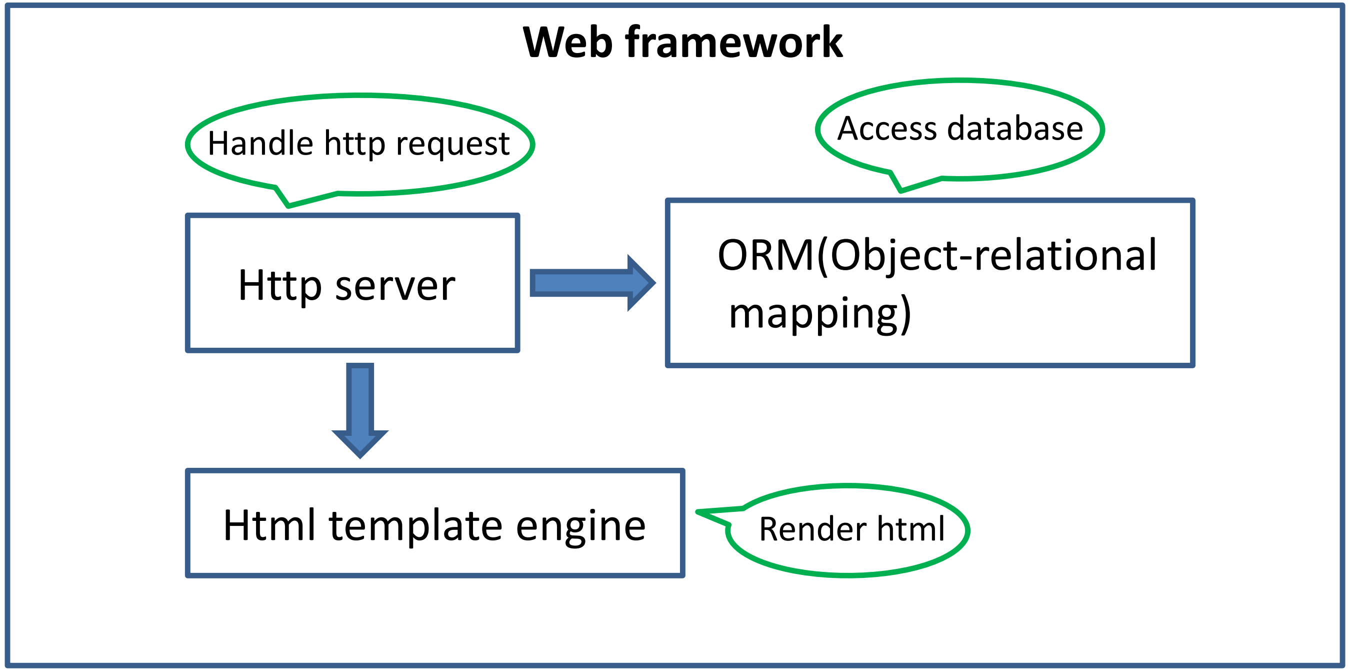
Feather: A Modern C++ Web Development Framework

Yu Qi
qicosmos@163.com

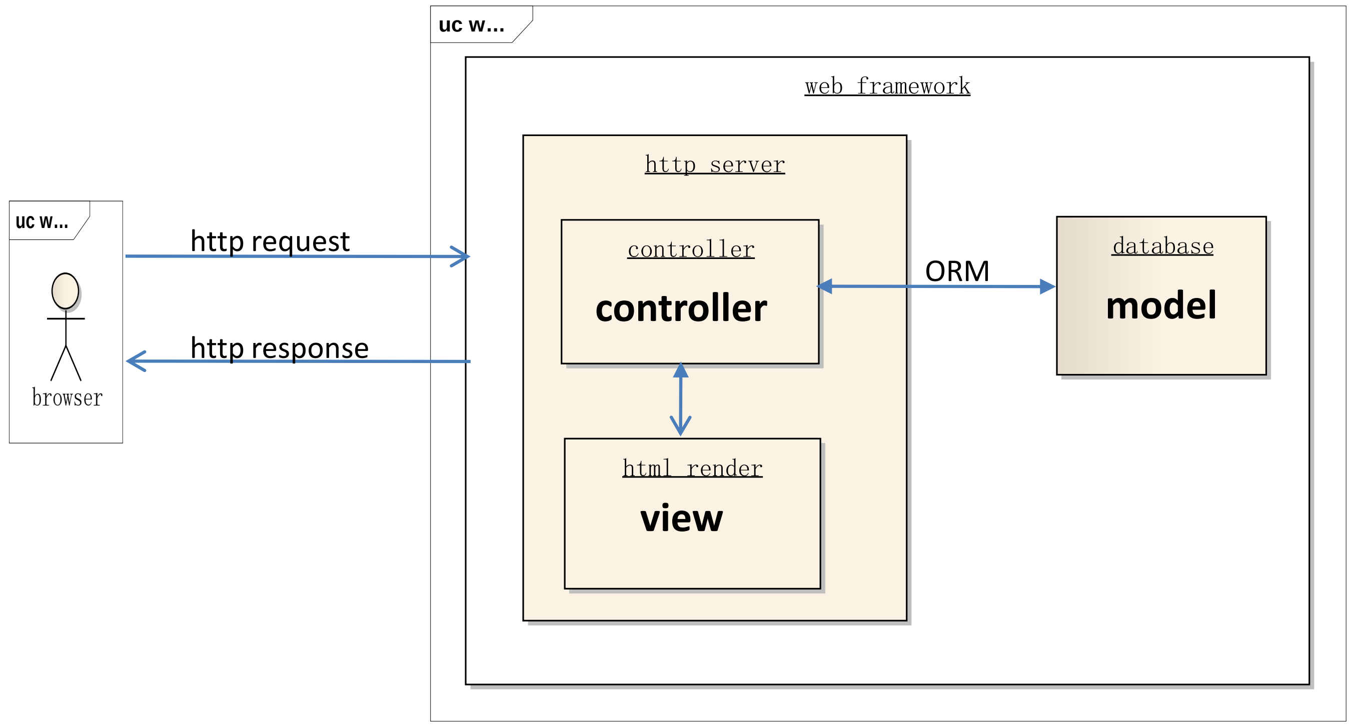
Outline

- Basic Concepts of Web Framework
- What is Feather?
- Components
- How to Rapidly Develop Web Applications?

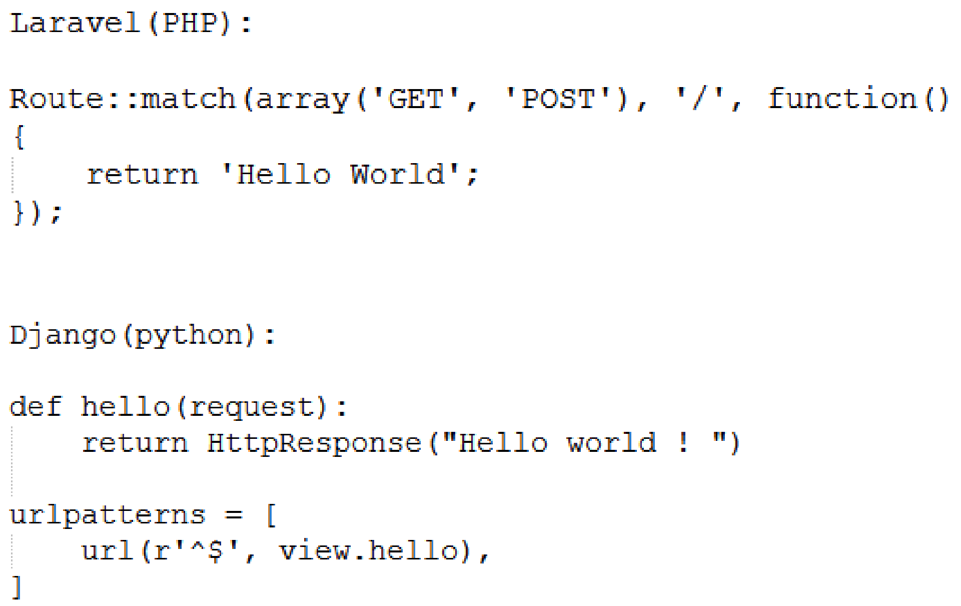
Basic concepts



Basic concepts

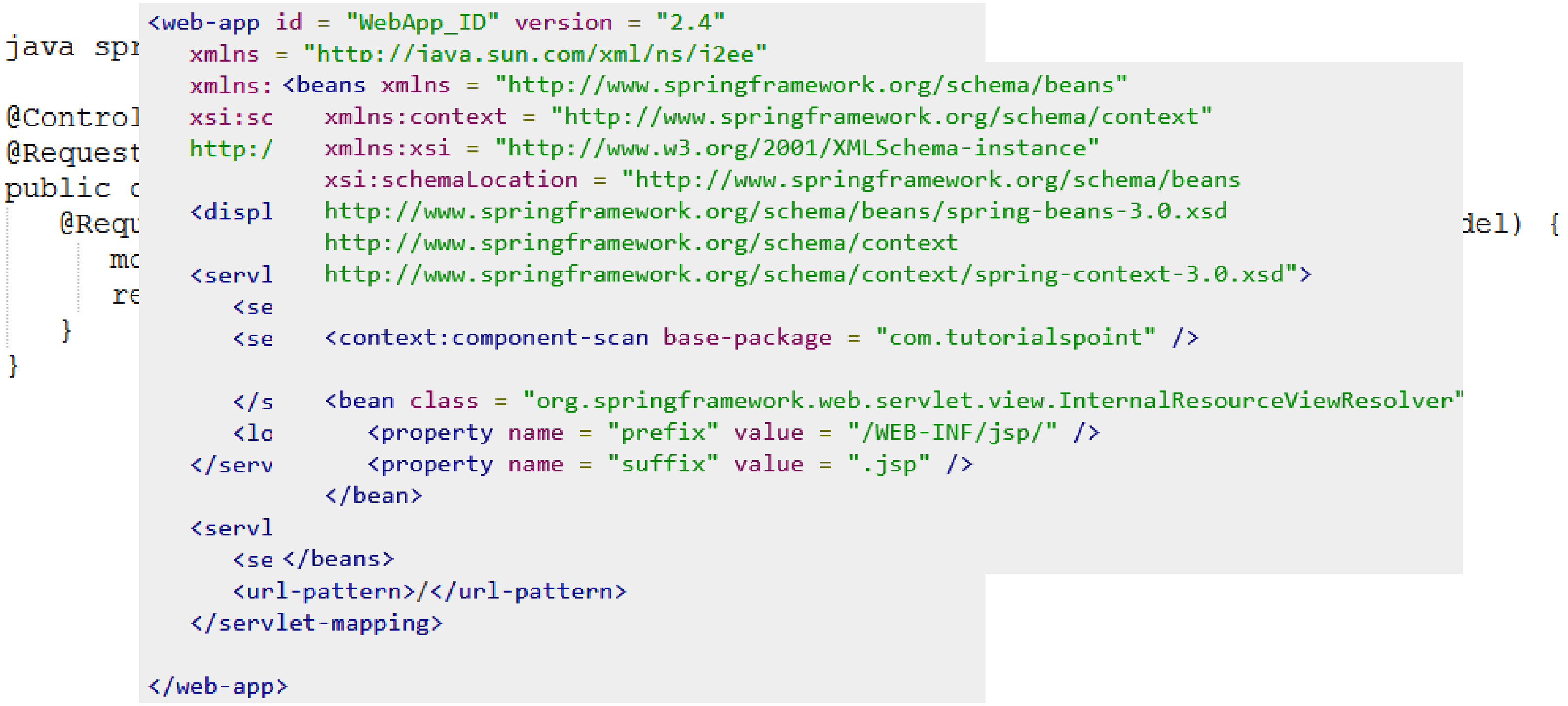


Some web frameworks



```
Laravel(PHP):

Route::match(array('GET', 'POST'), '/', function()
{
    return 'Hello World';
});


Django(python):

def hello(request):
    return HttpResponse("Hello world ! ")

urlpatterns = [
    url(r'^$', view.hello),
]
```

Some web frameworks



```
java spr

@Control
@Request
public c
    @Requ                                                          del) {
        mc
        re
    }
}
```

```
<web-app id = "WebApp_ID" version = "2.4"
   xmlns = "http://java.sun.com/xml/ns/j2ee"
   xmlns:
   xsi:sc
   http:/

   <displ

   <servl
      <se
      <se

      </s
      <lo
   </serv

   <servl
      <se
      <url-pattern>/</url-pattern>
   </servlet-mapping>

</web-app>
```

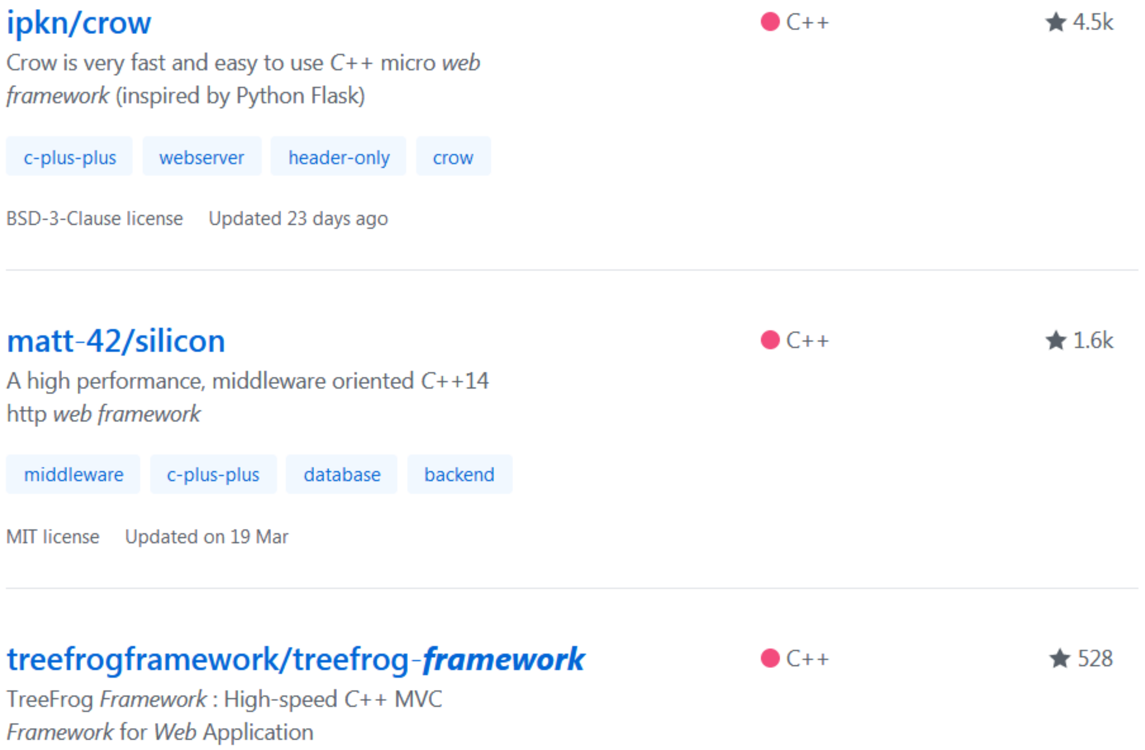
```
<beans xmlns = "http://www.springframework.org/schema/beans"
   xmlns:context = "http://www.springframework.org/schema/context"
   xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation = "http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
   http://www.springframework.org/schema/context
   http://www.springframework.org/schema/context/spring-context-3.0.xsd">

   <context:component-scan base-package = "com.tutorialspoint" />

   <bean class = "org.springframework.web.servlet.view.InternalResourceViewResolver"
      <property name = "prefix" value = "/WEB-INF/jsp/" />
      <property name = "suffix" value = ".jsp" />
   </bean>
</beans>
```

Some popular c++ web frameworks

cppcms.com/wikipp/en/page/main

CppCMS — C++ Web Framework

4 popular c++ web frameworks



ipkn/crow ● C++ ★ 4.5k

Crow is very fast and easy to use C++ micro *web framework* (inspired by Python Flask)

c-plus-plus webserver header-only crow

BSD-3-Clause license Updated 23 days ago

matt-42/silicon ● C++ ★ 1.6k

A high performance, middleware oriented C++14 http *web framework*

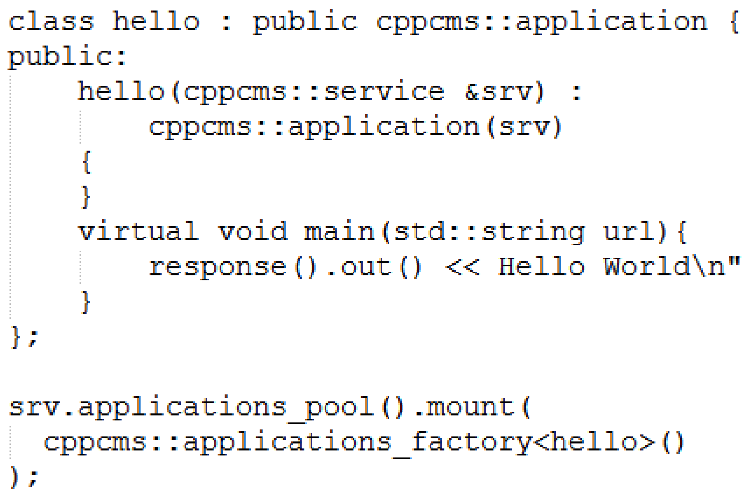
middleware c-plus-plus database backend

MIT license Updated on 19 Mar

treefrogframework/treefrog-***framework*** ● C++ ★ 528

TreeFrog *Framework* : High-speed C++ MVC *Framework* for *Web* Application

cppcms



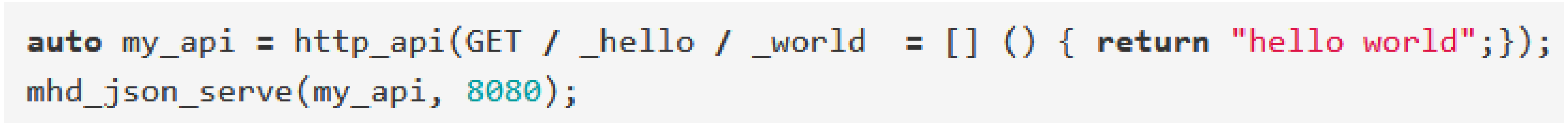
```
class hello : public cppcms::application {
public:
    hello(cppcms::service &srv) :
        cppcms::application(srv)
    {
    }
    virtual void main(std::string url){
        response().out() << Hello World\n"
    }
};

srv.applications_pool().mount(
  cppcms::applications_factory<hello>()
);
```

You have to know many details of the framework

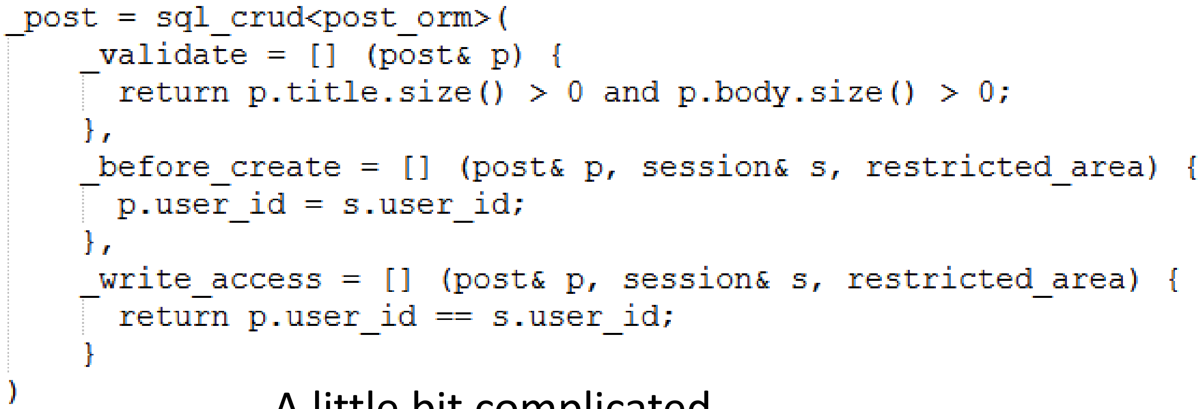
Need a special compiler to build
html templates

- Tree-frog
  - based on qt, too heavy
- Crow
  - Just a http server, not a real web framework

silicon



```cpp
auto my_api = http_api(GET / _hello / _world  = [] () { return "hello world";});
mhd_json_serve(my_api, 8080);
```



```cpp
_post = sql_crud<post_orm>(
    _validate = [] (post& p) {
      return p.title.size() > 0 and p.body.size() > 0;
    },
    _before_create = [] (post& p, session& s, restricted_area) {
      p.user_id = s.user_id;
    },
    _write_access = [] (post& p, session& s, restricted_area) {
      return p.user_id == s.user_id;
    }
)
```

A little bit complicated

Lacks of html template engine

We need a better c++ web framework

- Perfect infrastructures
  - http server, ORM, html template engine, AOP,…….
- Easy to use
- Focus on business only, low learning cost
- High performance, cross platform

We need a better c++ web framework

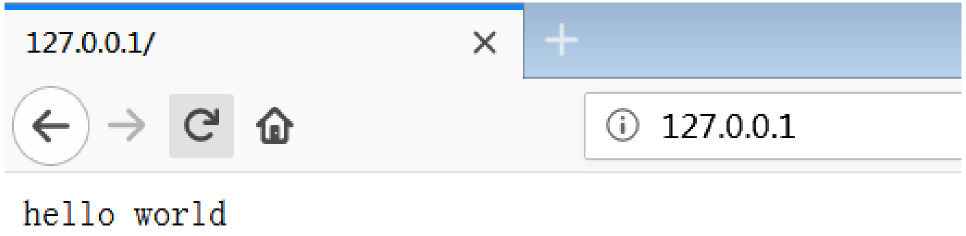
- Feather: a rapidly application framework of web development

  you can use feather to develop a web application rapidly

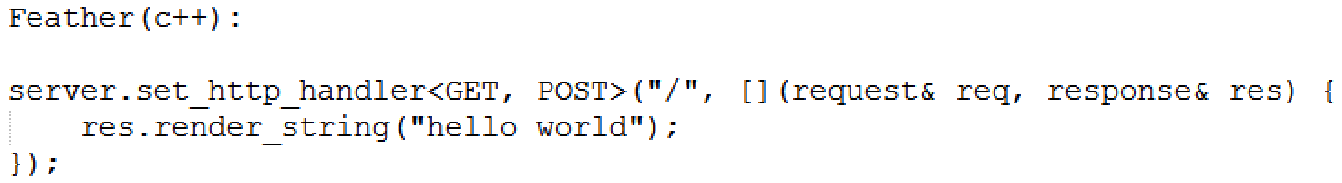
What is feather?

- A web application with 5 lines code

```
http_server server(4);
server.listen("0.0.0.0", "http");
server.set_http_handler<GET, POST>("/", [](request& req, response& res) {
    res.set_status_and_content(status_type::ok, "hello world");
});
server.run();
```



```
hello world
```



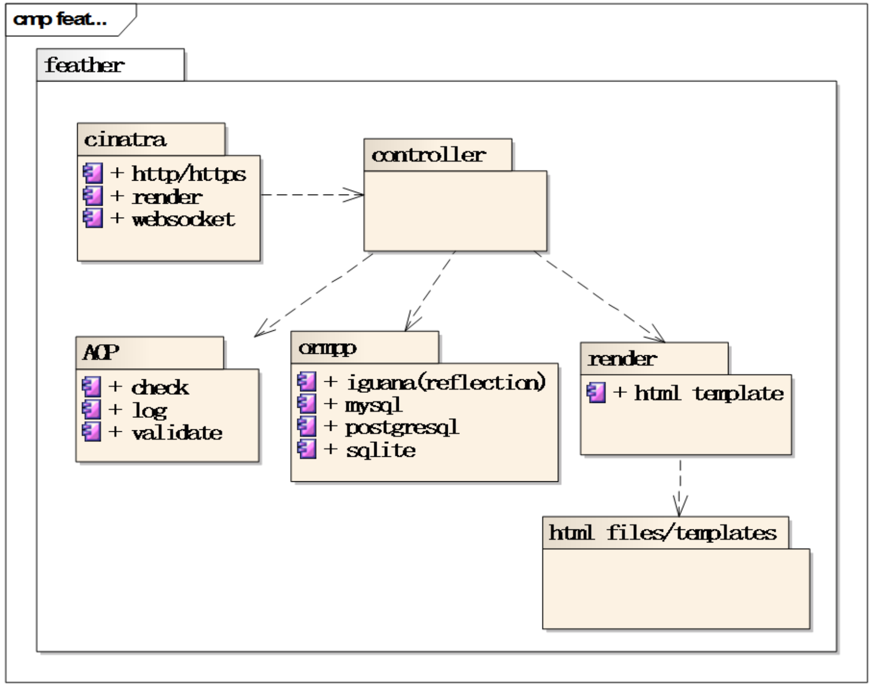
```
Feather(c++):

server.set_http_handler<GET, POST>("/", [](request& req, response& res) {
    res.render_string("hello world");
});
```

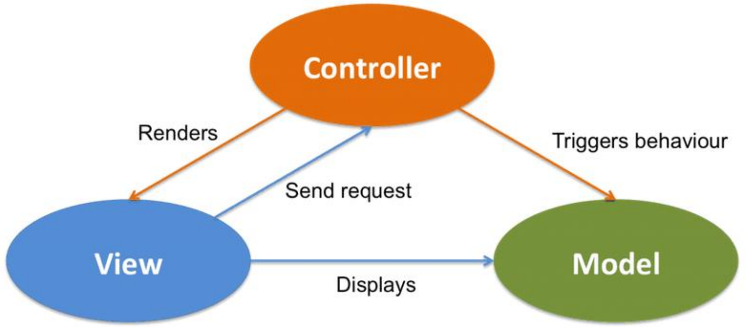
Very simple, no need to know many details about the framework

Just focus on business, as simple as possible

Become the master of the framework not the slave!

What is feather?





What is feather?

This website was developed in feather
http://purecpp.org/

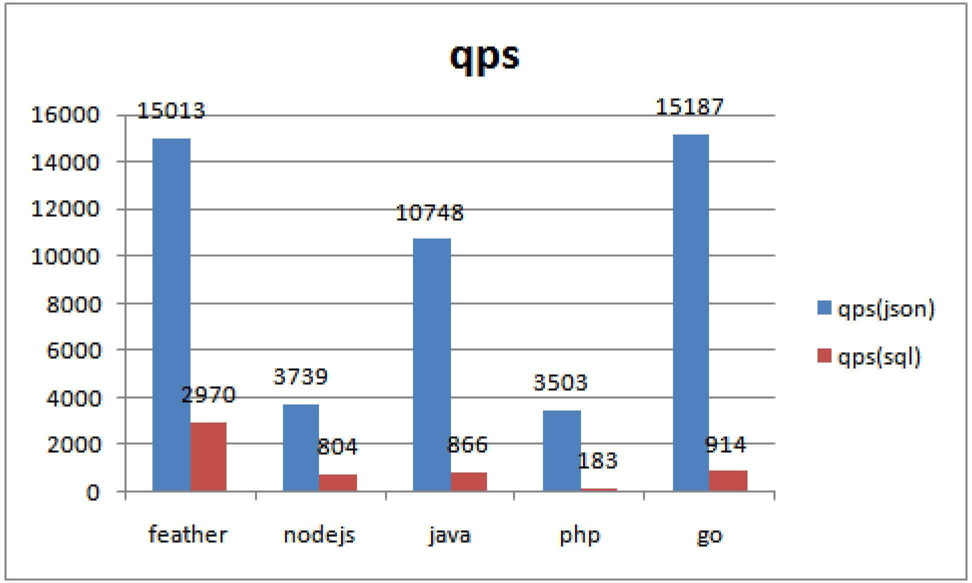
https://github.com/qicosmos/feather

Development efficiency

- How many business functions of http://purecpp.org ?
  1. add, edit, remove, query posts;
  2. add, edit, remove, query users;
  3. member login/quit, member registration
  4. upload/download files
  5. search posts/classify posts
  6. …….

More than **15** business functions
All the core business code is about **500** lines
30 lines code per business function

Performance

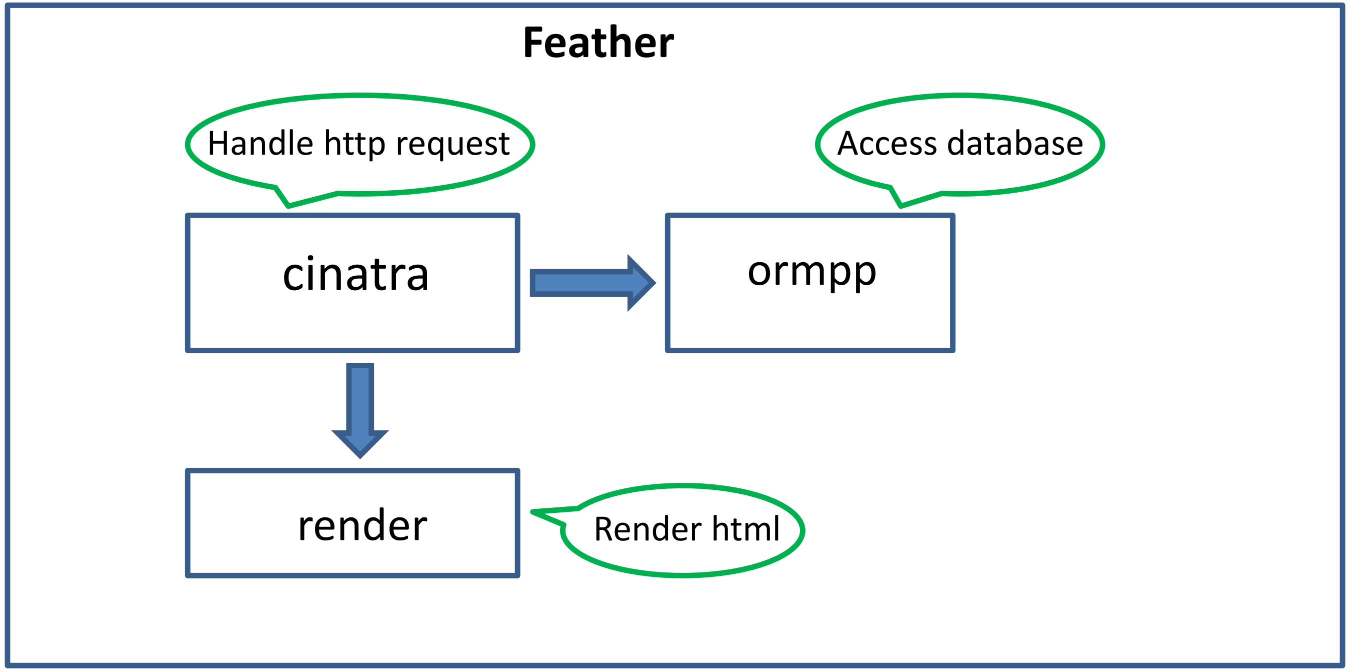


Higher is better

Why we need a c++ web framework feather?

- High development efficiency
- High performance

Components


Feather
Handle http request
Access database
cinatra
ormpp
render
Render html

Cinatra

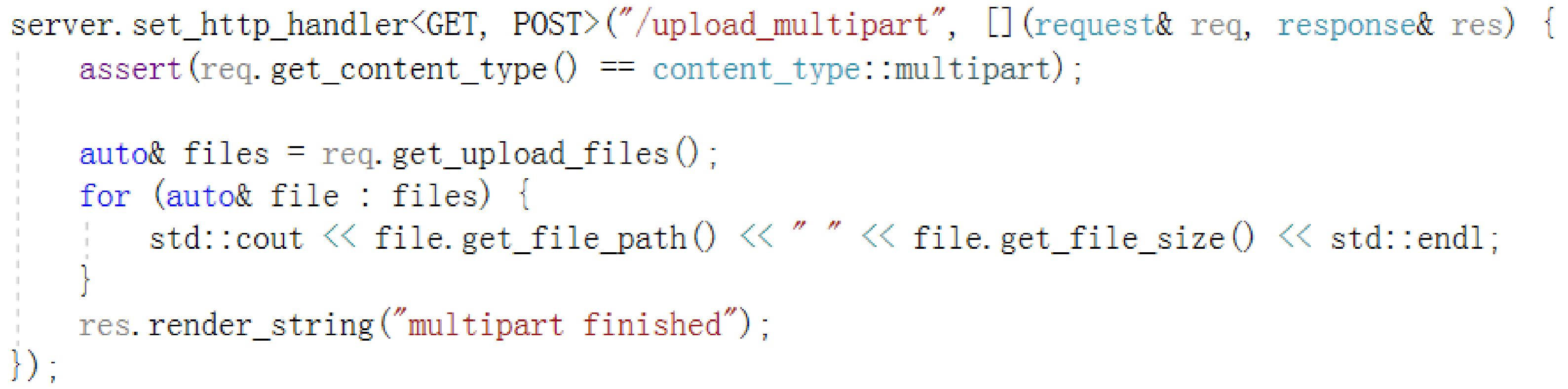
- Cinatra is a http server based on asio developed in c++17
  - Http1.1/https, websocket
  - Functional high level easy to use interface
  - Header-only
  - Support AOP(Aspect Oriented Programming)
  - Support upload/download files, session/cookie

- Differences between cinatra and beast
  - cinatra is an application library, provide high level interface, lead to very few code to finish a web application
  - cinatra do much work for users
  - beast is a fundamental library, provide low level interface
  - beast users have to write lots of code to finish a business function

The goal is different

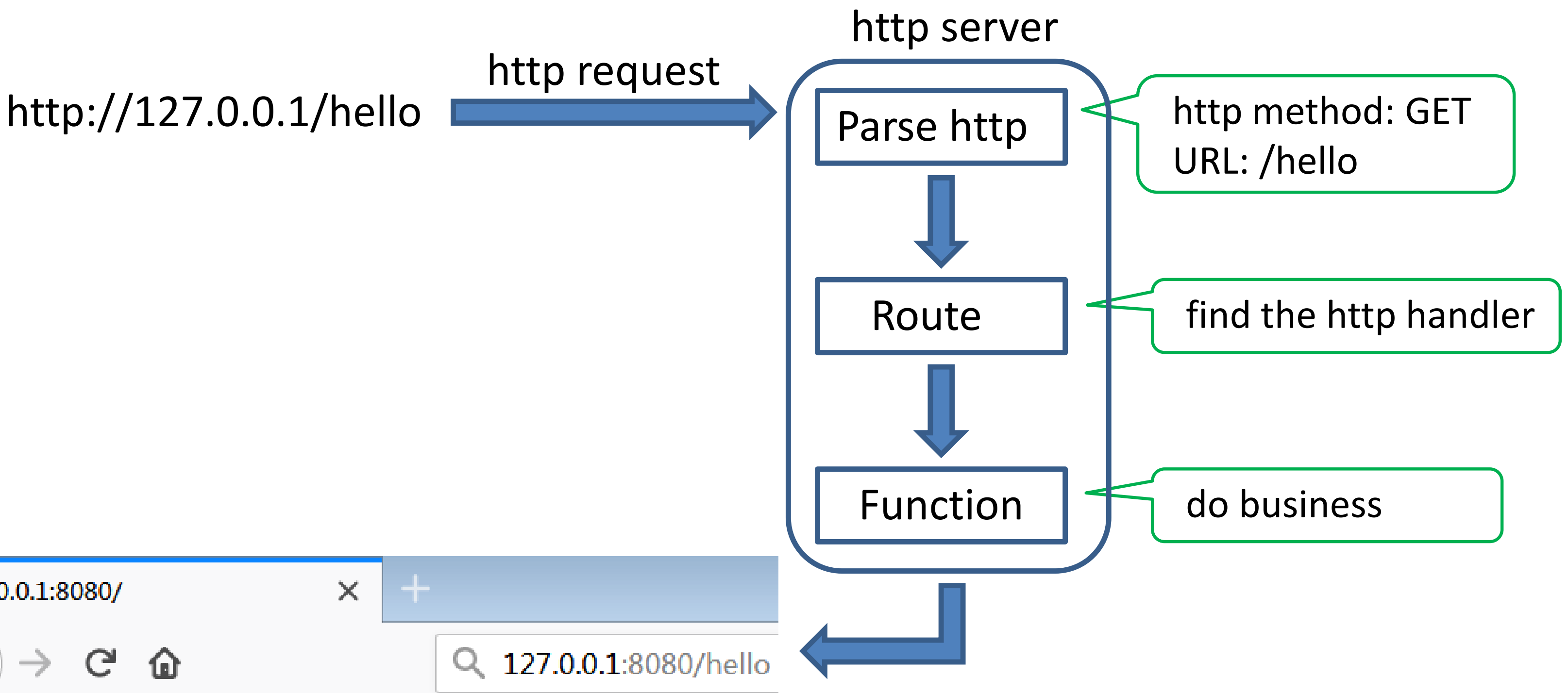
http upload/download

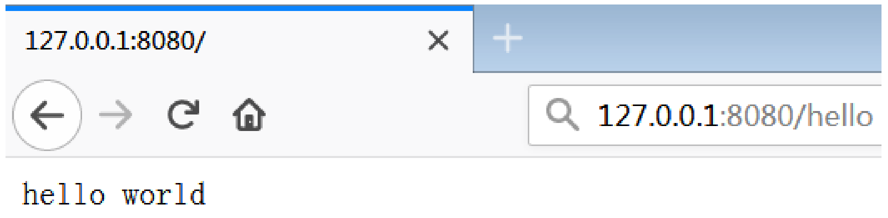
– Beast lead to more than 100 lines code
  - https://codereview.stackexchange.com/questions/148596/http-downloader-using-beast
– Cinatra just need less than 6 lines code



```
server.set_http_handler<GET, POST>("/upload_multipart", [](request& req, response& res) {
    assert(req.get_content_type() == content_type::multipart);

    auto& files = req.get_upload_files();
    for (auto& file : files) {
        std::cout << file.get_file_path() << " " << file.get_file_size() << std::endl;
    }
    res.render_string("multipart finished");
});
```

http download in cinatra: http://127.0.0.1/assets/show.jpg

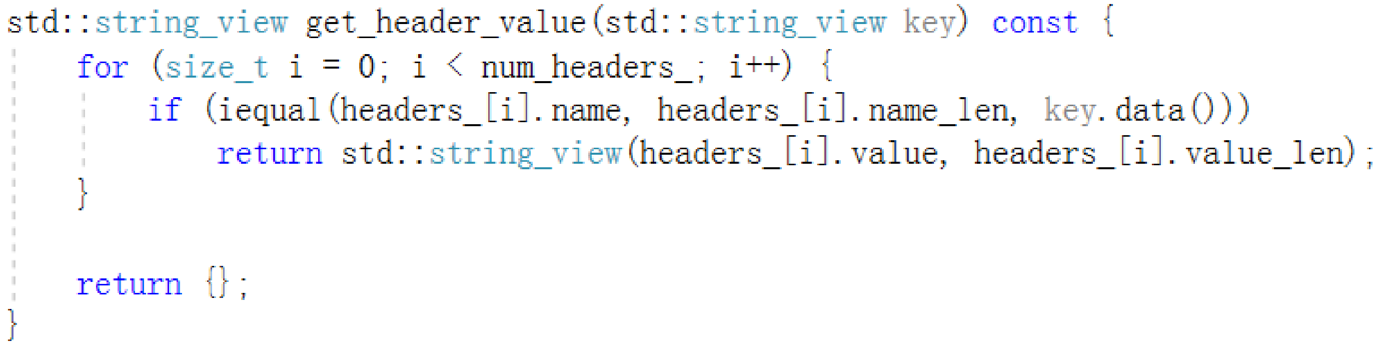

http://127.0.0.1/hello
http request
http server
Parse http
Route
Function
http method: GET
URL: /hello
find the http handler
do business


127.0.0.1:8080/
127.0.0.1:8080/hello

```
hello world
```

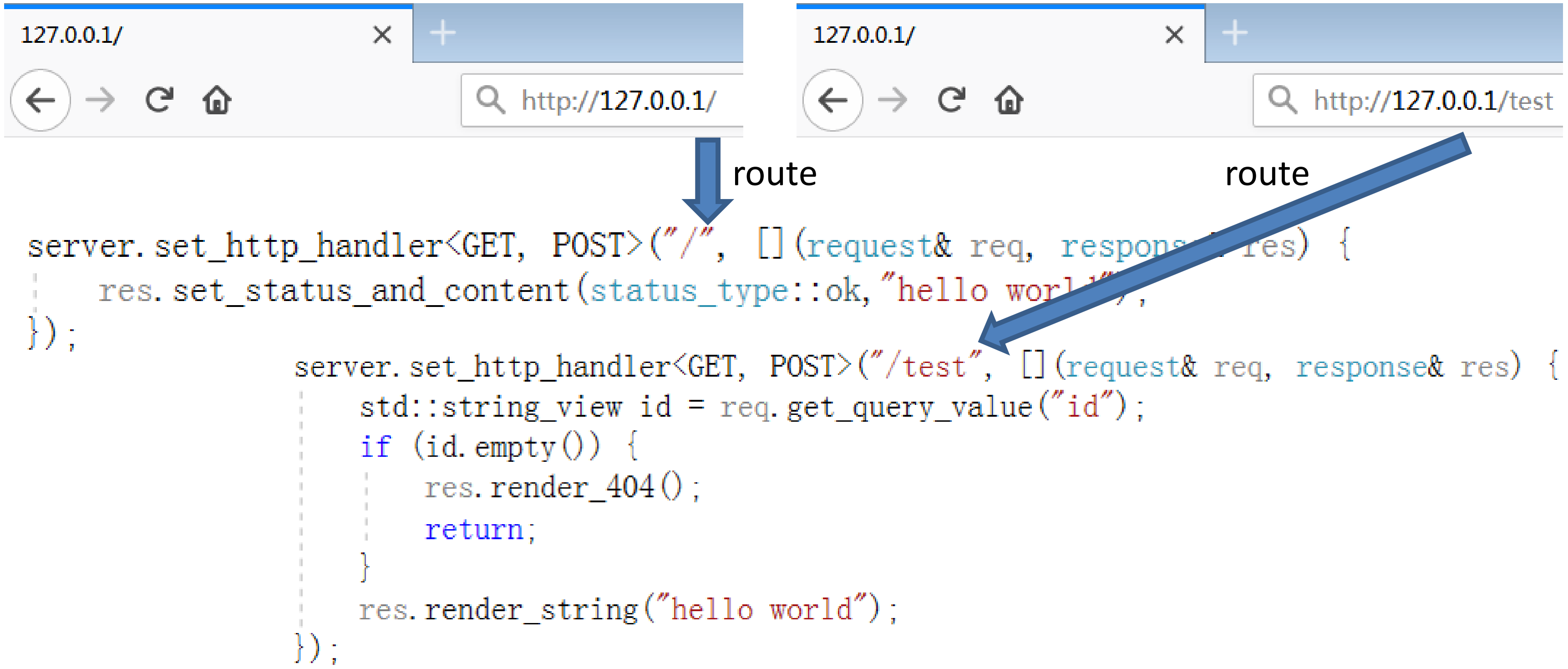
http parser

- Parse http request: method, url,version,headers,……
- std::string_view is very suitable for parsing http protocol



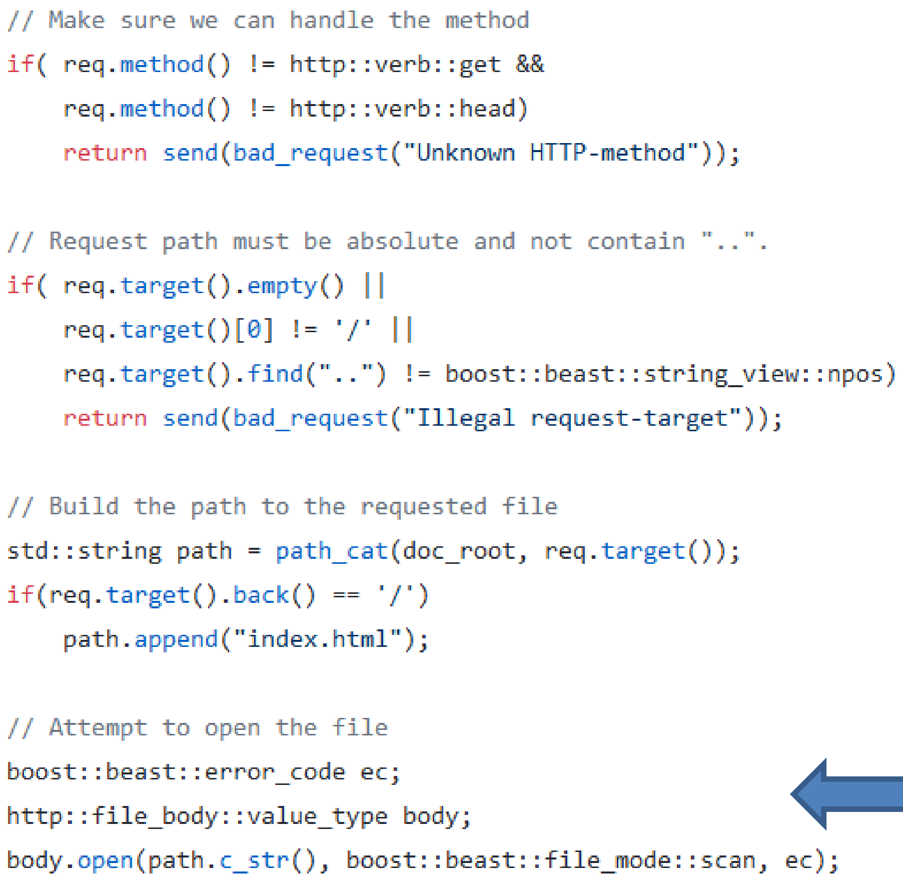
```
std::string_view get_header_value(std::string_view key) const {
    for (size_t i = 0; i < num_headers_; i++) {
        if (iequal(headers_[i].name, headers_[i].name_len, key.data()))
            return std::string_view(headers_[i].value, headers_[i].value_len);
    }

    return {};
}
```

http router



```
server.set_http_handler<GET, POST>("/", [](request& req, respons   res) {
    res.set_status_and_content(status_type::ok, "hello worl   ;
});
```

```
server.set_http_handler<GET, POST>("/test", [](request& req, response& res) {
    std::string_view id = req.get_query_value("id");
    if (id.empty()) {
        res.render_404();
        return;
    }
    res.render_string("hello world");
});
```

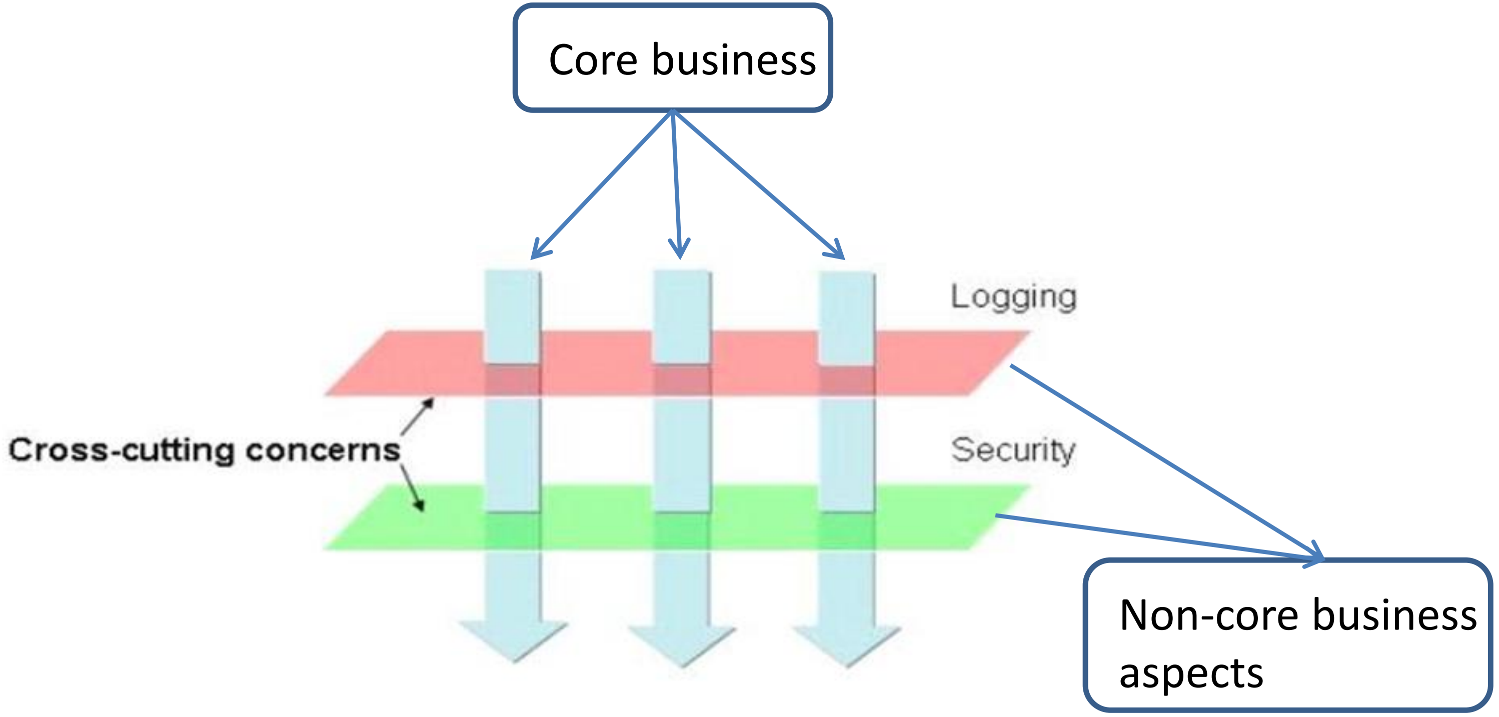


```
// Make sure we can handle the method
if( req.method() != http::verb::get &&
    req.method() != http::verb::head)
    return send(bad_request("Unknown HTTP-method"));

// Request path must be absolute and not contain "..".
if( req.target().empty() ||
    req.target()[0] != '/' ||
    req.target().find("..") != boost::beast::string_view::npos)
    return send(bad_request("Illegal request-target"));

// Build the path to the requested file
std::string path = path_cat(doc_root, req.target());
if(req.target().back() == '/')
    path.append("index.html");

// Attempt to open the file
boost::beast::error_code ec;
http::file_body::value_type body;
body.open(path.c_str(), boost::beast::file_mode::scan, ec);
```

should avoid long code

should focus on what we want

AOP(Aspect Oriented Programming)



AOP

```
server.set_http_handler<GET, POST>("/aspect", [](request& req, response& res) {
    std::cout << "in business function" << std::endl;
    res.render_string("hello world");
}, check{}, log_t{});
```

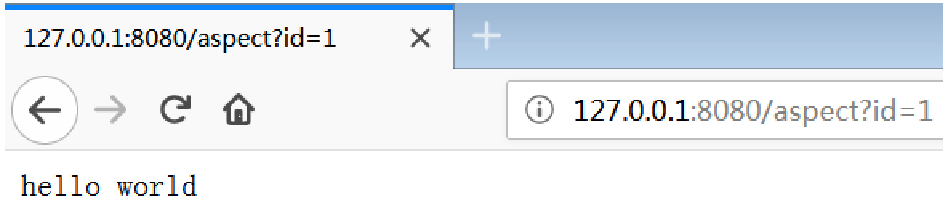
```
struct log_t{
    bool before(request& req, response& res) {
        std::cout << "before log" << std::endl;
        return true;
    }

    bool after(request& req, response& res) {
        std::cout << "after log" << std::endl;
        return true;
    }
};
```

```
struct check {
    bool before(request& req, response& res) {
        if (req.get_query_value("id").empty()) {
            res.render_404();
            return false;
        }
        std::cout << "check passed" << std::endl;
        return true;
    }

    bool after(request& req, response& res) {
        std::cout << "after check" << std::endl;
        return true;
    }
};
```

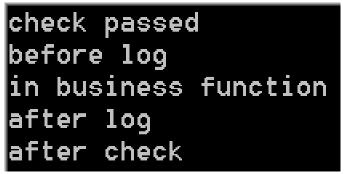
AOP

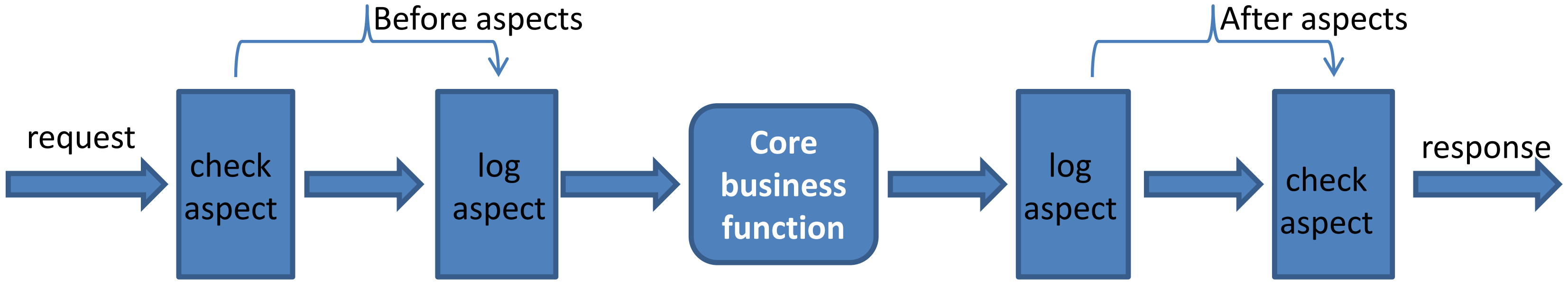
Separate non-core business from core business
Easy to add new aspect, make the framework more flexible

```
server.set_http_handler<GET, POST>("/aspect", [](request& req, response& res) {
    std::cout << "in business function" << std::endl;
    res.render_string("hello world");
}, check{}, log_t{});
```

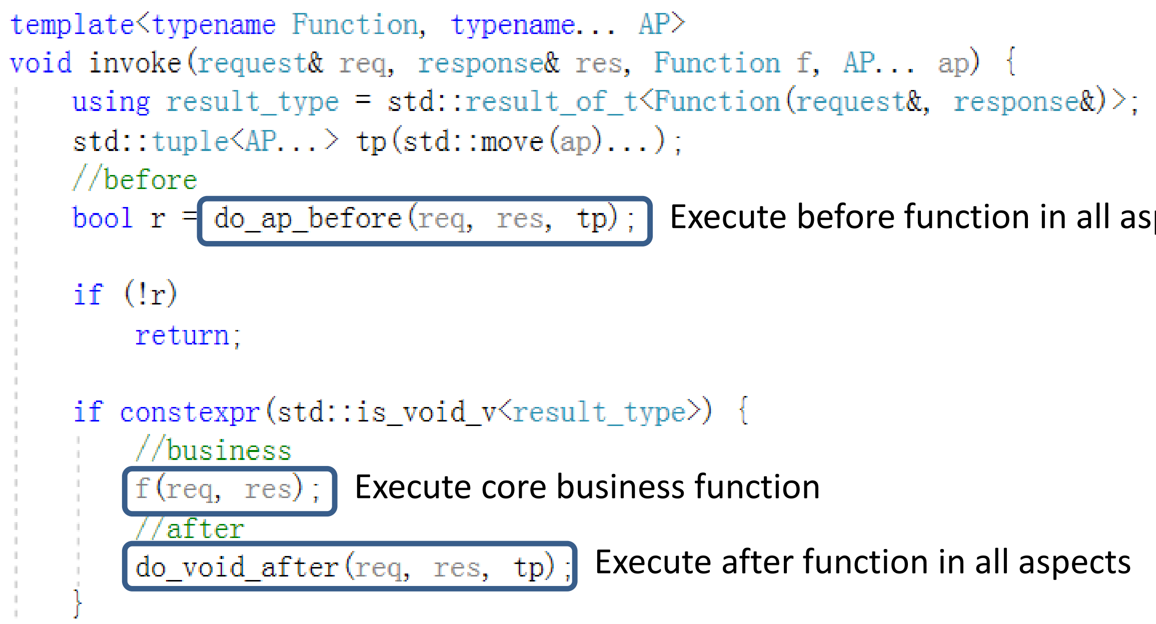

127.0.0.1:8080/aspect?id=1

```
hello world
```


```
check passed
before log
in business function
after log
after check
```



AOP



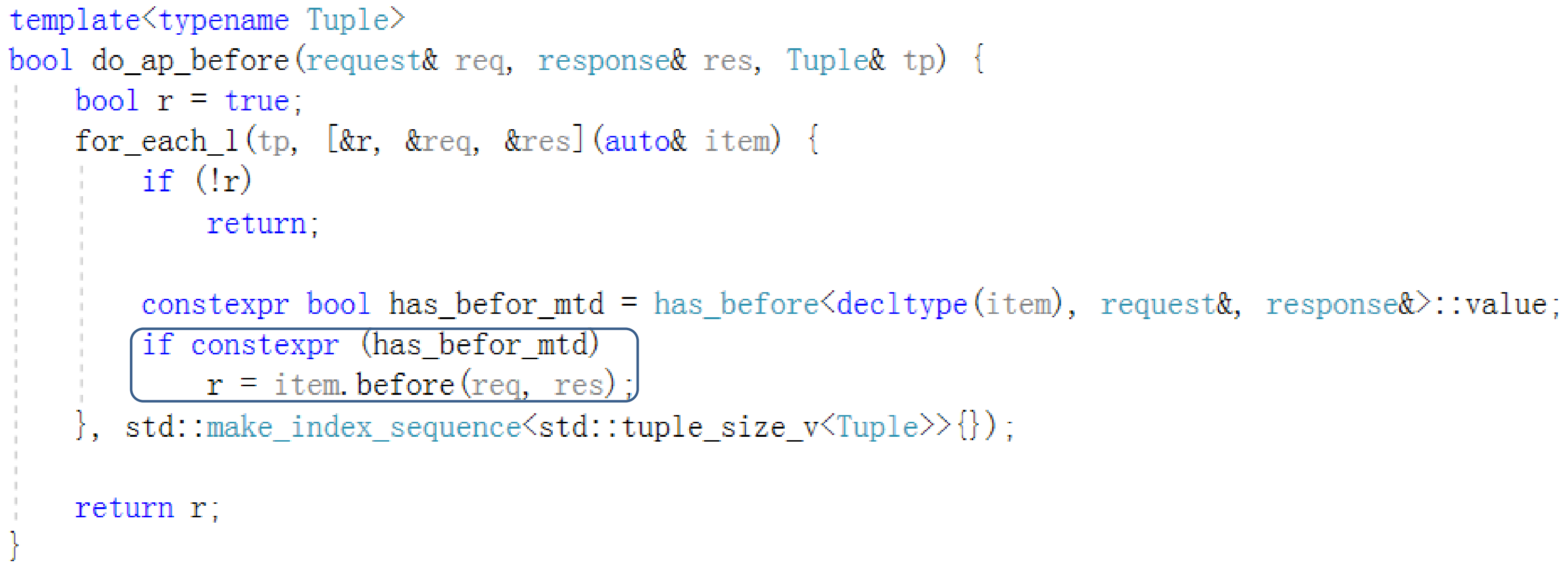
```cpp
template<typename Function, typename... AP>
void invoke(request& req, response& res, Function f, AP... ap) {
    using result_type = std::result_of_t<Function(request&, response&)>;
    std::tuple<AP...> tp(std::move(ap)...);
    //before
    bool r = do_ap_before(req, res, tp);

    if (!r)
        return;

    if constexpr(std::is_void_v<result_type>) {
        //business
        f(req, res);
        //after
        do_void_after(req, res, tp);
    }
```

Execute before function in all aspects

Execute core business function

Execute after function in all aspects

AOP



```cpp
template<typename Tuple>
bool do_ap_before(request& req, response& res, Tuple& tp) {
    bool r = true;
    for_each_l(tp, [&r, &req, &res](auto& item) {
        if (!r)
            return;

        constexpr bool has_befor_mtd = has_before<decltype(item), request&, response&>::value;
        if constexpr (has_befor_mtd)
            r = item.before(req, res);
    }, std::make_index_sequence<std::tuple_size_v<Tuple>>{});

    return r;
}
```

AOP

```
template<typename T, typename... Args>
struct has_before_t
{
private:
    template<typename U> static auto Check(int) ->
        decltype(std::declval<U>().before(std::declval<Args>()...), std::true_type());
    template<typename U> static std::false_type Check(...);
public:
    static constexpr bool value = std::is_same<decltype(Check<T>(0)), std::true_type>::value;
};
```

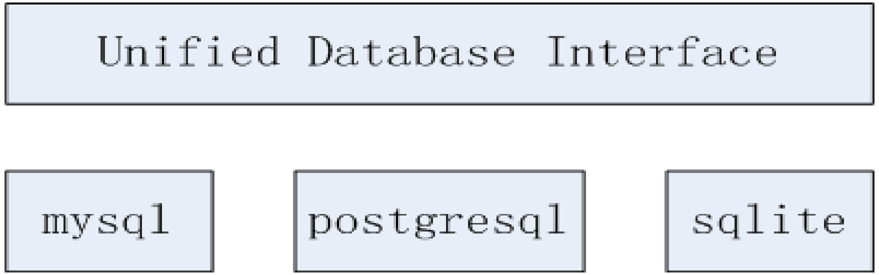
cinatra

- std::string_view helps to parse http protocol efficiently
- The http router and AOP make the user focus on core business function
- The framework does much work and then provides simple interface for the user

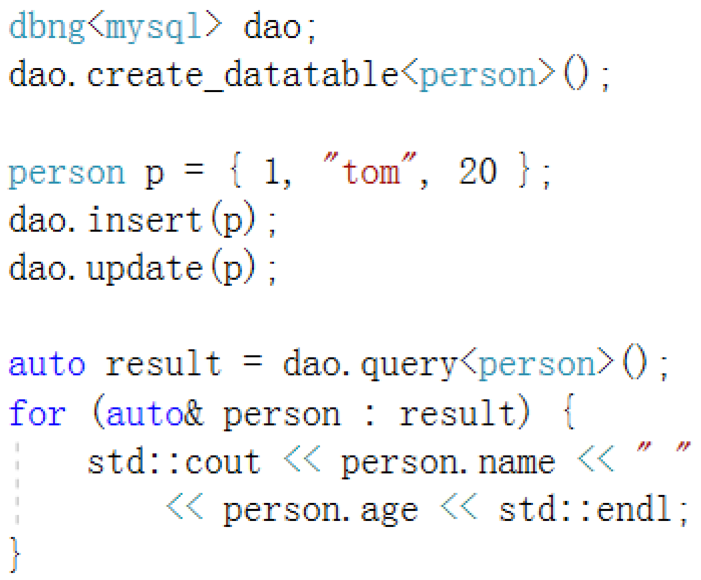
ormpp

- ormpp: an easy to use ORM library based on compile-time reflection developed in c++17

You can use ormpp to operate different kinds of databases in the same code



https://github.com/qicosmos/ormpp



```
dbng<mysql> dao;
dao.create_datatable<person>();

person p = { 1, "tom", 20 };
dao.insert(p);
dao.update(p);

auto result = dao.query<person>();
for (auto& person : result) {
    std::cout << person.name << " "
        << person.age << std::endl;
}
```

```
struct person
{
   int id;
   std::string name;
   int age;
};
REFLECTION(person, id, name, age)
```

How to change the database?

dbng<**postgresql**> dao;
dbng<**sqlite**> dao;

Ormpp make database programming simple
Ormpp hides databases differences
Accessing different databases with the same code

ormpp

- Unify interfaces
- Generate sql automatically
- Mapping table data to objects

ormpp

mysql

```
if (mysql_real_connect(con, "127.0.0.1", "root", "12345",
                        "testdb", 0, NULL, 0) == NULL)    {
    finish_with_error(con);
}
```

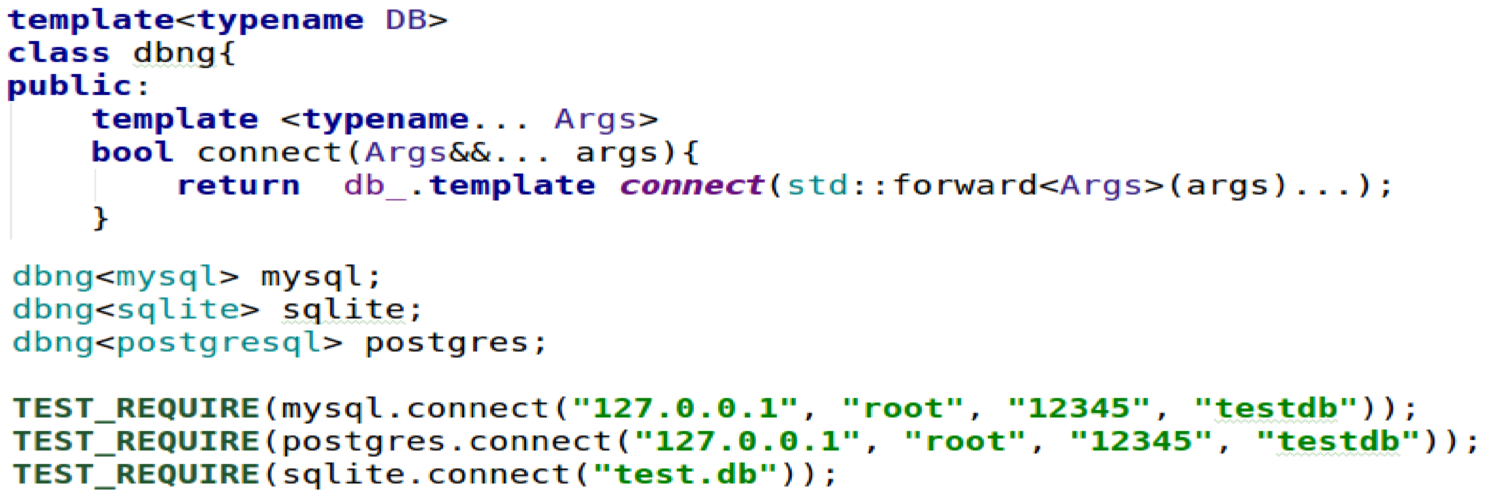
postgresql

```
PGconn *conn = PQconnectdb("host=127.0.0.1 user=root password=12345 dbname=testdb");
if (PQstatus(conn) != CONNECTION_OK)
    error(PQerrorMessage(conn));
```

sqlite

```
sqlite3* handle_ = NULL;
sqlite3_open("test.db", &handle_);
```

ormpp



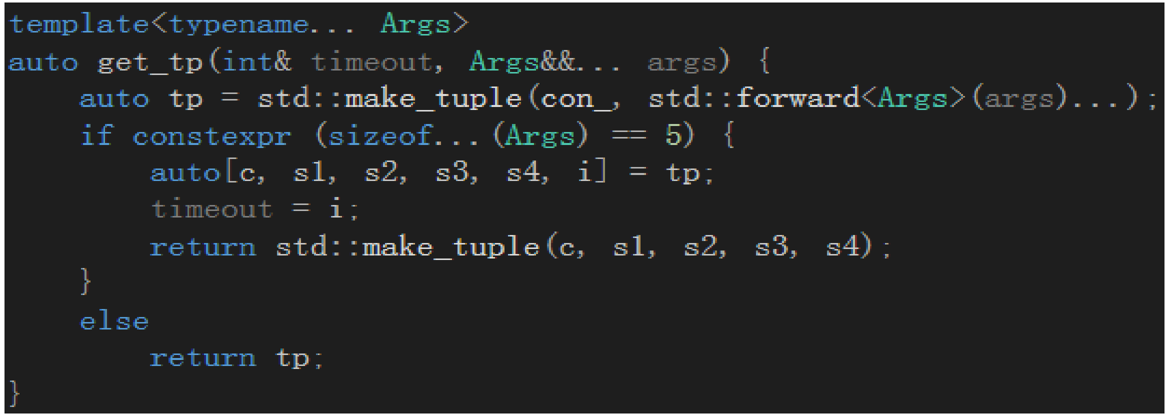
```
template<typename DB>
class dbng{
public:
    template <typename... Args>
    bool connect(Args&&... args){
        return  db_.template connect(std::forward<Args>(args)...);
    }

dbng<mysql> mysql;
dbng<sqlite> sqlite;
dbng<postgresql> postgres;

TEST_REQUIRE(mysql.connect("127.0.0.1", "root", "12345", "testdb"));
TEST_REQUIRE(postgres.connect("127.0.0.1", "root", "12345", "testdb"));
TEST_REQUIRE(sqlite.connect("test.db"));
```

Unified interface with variadic templates

Policy based design make it easy to change database

ormpp

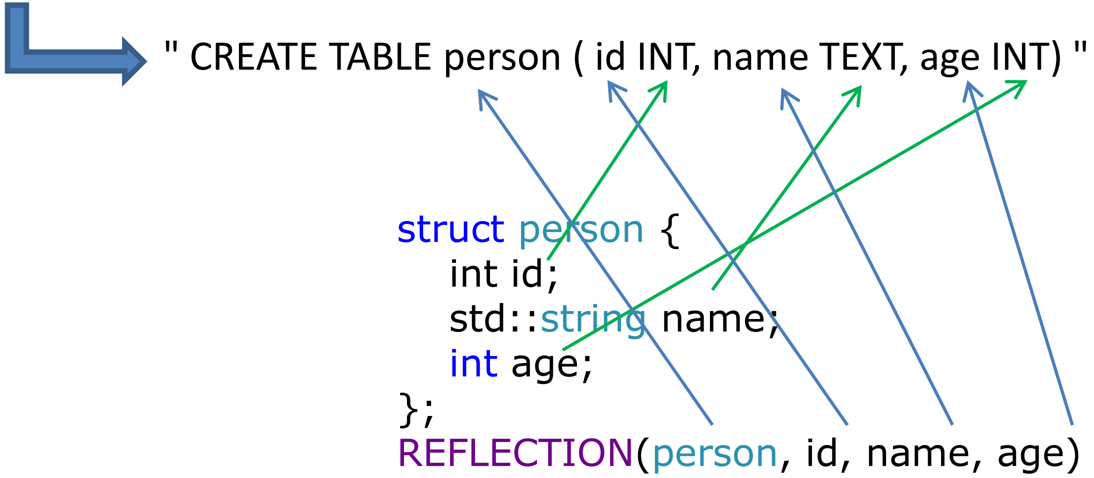
- implement static polymorphism with constexpr if and variadic templates



```cpp
template<typename... Args>
auto get_tp(int& timeout, Args&&... args) {
    auto tp = std::make_tuple(con_, std::forward<Args>(args)...);
    if constexpr (sizeof...(Args) == 5) {
        auto[c, s1, s2, s3, s4, i] = tp;
        timeout = i;
        return std::make_tuple(c, s1, s2, s3, s4);
    }
    else
        return tp;
}
```

ormpp-- Automatically generate sql

```
mysql.create_datatable<person>();
```



```
" CREATE TABLE person ( id INT, name TEXT, age INT) "
```

```
struct person {
    int id;
    std::string name;
    int age;
};
REFLECTION(person, id, name, age)
```

<<Compile-time Reflection, Serialization and ORM Examples>>
https://github.com/CppCon/CppCon2017
https://www.youtube.com/watch?v=WlhoWjrR41A

generate sql

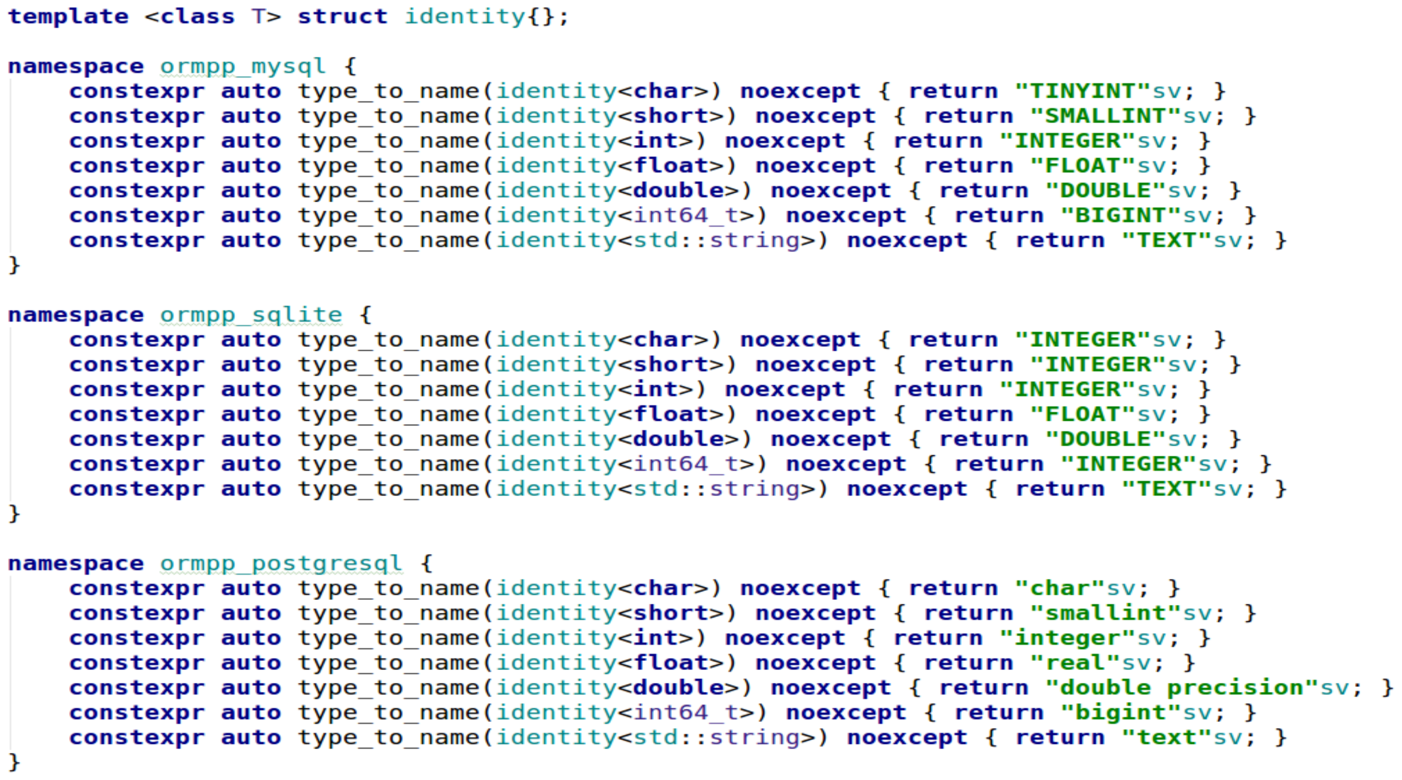
- generate sql by compile-time reflection
  - Get the table name
  - Get the fields name of a table
  - Get the fields type

constexpr auto table_name = iguana::get_name<T>(); → "person"

constexpr auto arr = iguana::get_array<T>(); → {"id" , "name" , "age" }

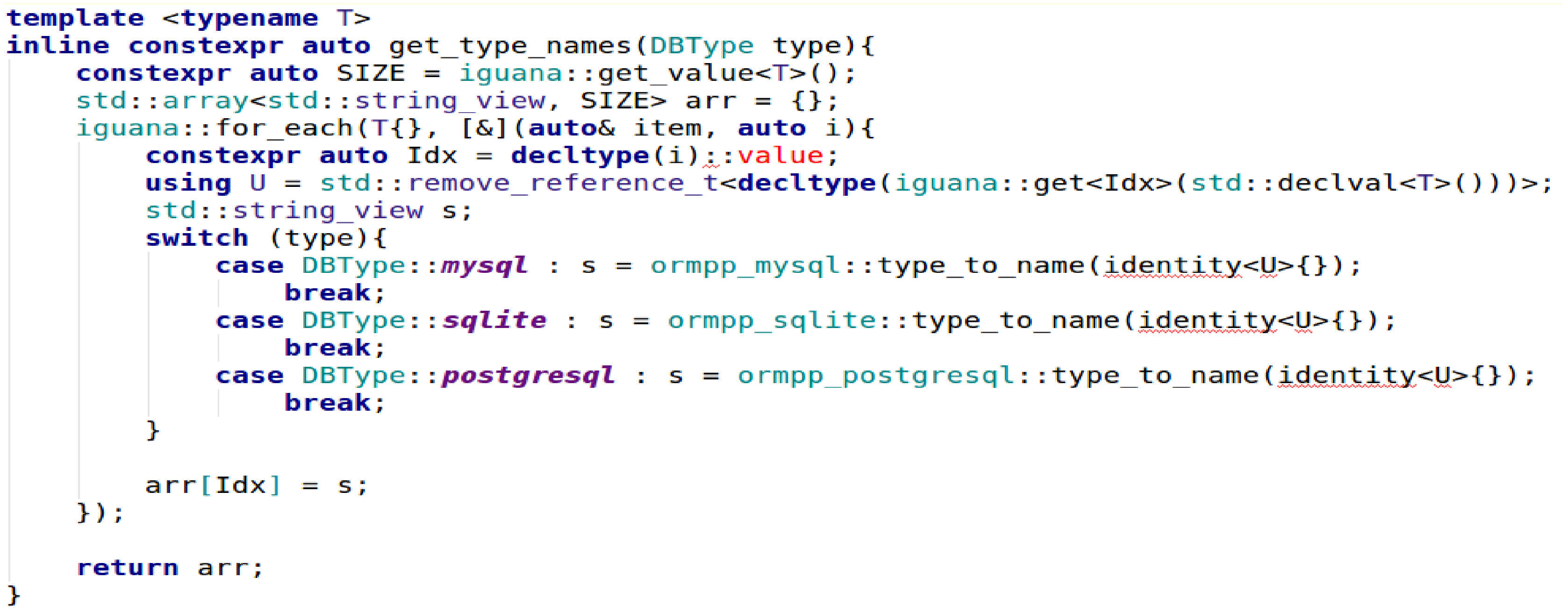
How to get the fields type name? →{"INTEGER" , "TEXT”, "INTEGER" }

Type mapping



```cpp
template <class T> struct identity{};

namespace ormpp_mysql {
    constexpr auto type_to_name(identity<char>) noexcept { return "TINYINT"sv; }
    constexpr auto type_to_name(identity<short>) noexcept { return "SMALLINT"sv; }
    constexpr auto type_to_name(identity<int>) noexcept { return "INTEGER"sv; }
    constexpr auto type_to_name(identity<float>) noexcept { return "FLOAT"sv; }
    constexpr auto type_to_name(identity<double>) noexcept { return "DOUBLE"sv; }
    constexpr auto type_to_name(identity<int64_t>) noexcept { return "BIGINT"sv; }
    constexpr auto type_to_name(identity<std::string>) noexcept { return "TEXT"sv; }
}

namespace ormpp_sqlite {
    constexpr auto type_to_name(identity<char>) noexcept { return "INTEGER"sv; }
    constexpr auto type_to_name(identity<short>) noexcept { return "INTEGER"sv; }
    constexpr auto type_to_name(identity<int>) noexcept { return "INTEGER"sv; }
    constexpr auto type_to_name(identity<float>) noexcept { return "FLOAT"sv; }
    constexpr auto type_to_name(identity<double>) noexcept { return "DOUBLE"sv; }
    constexpr auto type_to_name(identity<int64_t>) noexcept { return "INTEGER"sv; }
    constexpr auto type_to_name(identity<std::string>) noexcept { return "TEXT"sv; }
}

namespace ormpp_postgresql {
    constexpr auto type_to_name(identity<char>) noexcept { return "char"sv; }
    constexpr auto type_to_name(identity<short>) noexcept { return "smallint"sv; }
    constexpr auto type_to_name(identity<int>) noexcept { return "integer"sv; }
    constexpr auto type_to_name(identity<float>) noexcept { return "real"sv; }
    constexpr auto type_to_name(identity<double>) noexcept { return "double precision"sv; }
    constexpr auto type_to_name(identity<int64_t>) noexcept { return "bigint"sv; }
    constexpr auto type_to_name(identity<std::string>) noexcept { return "text"sv; }
}
```

Get mapped type name



```cpp
template <typename T>
inline constexpr auto get_type_names(DBType type){
    constexpr auto SIZE = iguana::get_value<T>();
    std::array<std::string_view, SIZE> arr = {};
    iguana::for_each(T{}, [&](auto& item, auto i){
        constexpr auto Idx = decltype(i)::value;
        using U = std::remove_reference_t<decltype(iguana::get<Idx>(std::declval<T>()))>;
        std::string_view s;
        switch (type){
            case DBType::mysql : s = ormpp_mysql::type_to_name(identity<U>{});
                break;
            case DBType::sqlite : s = ormpp_sqlite::type_to_name(identity<U>{});
                break;
            case DBType::postgresql : s = ormpp_postgresql::type_to_name(identity<U>{});
                break;
        }

        arr[Idx] = s;
    });

    return arr;
}
```

generate sql automatically

```cpp
struct person {
    int id;
    std::string name;
    int age;
};
REFLECTION(person, id, name, age)
```

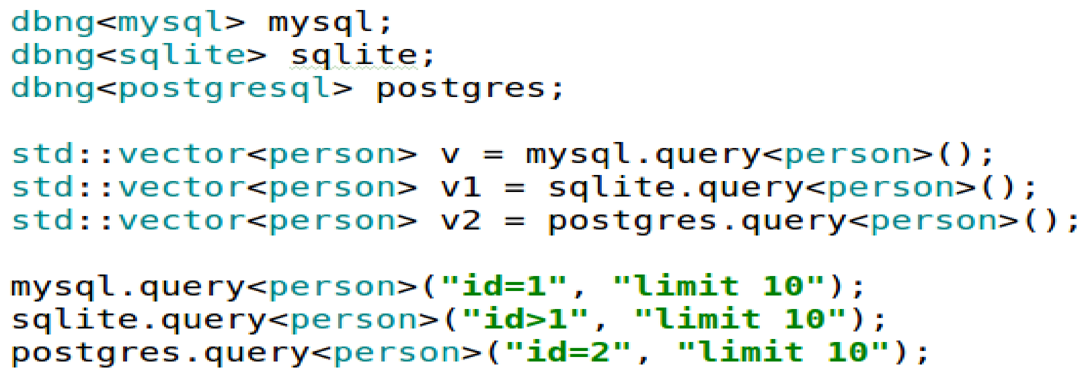
generate sql →

" CREATE TABLE person (id INT, name TEXT, age INT) "

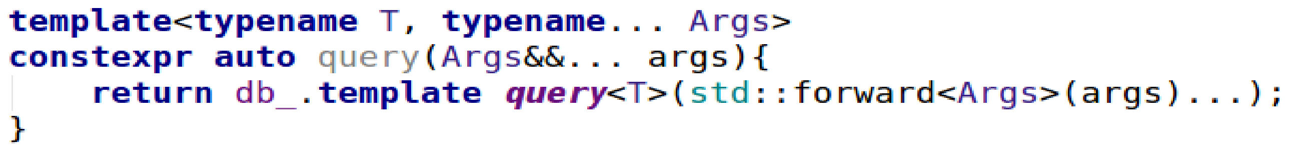
```cpp
template<typename T, typename... Args>
constexpr auto create_datatable(Args&&... args){
    return db_.template create_datatable<T>(std::forward<Args>(args)...);
}
```

use an object to do everything what you want

entity mapping



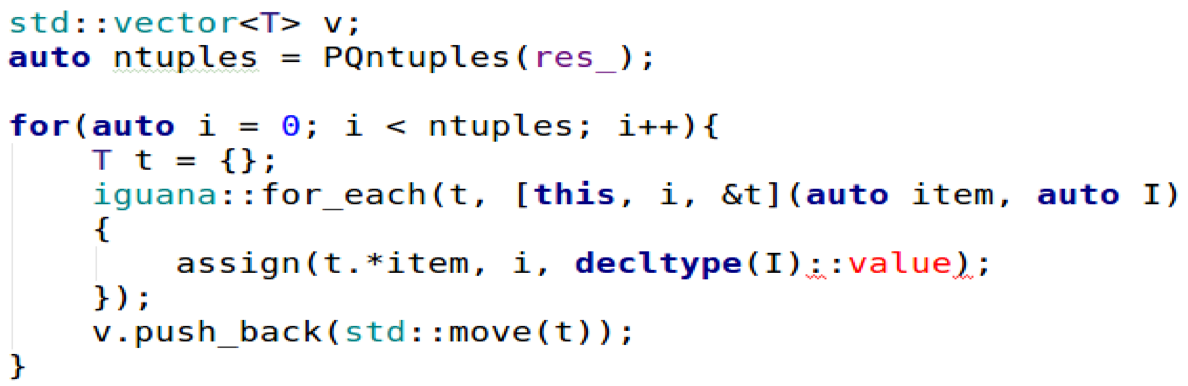
```cpp
dbng<mysql> mysql;
dbng<sqlite> sqlite;
dbng<postgresql> postgres;

std::vector<person> v = mysql.query<person>();
std::vector<person> v1 = sqlite.query<person>();
std::vector<person> v2 = postgres.query<person>();

mysql.query<person>("id=1", "limit 10");
sqlite.query<person>("id>1", "limit 10");
postgres.query<person>("id=2", "limit 10");
```



```cpp
template<typename T, typename... Args>
constexpr auto query(Args&&... args){
    return db_.template query<T>(std::forward<Args>(args)...);
}
```

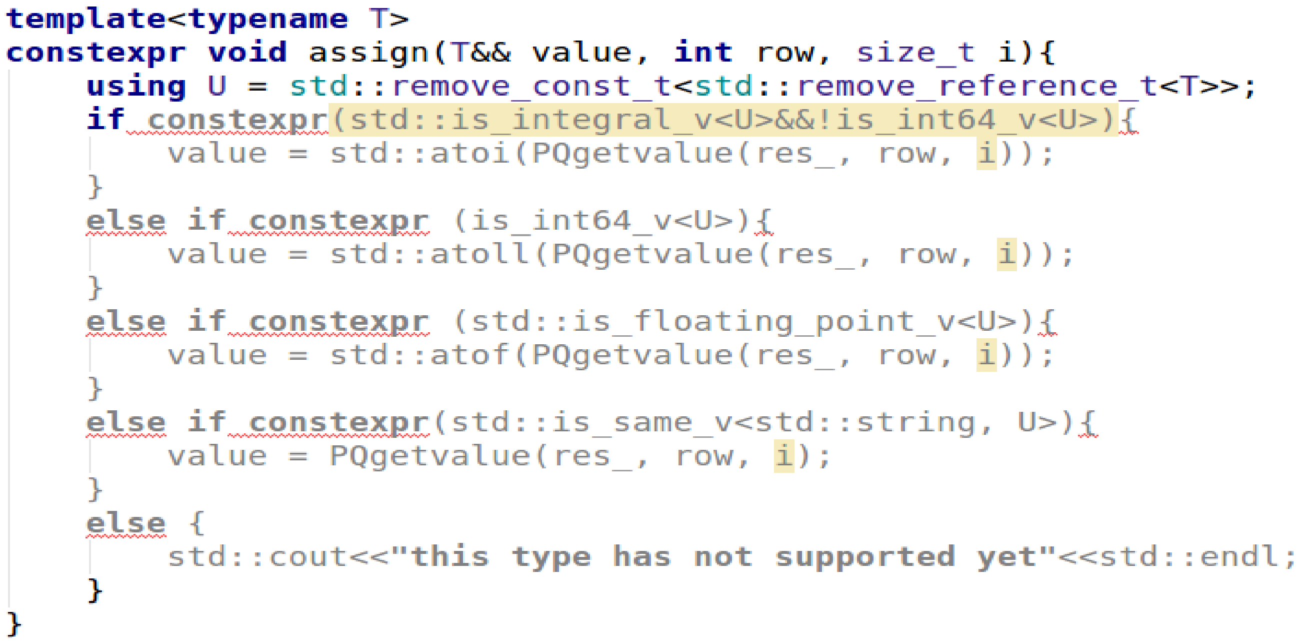
entity mapping

- Data table to object



```
std::vector<T> v;
auto ntuples = PQntuples(res_);

for(auto i = 0; i < ntuples; i++){
    T t = {};
    iguana::for_each(t, [this, i, &t](auto item, auto I)
    {
        assign(t.*item, i, decltype(I)::value);
    });
    v.push_back(std::move(t));
}
```

entity mapping



```cpp
template<typename T>
constexpr void assign(T&& value, int row, size_t i){
    using U = std::remove_const_t<std::remove_reference_t<T>>;
    if constexpr(std::is_integral_v<U>&&!is_int64_v<U>){
        value = std::atoi(PQgetvalue(res_, row, i));
    }
    else if constexpr (is_int64_v<U>){
        value = std::atoll(PQgetvalue(res_, row, i));
    }
    else if constexpr (std::is_floating_point_v<U>){
        value = std::atof(PQgetvalue(res_, row, i));
    }
    else if constexpr(std::is_same_v<std::string, U>){
        value = PQgetvalue(res_, row, i);
    }
    else {
        std::cout<<"this type has not supported yet"<<std::endl;
    }
}
```

ormpp

- Unified interface with variadic temlates and constexpr if
- Generate sql with compile-time reflection
- Mapping entity with compile-time reflection

Render

- Html template engine
  - Fill html pages with dynamic data
  - Control UI display
  - Reuse html code

Render

```
t["value"] = 100;
t["map"] = std::map<std::string, int>{ { "hoge", 1 }, { "fuga", 2 } };
TEST_EQ_T("${value}", "100", t);
TEST_EQ_T("${map.hoge}, ${map.fuga}", "1, 2", t);

TEST_EQ_T("$if true{{hoge}}$elseif true{{fuga}}", "hoge", t);
TEST_EQ_T("$if true{{hoge}}$elseif undefined{{${undefined}}}", "hoge", t);

TEST_EQ_T("$for x in xs{{$for x in xs{{test}}}}", "testtesttesttest", t);
TEST_EQ_T("$for y in ys.hoge{{${y}}}", "123", t);
```

Custom script in html pages
Custom script parser

```
<$for> = $for <var-name> in <var> {{ <block> }}
<$if> = $if <var> {{ <block> }} ($elseif <var> {{ <block> }})? ($else {{ <block> }})?
<$variable> = ${<var>}
```

https://github.com/qicosmos/render
https://github.com/melpon/ginger

Render

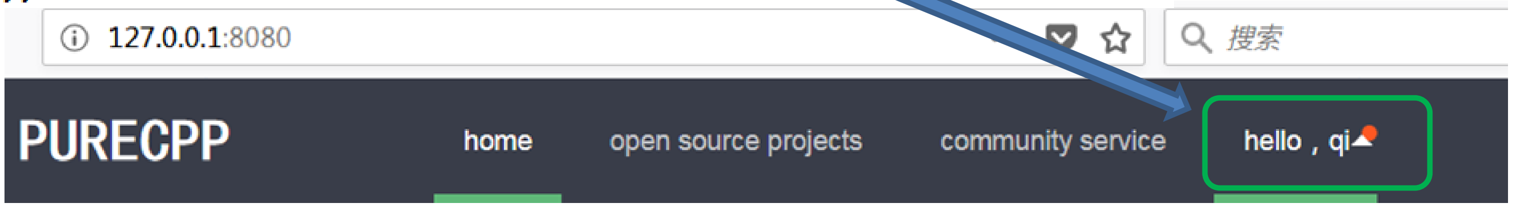
```
auto login_user_name = get_user_name_from_session(req);
nlohmann::json result;
result["has_login"] = !login_user_name.empty();
result["login_user_name"] = login_user_name;
res.add_header("Content-Type", "text/html; charset=utf-8");
res.set_status_and_content(status_type::ok, render::render_file("./purecpp/html/login.html", result));
```

Fill html pages with dynamic data

fill

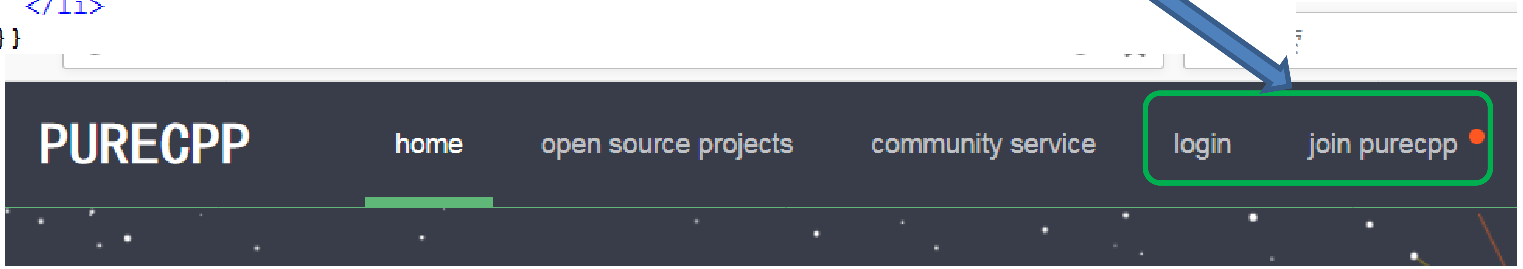
```
$if has_login{{
  <li class="layui-nav-item">
    <a href="" style="color:white" >hello, ${login_user_name}</a>
  </li>
}}
$else{{
  <li class="layui-nav-item $if category==login {{layui-this}}">
    <a href="login_page">login</a>
  </li>
  <li class="layui-nav-item $if category==sign_out {{layui-this}}">
    <a href="sign_out_page">join purecpp<span class="layui-badge-dot"></span></a>
  </li>
}}
```

```
$if has_login{{
  <li class="layui-nav-item">
    <a href="" style="color:white" >hello, ${login_user_name}</a>
  </li>
}}
```

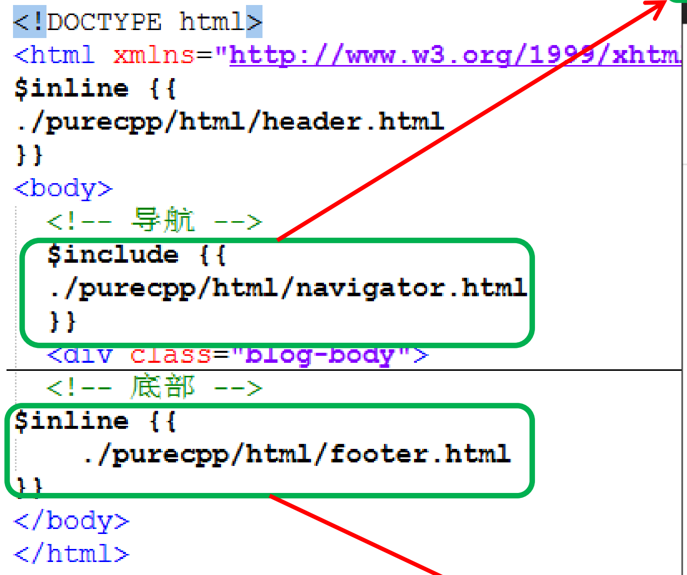


```
$else{{
  <li class="layui-nav-item $if category==login {{layui-this}}">
    <a href="login_page">login</a>
  </li>
  <li class="layui-nav-item $if category==sign_out {{layui-this}}">
    <a href="sign_out_page">join purecpp<span class="layui-badge-dot"></span></a>
  </li>
}}
```

Control UI display

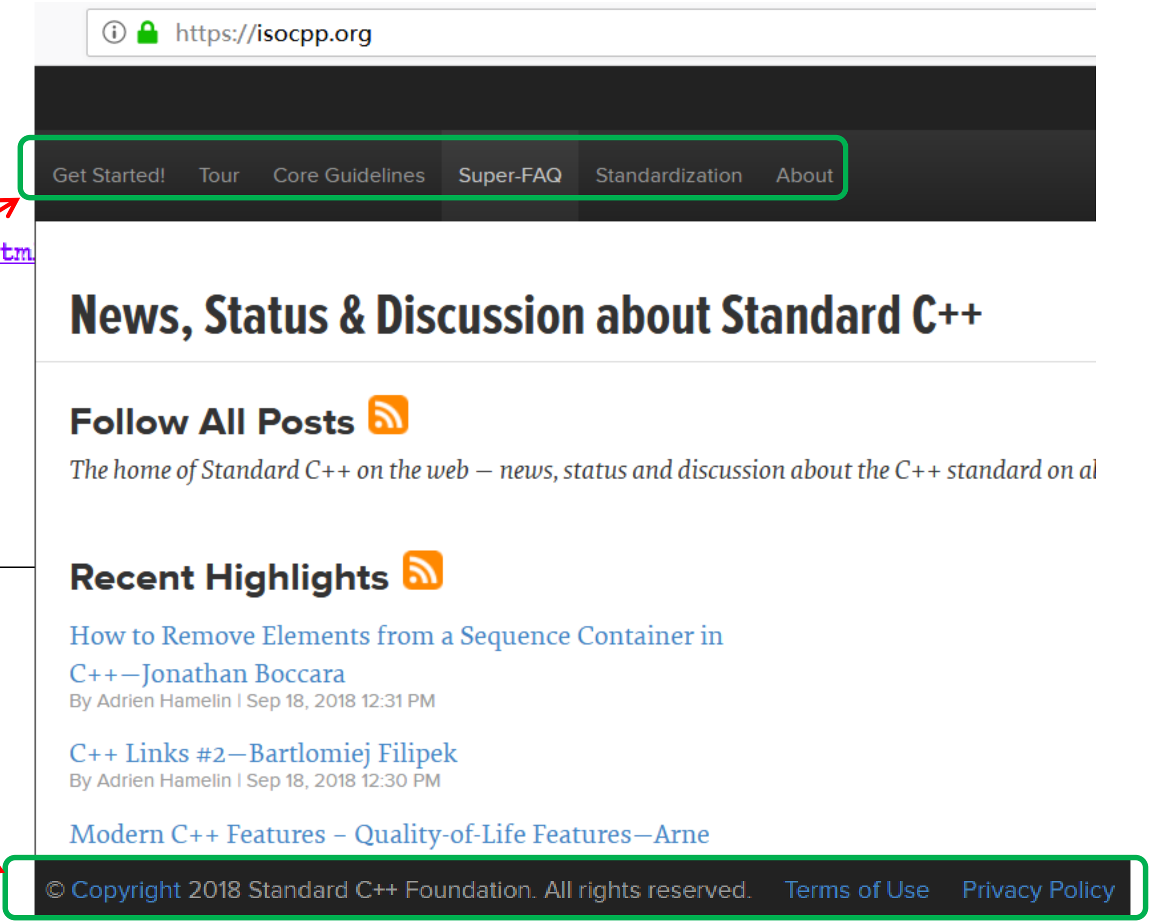


render



```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtm
$inline {{
./purecpp/html/header.html
}}
<body>
  <!-- 导航 -->
  $include {{
  ./purecpp/html/navigator.html
  }}
  <div class="blog-body">
  <!-- 底部 -->
$inline {{
    ./purecpp/html/footer.html
}}
</body>
</html>
```

Reuse html code



https://isocpp.org

Get Started! Tour Core Guidelines Super-FAQ Standardization About

News, Status & Discussion about Standard C++

Follow All Posts

The home of Standard C++ on the web — news, status and discussion about the C++ standard on al

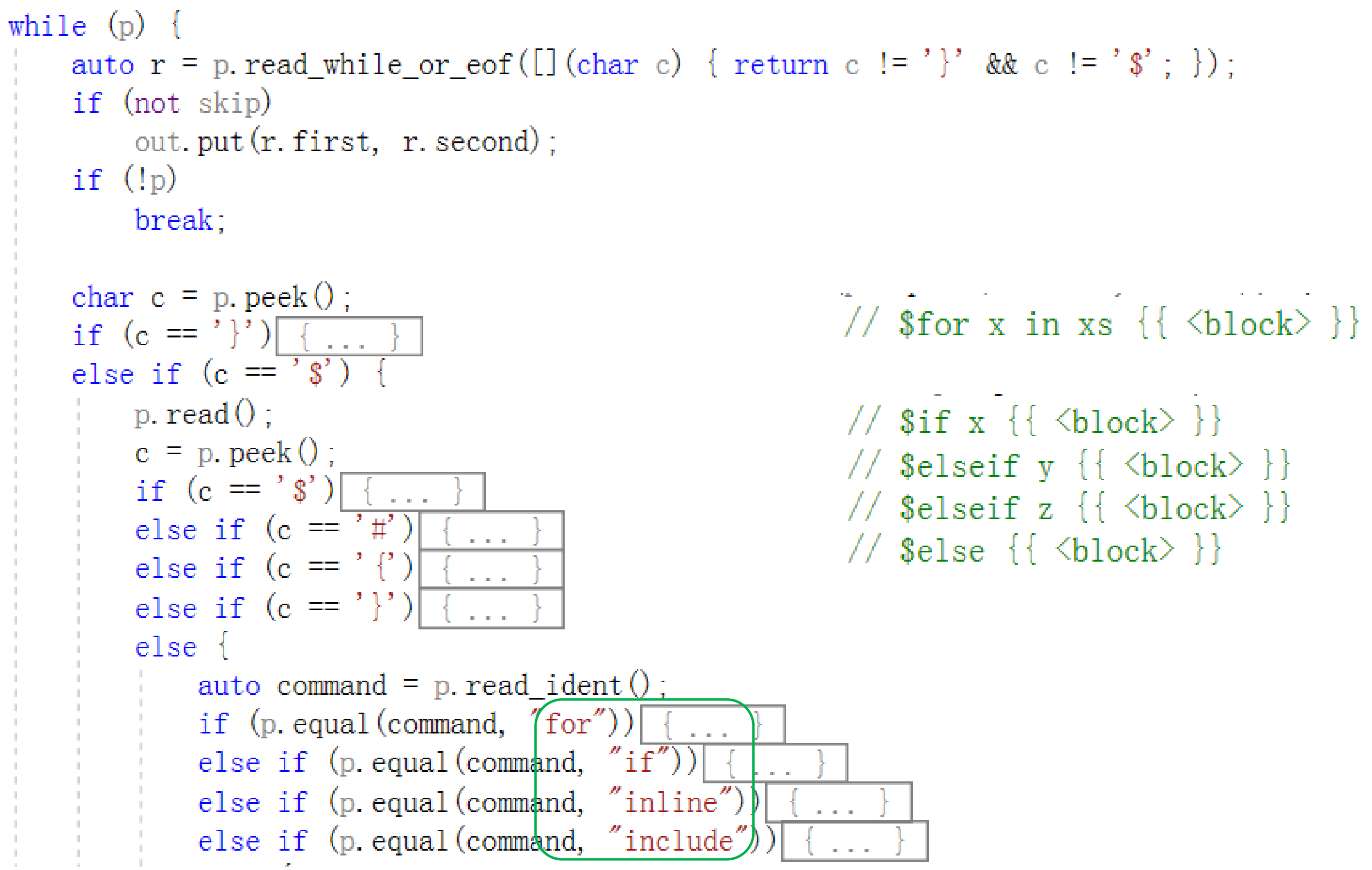
Recent Highlights

How to Remove Elements from a Sequence Container in C++—Jonathan Boccara
By Adrien Hamelin | Sep 18, 2018 12:31 PM

C++ Links #2—Bartlomiej Filipek
By Adrien Hamelin | Sep 18, 2018 12:30 PM

Modern C++ Features – Quality-of-Life Features—Arne

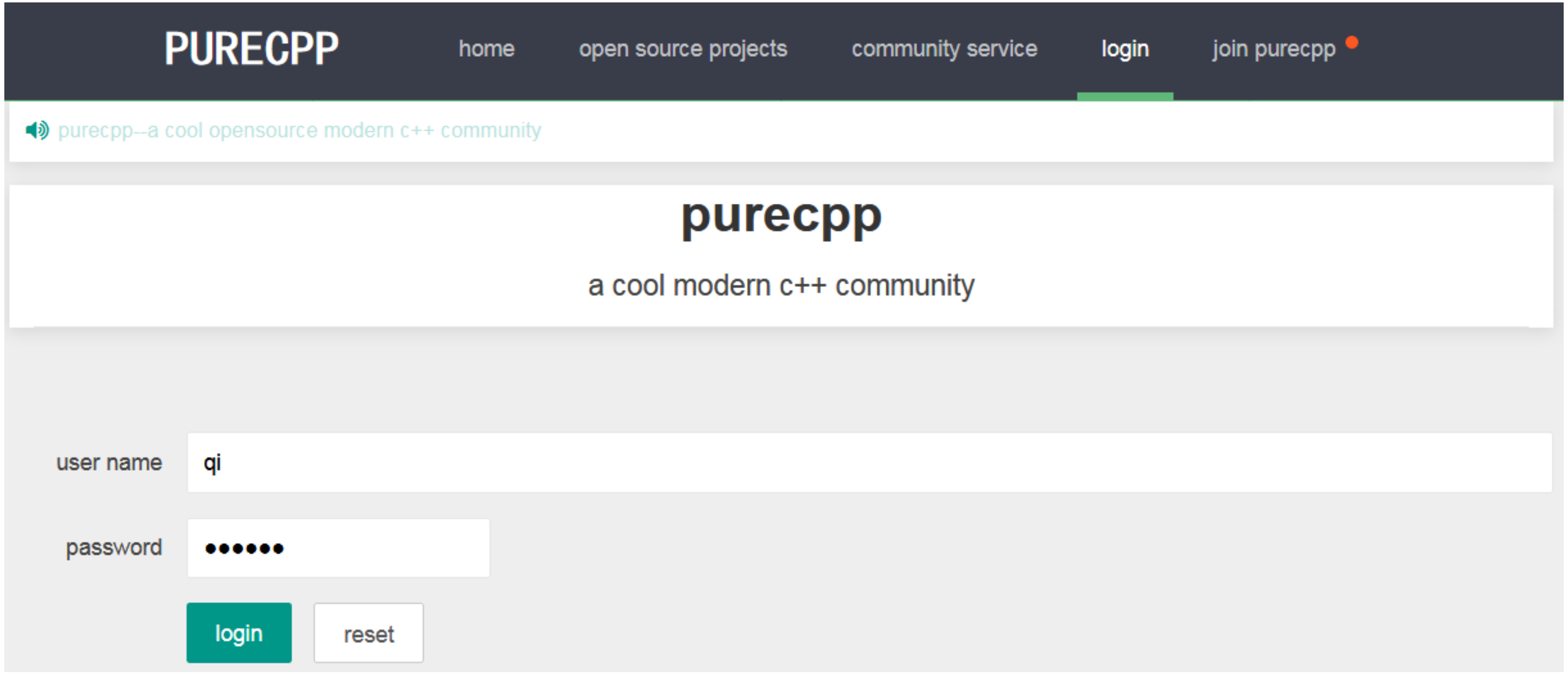
© Copyright 2018 Standard C++ Foundation. All rights reserved. Terms of Use Privacy Policy



```
while (p) {
    auto r = p.read_while_or_eof([](char c) { return c != '}' && c != '$'; });
    if (not skip)
        out.put(r.first, r.second);
    if (!p)
        break;

    char c = p.peek();
    if (c == '}') { ... }
    else if (c == '$') {
        p.read();
        c = p.peek();
        if (c == '$') { ... }
        else if (c == '#') { ... }
        else if (c == '{') { ... }
        else if (c == '}') { ... }
        else {
            auto command = p.read_ident();
            if (p.equal(command, "for")) { ... }
            else if (p.equal(command, "if")) { ... }
            else if (p.equal(command, "inline")) { ... }
            else if (p.equal(command, "include")) { ... }
```

```
// $for x in xs {{ <block> }}

// $if x {{ <block> }}
// $elseif y {{ <block> }}
// $elseif z {{ <block> }}
// $else {{ <block> }}
```

Example—login



purecpp--a cool opensource modern c++ community

purecpp

a cool modern c++ community

user name	qi
password	••••••

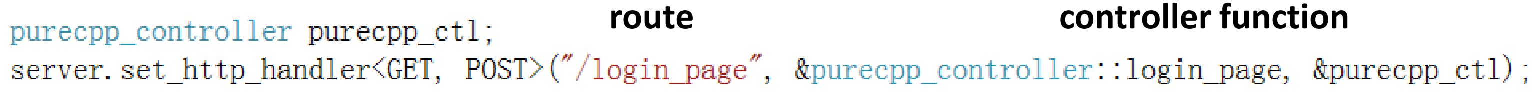
Login html code

```html
<div class="blogerinfo shadow">
    <p class="blogerinfo-nickname"><a href="/">purecpp</a></p>
    <p class="blogerinfo-introduce">a cool modern c++ community</p>
    <hr />
  </div>
<div class="blog-body">
  <form class="layui-form" method="post" action="/login">
      <div class="layui-form-item">
        <label class="layui-form-label">user name</label>
        <div class="layui-input-block">
          <input type="text" name="user_name" required  lay-verify="required" autocomplete="off" class="layui-input">
        </div>
      </div>
      <div class="layui-form-item">
        <label class="layui-form-label">password</label>
        <div class="layui-input-inline layui-input-password">
          <input type="password" name="password" required lay-verify="required" autocomplete="off" class="layui-input">
        </div>
      </div>
      <div class="layui-form-item">
        <div class="layui-input-block">
          <button class="layui-btn" lay-submit="login" lay-filter="login">login</button>
          <button type="reset" class="layui-btn layui-btn-primary">reset</button>
        </div>
      </div>
    </form>
</div>
```

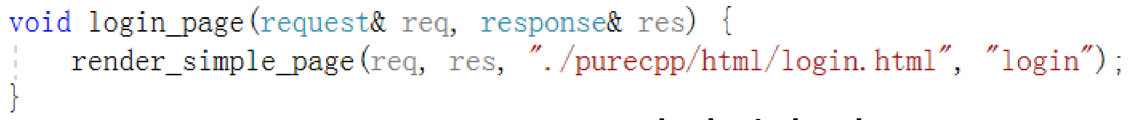
Controller function

http://127.0.0.1/login_page

route **controller function**



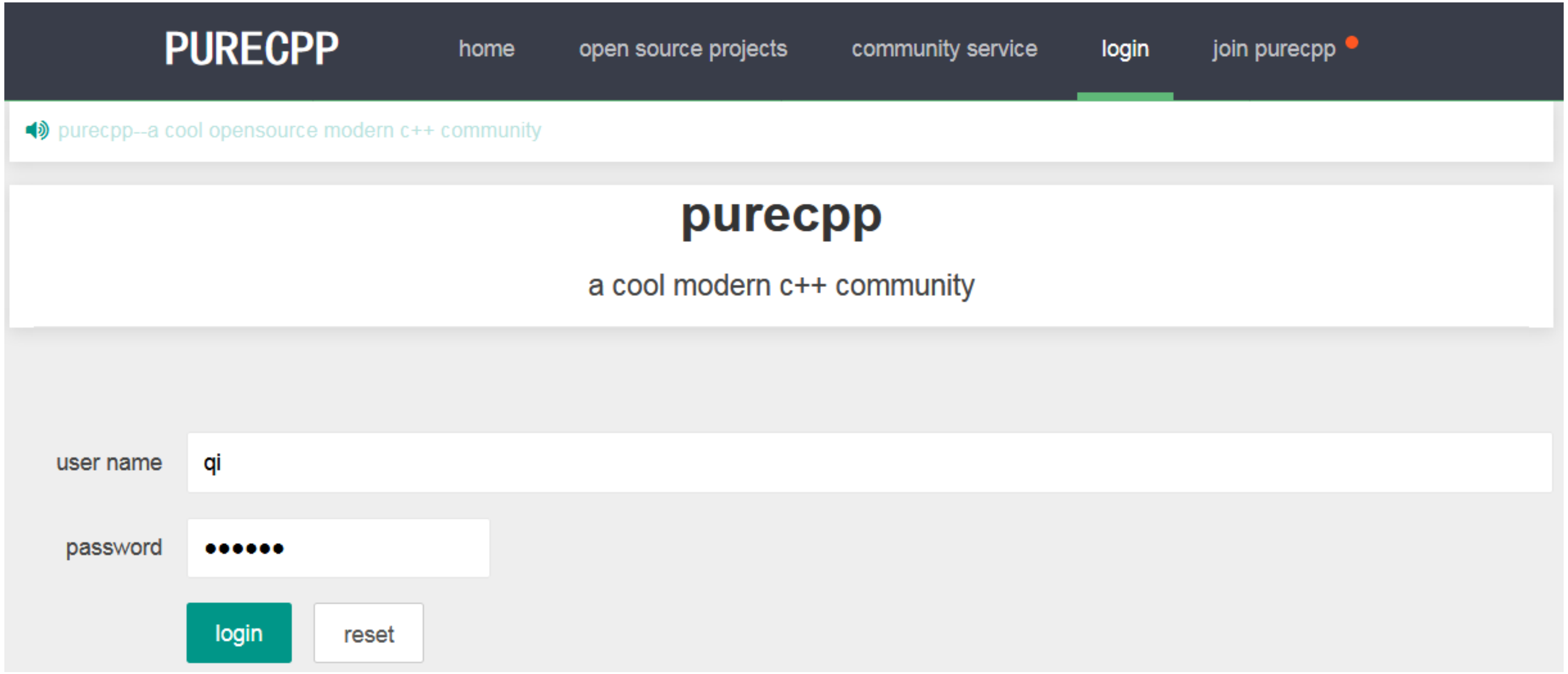
```
purecpp_controller purecpp_ctl;
server.set_http_handler<GET, POST>("/login_page", &purecpp_controller::login_page, &purecpp_ctl);
```



```
void login_page(request& req, response& res) {
    render_simple_page(req, res, "./purecpp/html/login.html", "login");
}
```

render login html page

Example—login



purecpp--a cool opensource modern c++ community

purecpp

a cool modern c++ community

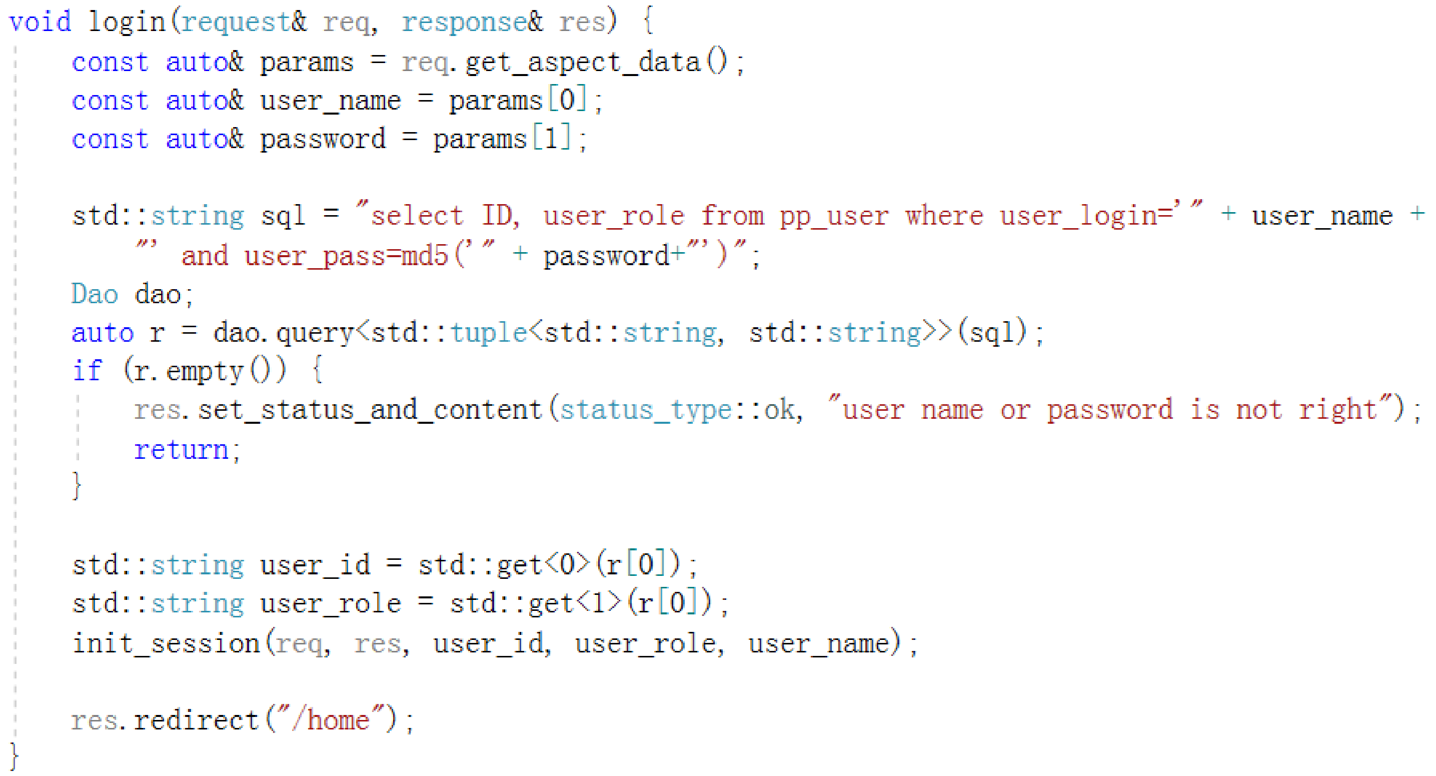
user name: qi

password: ●●●●●●

```
purecpp_controller purecpp_ctl;
server.set_http_handler<GET, POST>("/login", &purecpp_controller::login,
    &purecpp_ctl, check_login_input{});
struct check_login_input {
    bool before(request& req, response& res) {
        auto user_name = req.get_query_value("user_name");
        auto password = req.get_query_value("password");
        if (len_more_than<255>(user_name, password)) {
            res.set_status_and_content(status_type::bad_request, "the input parameter is too long");
            return false;
        }

        if (!check_input(res, user_name, password)) {
            return false;
        }

        req.set_aspect_data(sv2s(user_name), sv2s(password));
        return true;
    }
};
```

Separate non-core business

Simple and direct core business code



```
void login(request& req, response& res) {
    const auto& params = req.get_aspect_data();
    const auto& user_name = params[0];
    const auto& password = params[1];

    std::string sql = "select ID, user_role from pp_user where user_login='" + user_name +
        "' and user_pass=md5('" + password+"')";
    Dao dao;
    auto r = dao.query<std::tuple<std::string, std::string>>(sql);
    if (r.empty()) {
        res.set_status_and_content(status_type::ok, "user name or password is not right");
        return;
    }

    std::string user_id = std::get<0>(r[0]);
    std::string user_role = std::get<1>(r[0]);
    init_session(req, res, user_id, user_role, user_name);

    res.redirect("/home");
}
```

How to develop web rapidly

- Focus on business function
- Separate core business and non-core business
- Utilize the framework to help you to solve trivial details
  - Use cinatra to solve http details, just need focus on the parsed results
  - Use ormpp to solve the database details, just need focus on objects mapped from the database
  - Use render to solve html details, just need focus on how to fill the html templates

FAQ

Thank you