

Erstellung performanter Webapplikationen mit Tntnet



- Schneller und sicherer durch compilierten Code
im Web
 - Tommi Mäkitalo



Was ist Tntnet?

- Web-Applikations-Server
- Template-Sprache zum einbetten von C++ in HTML
- Seiten werden zur Compilezeit übersetzt
- keine Interpretation zur Laufzeit mehr erforderlich
- eigenständiger Server



Warum C++

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



Nachteile von C++

- Entwicklung umständlich, da Compilerlauf notwendig
- Momentan weniger populär als z. B. Java
- Standardbibliothek nicht so umfangreich



Webapplikationen

- Deployment einfach
- skalierbar durch Client-Server-Architektur
- skalierbar durch nicht persistente Connections
- schlanke Clients



Webapplikationen: PHP

- interpretiert (langsam, grosser Ressourcenverbrauch)
- einfach
- Sicherheit nicht optimal (Sourcecode und Interpreter auf dem Server verfügbar)
- populär, billige Hoster verfügbar



Webapplikationen: CGI

- CGI mit Perl
 - langsam, da für jeden Request Prozess gestartet werden muß
 - umständlich, da keine Templates
 - CPAN hilfreich
 - Template-engines verfügbar
- CGI mit C oder C++
 - auch nicht viel schneller



Webapplikationen: Java

- recht schnell
- Speicherintensiv
- Templates mit JSP verfügbar
- Mainstream
- Deployment gut gelöst (JAR/WAR)
- Java als Sprache eingeschränkt
- proprietär, instabile Spezifikation



Webapplikationen: Tntnet

- Tntnet verbindet C++ mit dem Web
- Applikationen sind compiliert, daher schnell, sicher und kompakt
- Der Content wird grundsätzlich mit Templates erstellt (ecpp)
- C++-Klassen und -Bibliotheken uneingeschränkt nutzbar
- portabler Sourcecode verfügbar



Tntnet - Technik

- standalone Webengine
- multithreaded (skalierbar)
- persistente Applikationen
- optimierter C++-Code
- kompakt
- automatische Http-Komprimierung
- SSL-Unterstützung (gnutls, openssl)



Features

- automatisches Sessionmanagement
- Session-, Request- und Applicationscope für beliebige C++-Objekte mit automatischen locking
- Komponentenorientierte Entwicklung
- Request-paramter werden geparst
- File-upload, Cookies, i18n, ...



Features (2)

- Grafiken und andere Binärdaten können mit compiliert werden
- Compiler oder Interpreter auf dem Server nicht notwendig
- Fehlerbehandlung durch exceptions
- portabel durch die Verwendung von Standard-C++ (Linux, Unix)
- Logging



Features (3)

- Automatisches HTML-encoding
- savepoint: nehme Ausgabe im Fehlerfall bis zu einem sicheren Punkt zurück, um vollständige HTML-Seiten zu erhalten

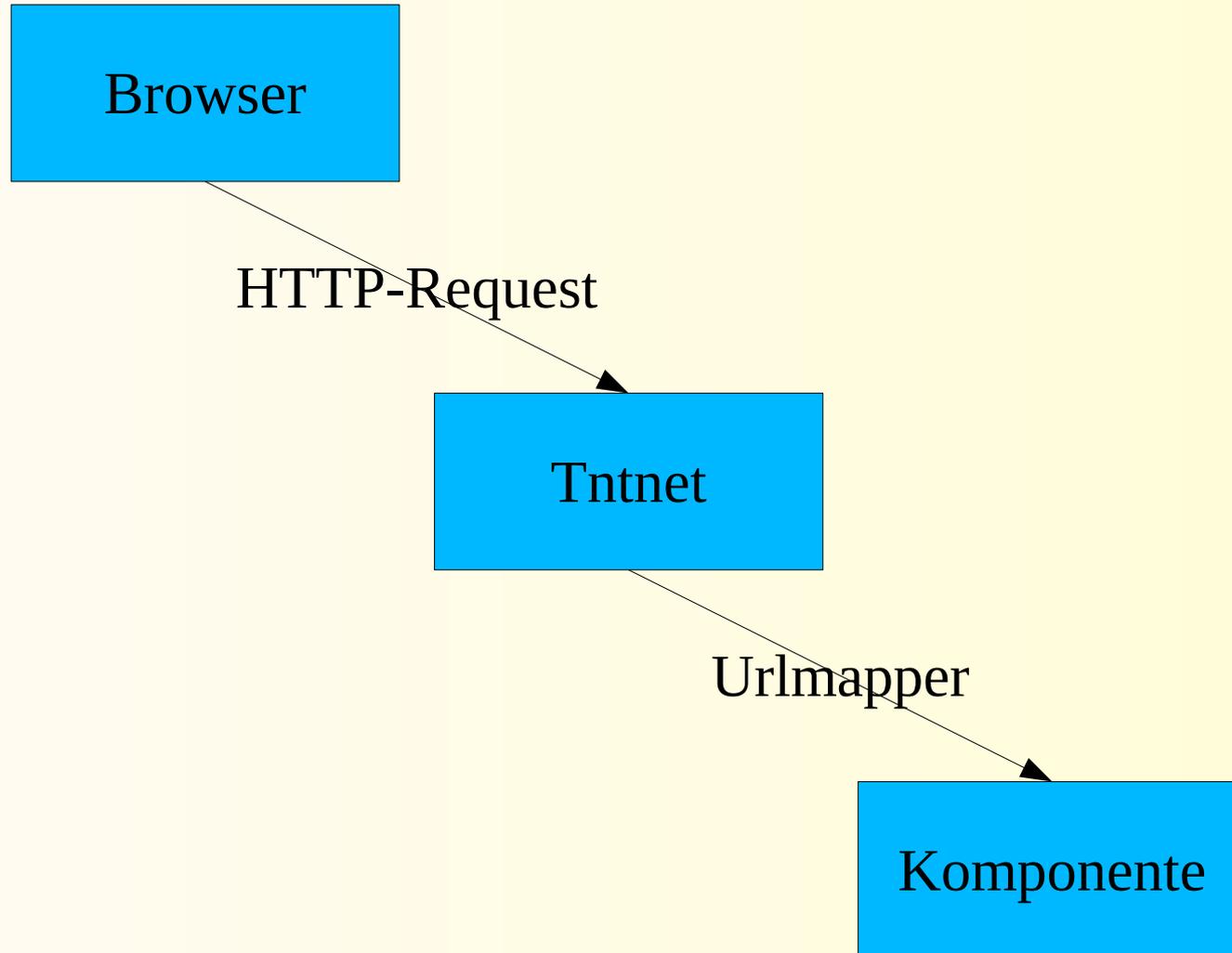


Nachteile von Tntnet

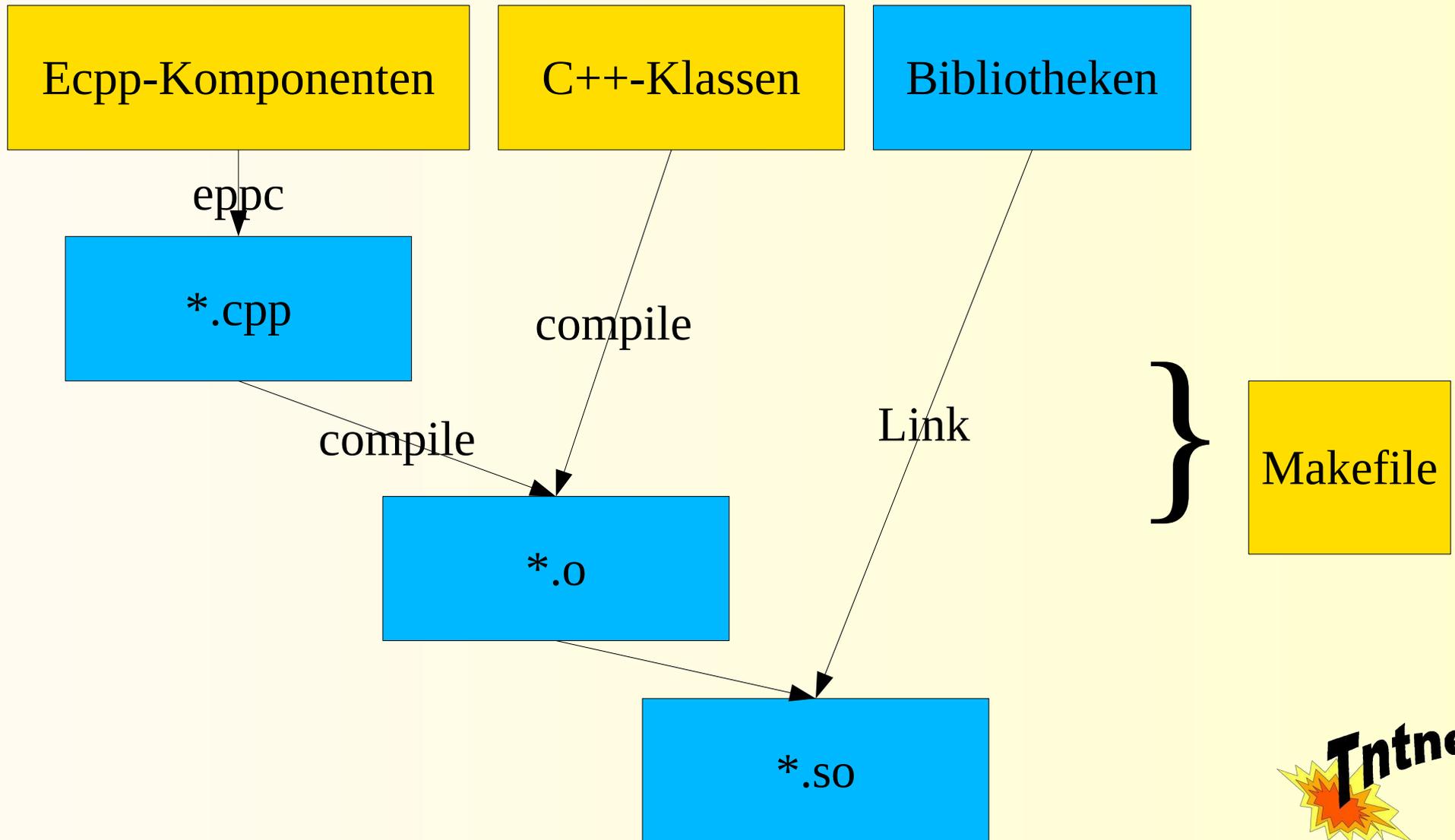
- fehlerhafte Applikationen bringen Server zum Absturz
 - Monitorprozeß überwacht Arbeiterprozeß
 - Zwingt Programmierer zu sauberem Code
- Proprietär, aber Investitionsschutz durch Open-Source
- keine weite Verbreitung (noch?)



Funktionsweise



Entwicklung von Applikationen



Schnellstart

- `tntnet-config --project=app`
- `cd app`
- `make`
- `tntnet tntnet.conf`
- <http://localhost:8000/app>
- logging über `tntnet.properties`



Demos

- Vorführung einiger Demo-Anwendungen
 - hello
 - calc
 - calcmvc
 - calcajax
- Kurzübersicht über die weiteren Demos



die wichtigsten Tags

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



Hilfreiche Bibliotheken

- Tntdb – Bibliothek zum Zugriff auf Datenbanken
 - Datenbankunabhängig
 - Treiber für Sqlite3, Postgresql und Mysql
- Tntpdf – Erstellung von einfachen PDFs
 - sehr schnell
 - eingeschränkte Features, nur einfache Text-PDFs (momentan)



Kontakt

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



Fragen?

