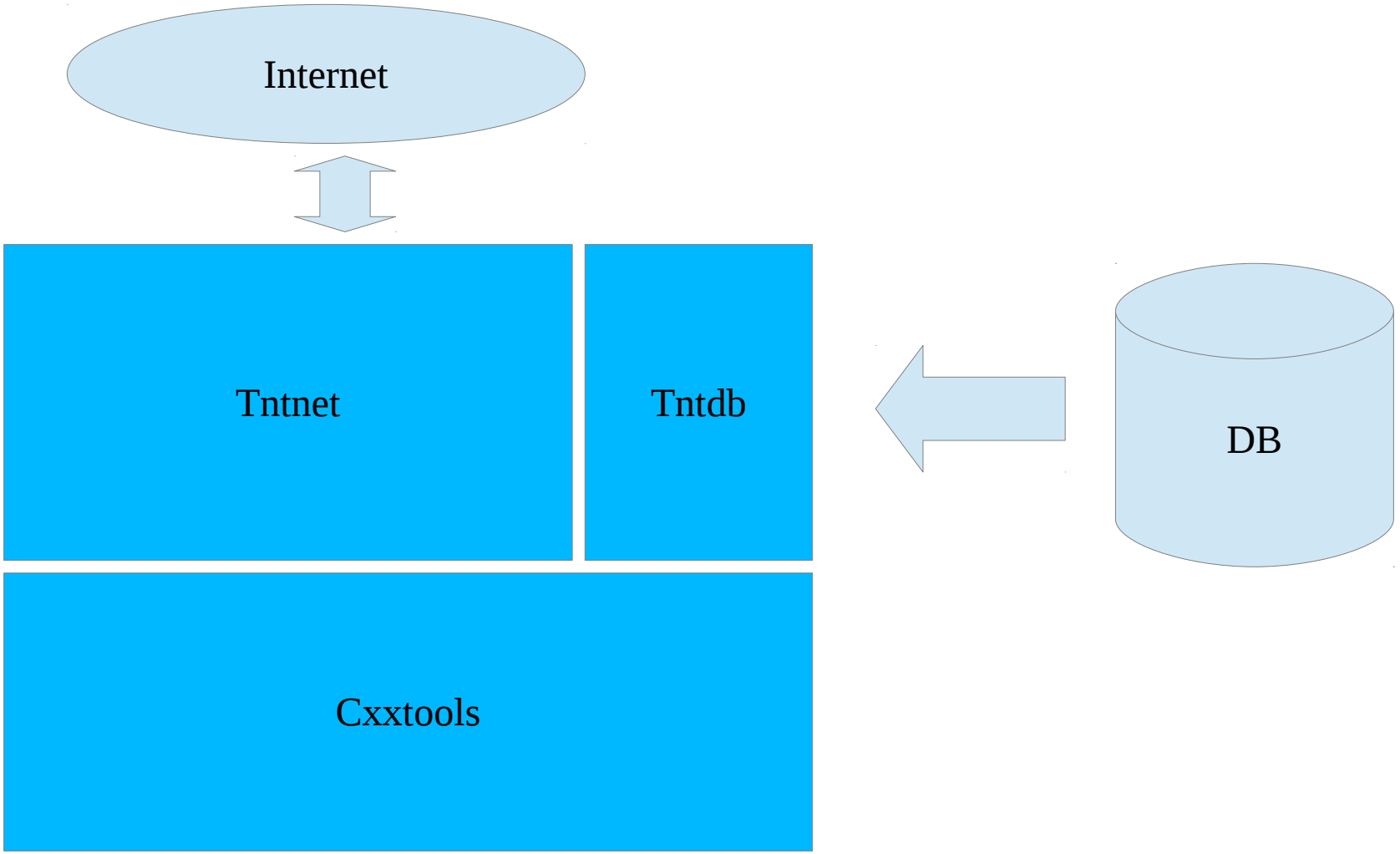
- Datenbankzugriff mit C++ war noch nie so einfach
- Datenbankunabhängige API
- Treiber für Postgresql, Sqlite, Mysql und Oracle

Tntdb Beispiel


```
#include <exception>
#include <iostream>
#include <tntdb/connect.h>
#include <tntdb/statement.h>

int main(int argc, char* argv[])
{
  try
  {
    tntdb::Connection conn = tntdb::connect("sqlite:mydb.db");
    tntdb::Statement st = conn.prepare(
      "select per_pid, per_vorname, per_nachname"
      "  from person"
      " where per_vorname like :Name"
      "    or per_nachname like :Name");

    st.set("Name", "Tommi%");
    for (tntdb::Statement::const_iterator cur = st.begin(); cur != st.end(); ++cur)
    {
      tntdb::Row row = *cur;

      unsigned pid;
      std::string nachname;
      std::string vorname;

      row[0].get(pid);
      row[1].get(nachname);
      row[2].get(vorname);

      std::cout << pid << '\t' << vorname << ' ' << nachname << '\n';
    }
  }
  catch (const std::exception& e)
  {
    std::cerr << e.what() << std::endl;
  }
}
```

< Anfang

Zusammenfassung



Kontakt

- http://www.tntnet.org/
- E-Mail: tommi@tntnet.org
- IRC: Freenode #tntnet
- Mailingliste: tntnet-general@lists.sourceforge.net

Fragen?

Vielen Dank für die Aufmerksamkeit