C++ and Apache: Using C++ Server

Christian Gross
CTO Tredix AG
cgross@tredix.com

Bio: Christian Gross

- Author
  - Book APress: *A Programmers Introduction to Windows DNA*
  - Upcoming APress : *From Windows Programming to Linux Programming*
  - Articles: MIND, BasicPro
- Conferences
  - TechEd, Visual C++ DevCon, SD, DevDays

How to approach Apache development

- Need to understand (UNIX derivatives)
  - Shell scripting (mainly for reading)
  - Makefiles (reading and writing)
- Tools that can be used
  - Code Fusion AKA Source Navigator, now Open Source Toolkit for Linux RedHat (5 stars)
  - C-Forge Development Environment
    - C++, C, Java, HTML, etc (5 stars)
  - XWPE
    - C source (3.5 stars)
  - Visual C++
    - Windows development (5 stars)

Agenda

- Outline of the why's of C++ Server
- Refresher of Apache Modules
- Outline of C++ server

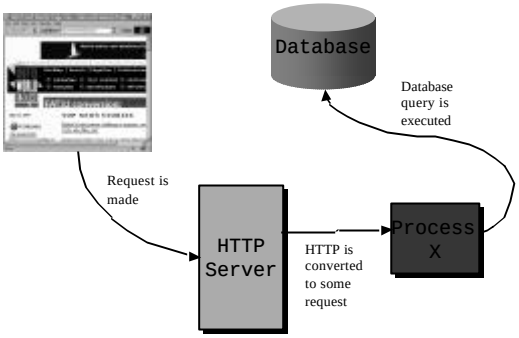
What is the objective?

- Use Apache as a foundation for Internet Applications
- Build applications using another technology that realizes some higher level task
- Notice the differentiation between lower level and higher level applications
  - Each level requires different programming techniques and styles

Application Types

- Apache: Low level plumbing
  - Needs to be stable, fast and efficient
  - Hacks not tolerated
  - "Do it right!"
- Business Application
  - Easy to create, maintainable, easy to catch errors
  - Sometimes need hacks
  - "Get it done!"

Business Application Architecture



Why C++?

- Because the really powerful and fast applications need it
  - HotMail -> ISAPI
  - Tredix AG (Apache and C++ Server)
- C++ is more effective than C as a business application language
  - Easy to write fast, maintainable, error proof applications
- Could use Perl, or PHP…
  - But the power is still with C++

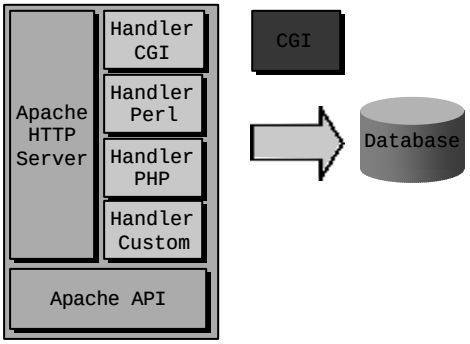
What is C++ Server

- A set of classes that make it simpler to write HTTP Request Handlers
  - Includes a page compiler that converts "ASP" code to C++
- Makes it simpler to write applications that perform specific business tasks
- Allows for easy integration of multi-processing, threading and database
- Includes a user defined persistence framework

Apache modules

- Make it possible to hook in
  - without "breaking" the HTTP server
- Can hook in various locations
  - Perform single tasks or multiple tasks
- Is geared towards performance
  - So long as your module is well written
    - Will not hang server, but process
- More powerful and flexible than any other external API
  - CGI, NSAPI, ISAPI, etc
- Can be static or dynamic DSO

Apache modules



Module defintion

```
typedef struct module_struct {
    int version;                    int minor_version;
    int module_index;               const char *name;
    void *dynamic_load_handle;      struct module_struct *next;
    unsigned long magic;

    void (*init) (server_rec *, pool *);
    void *(*create_dir_config) (pool *p, char *dir);
    void *(*merge_dir_config) (pool *p, void *base_conf, void *new_conf);
    void *(*create_server_config) (pool *p, server_rec *s);
    void *(*merge_server_config) (pool *p, void *base_conf, void *new_conf);

    const command_rec *cmds;
    const handler_rec *handlers;

    int (*translate_handler) (request_rec *);
    int (*ap_check_user_id) (request_rec *);
    int (*auth_checker) (request_rec *);
    int (*access_checker) (request_rec *);
    int (*type_checker) (request_rec *);
    int (*fixer_upper) (request_rec *);
    int (*logger) (request_rec *);
    int (*header_parser) (request_rec *);
    void (*child_init) (server_rec *, pool *);
    void (*child_exit) (server_rec *, pool *);
    int (*post_read_request) (request_rec *);
} module;
```

Module development

- Fill in the structure with appropriate values
  - NULL defines not implemented
- To send straight content based on some server configuration minimum structure requires
  - command_rec structure
  - handler_rec structure

Phases outlined

- There are three module phases
  - Configuration
  - Instantiation / Exit
  - Request handling

Configuration Phases

- Events
  - Create per server configuration
  - Create per directory configuration
  - Per server merger
  - Per directory merger
  - Command handlers
- Is called for the main server and the various virtual servers

Instantiation / Exit

- Events
  - Module instantiation
  - Child instantiation
  - Child exit
- There are global events and per child events
- Child events are called whenever the child processes are started and stopped

External Apache modules

Design considerations

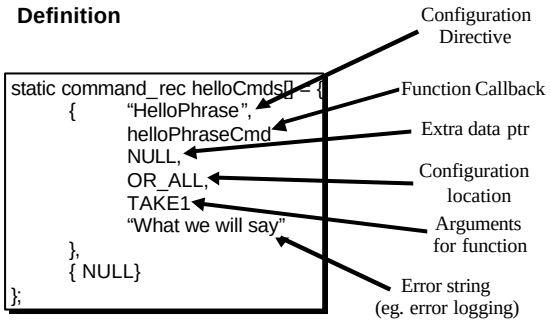
- Hook into the HTTP request
  - Post read request
  - URL translation
  - Header parsing
  - Access control (access control, authentication, authorization)
  - Type checking
  - Apache handler
  - Fixups
  - Logging of the request
- Can hook into one or all steps
- Reply with
  - Success, abort and decline

Module Structure

```
module config_log_module = {
   STANDARD_MODULE_STUFF,
   NULL,          /* initializer */
   NULL,          /* create per-dir config */
   NULL,          /* merge per-dir config*/
   NULL,          /* server config*/
   NULL,          /* merge server config*/
   helloCmds,     /* command table */
   helloHandlers,          /* content handlers */
   NULL,          /* URL to filename translation */
   NULL,          /* check user_id is valid */
   NULL,          /* check auth */
   NULL,          /* check access */
   NULL,          /* type_checker */
   NULL,          /* fixups */
   NULL,          /* logger */
   NULL,          /* process initialization*/
   NULL,          /* process exit/cleanup */
   NULL           /* post read_request handling*/
};
```

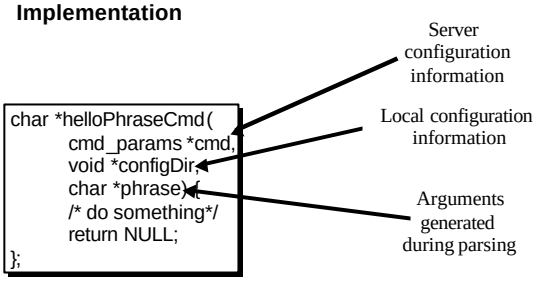
Command structures

Definition



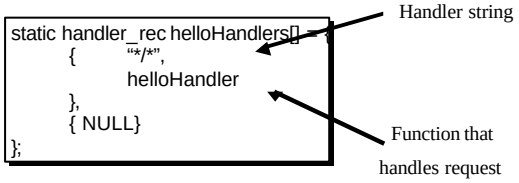
Command structures

Implementation



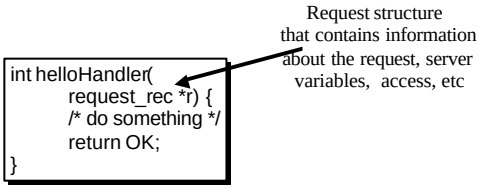
Handler structures

Definition



Handler structures

Implementation



APR (Apache Runtime)

- **A neutral programming layer that is embedded on top of the operating system**
- **Makes it simpler to port native code modules to other platforms**
- **Typically most function calls are defined with *ap_***
- **Before careful how you use the API with Dynamic (DLL or DSO) Libraries**

Apache memory pools

- **Memory is managed in pools in Apache**
  - **Makes it simpler to clear memory**
  - **Ensures that nothing is leaked**
  - **And it can be faster**
- **When a request is sent to the module it has a handle to the pool via the *request_rec structure***
- **Example**
  foo = (foo *)ap_malloc(r->pool, sizeof(foo))

C++ Server Design Concepts

- Do not reinvent modPerl or PHP
- Convert as much dynamic content to static
  - Compiled is faster than scripted
- Ensure that it is possible to associate "session" data with a server side object
- Make it easily possible to do multi-threading, multi-processing
- Do not hinder C++, complement it
- Handle only HTTP request phase
  - Leave Apache Server configuration as is

Problem 1: Static functions

- In Apache 1.3.x series need static functions that are associated with a structure
- In Apache 2.x series still need a static function that can be dynamically associated
- Static functions and C++ do not easily mix
- Solution is to create a static class member and hand off the request to an instantiated class

Static Function Handoff
(Concept)

```
static int staticHandler( request_rec *r) {
        baseClass *cls;
        int retval;
        cls = new baseClass;
        retval = cls->handleRequest();
        delete cls;
        return retval;
}
```

Static Function Handoff
(Concept) cont…

- Each request allocates a new class
  - While it would seem to optimize by pooling objects, it is not better
    - Adds overhead and complexity
  - Cost is not in instantiating the objects, but the memory that the object manages
    - Two solutions: Trust heap manager, or use Apache Pooled memory manager
      - Using APR makes it simpler for object clean-up
      - Ownerless pointers solved by APR

C++ Server SFH Implementation

- For module level calls header processing, fixup, etc
  - Static global class
    - Need thread synchronization
- For an individual Apache Handler
  - Newly instantiated class for each request
    - Does not need thread synchronization
  - Bulk of application server work done in the handler

C++ Server For the Impatient

- The simplest C++ Server class is "sessionless"
- Steps to create a simple C++ server
  - Copy file mod_cpp.cpp to your module directory and rename the file
  - In top of your "mod_cpp.cpp" redefine the macros:
    - MOD_NAME: Apache LoadModule name
    - MOD_INSTANTIATE: Apache Configuration Item used in HTTP.CONF to activate your C++ Server extension
    - MOD_BASE_CLASS: Name of Module

C++ Server For the Impatient

cont…

- Implement the class defined by the macro MOD_BASE_CLASS
  - Ensure class is derived from class ApacheModule
- Implement an Apache handler by defining a class derived from template ApacheSessionlessHandler
- Implement the virtual function handleRequest
- In handleRequest use APR to send output
  - rec is Apache request_rec class

C++ Server For the Impatient

cont…

- For ApacheSessionlessHandler derived class add macro IDENTIFIER with an Apache Handler String
- In ApacheModule derived class add

```
BEGIN_HANDLER_MAP()
  HANDLER_ENTRY( [My Handler])
END_HANDLER_MAP()
```

- Add IDENTIFIER String to HTTP.CONF
- Compile
- Run Apache

C++ Server example

- Actually build the sessionless handler to output hello world

What is happening…

- When Apache Module is loaded using the macro MOD_INSTANTIATE the various handlers are wired into Apache like a dynamic module
  - See function InstantiateModCPP
- Once wired the IDENTIFIER string maps the Apache Handler request to a static function of ApacheSessionlessHandler
- Request is then handled

Problem 2: State

- When a browser visits your module multiple times you may want to build some state
  - eg shopping cart, stock tracking, etc
- ApacheHandler template class is like ApacheSessionlessHandler, but has an extra data member ApacheSession
- ApacheSession is a class that you derive from to load and save state regarding a session managed with a cookie

State Handler Demo

- Extend the sessionless handler to include state
  - The state counts the number of times that the user visits the web page

"Printf"s for output

- Do we really want yet another series of printf's to display output?
- One solution is to use a library that builds the HTML on the fly
  - Not good because it makes it very hard for "normal" website designers to modify the page
- Better solution is to convert dynamic pages into static pages
  - ASP (Active Server Pages) is a defacto standard

C++ Server ASP compiler

- A page compiler that translates ASP pages into a series of C++ modules that are compiled into a module
  - More robust
  - Makes it possible to distribute your application, without need to distribute many files
  - Faster, unless of course your write some REALLY BAD C++

C++ Server ASP Compiler

- Concept is to generate look in ASP, but derive functionality from a C++ business class
- PCOMP (ASP Compiler) generates a class that derives from a C++ business class
  - Web designers only need "interfaces" to your various business classes

ASP Compiler Demo

- Extending Hello world to use ASP pages

C++ Server Details

- Is Open Source using the Apache license
- Is available from www.devspace.com
- I will accept bug reports mail me at cgross@devspace.com
- Is used in the Tredix WebSite

Reference

- Apache Web site www.apache.org
  - Modules development mailing list
  - Modules registry list
  - Web site module.apache.org
  - ASP support is available
  - PHP support is also available
- O'Reilly Website / Books
  - Apache the definitive guide (administrative)
  - Writing Apache modules with Perl and C

Thanks!

Questions?