Contemporary C++ Web Scraping

It's not as low level as one might think

Darrell Wright
@beached_whale

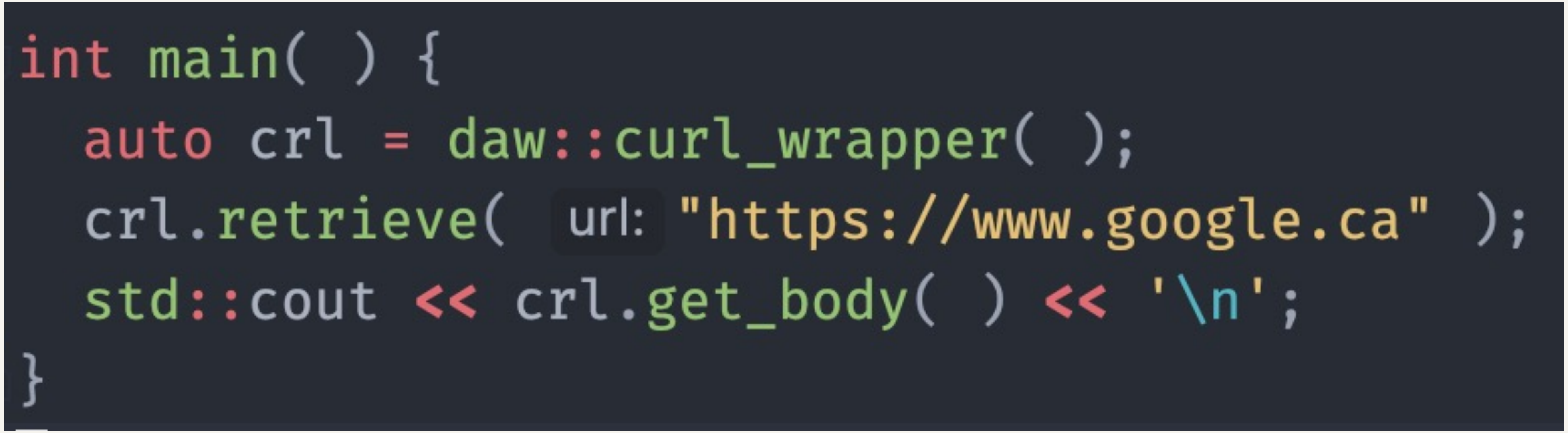
```
allbac
inks_with_text(
                                     st
                                     Cal
amespace match = daw::gumbo::m
auto node_rng = gumbo::gumbo_nod
auto const matcher : const match_all<...>
match::tag::A and
match::attribute::value::starts_wit
match::content_text::contains( c: que
daw::algorithm::for_each_if(
first: node_rng.begin( ),
last: node_rng.end( ),
pred: matcher,
onEach: [&]( GumboNode const &node ) {
auto uri : basic_string_view<...> = daw::gumbo::nod
to title : string = daw::gumbo::node_conten
)onEach( uri, title );
```

What is needed

- Retrieve documents
- Parse documents
- Query documents

Retrieving Documents

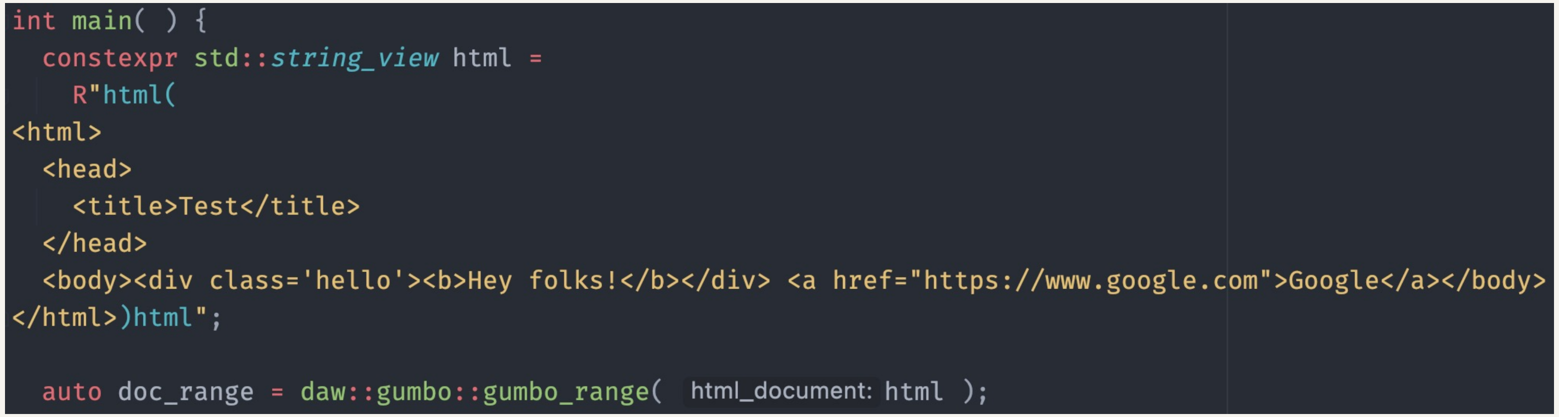
- Surprise, it's Curl.
- Using a simple Curl wrapper
- https://github.com/beached/curl_wrapper



```
int main( ) {
  auto crl = daw::curl_wrapper( );
  crl.retrieve( url: "https://www.google.ca" );
  std::cout << crl.get_body( ) << '\n';
}
```

Parse Documents

- Wrapping Gumbo to get document tree
- https://github.com/beached/gumbo_pp
- https://github.com/google/gumbo-parser
- We have an iterator interface into the document tree that uses DFS ordering



```
int main( ) {
  constexpr std::string_view html =
    R"html(
<html>
  <head>
    <title>Test</title>
  </head>
  <body><div class='hello'><b>Hey folks!</b></div> <a href="https://www.google.com">Google</a></body>
</html>)html";

  auto doc_range = daw::gumbo::gumbo_range(  html_document: html  );
```

Query Documents

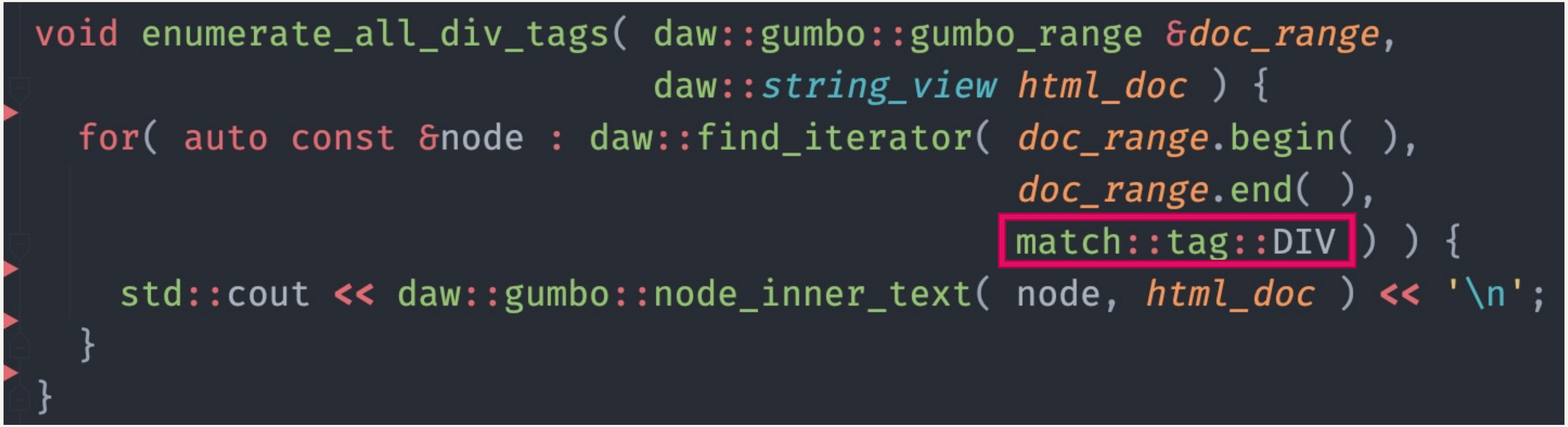
- Gumbo_pp provides a set of combinable predicates based on

attribute	class type	id	inner text
outer text	content text	tag	

- Each predicate type has associated verbs like attribute::is
- All have the where clause to allow for using custom matcher predicates
- They can be combined with and(match_all), or(match_any), not(negate match), xor(match_one) operators to form complex expressions
- Usable with std::algorithms, e.g. std::find_if, daw::algorithm::for_each_if
- Does not allocate unless asked to

Show me the Code

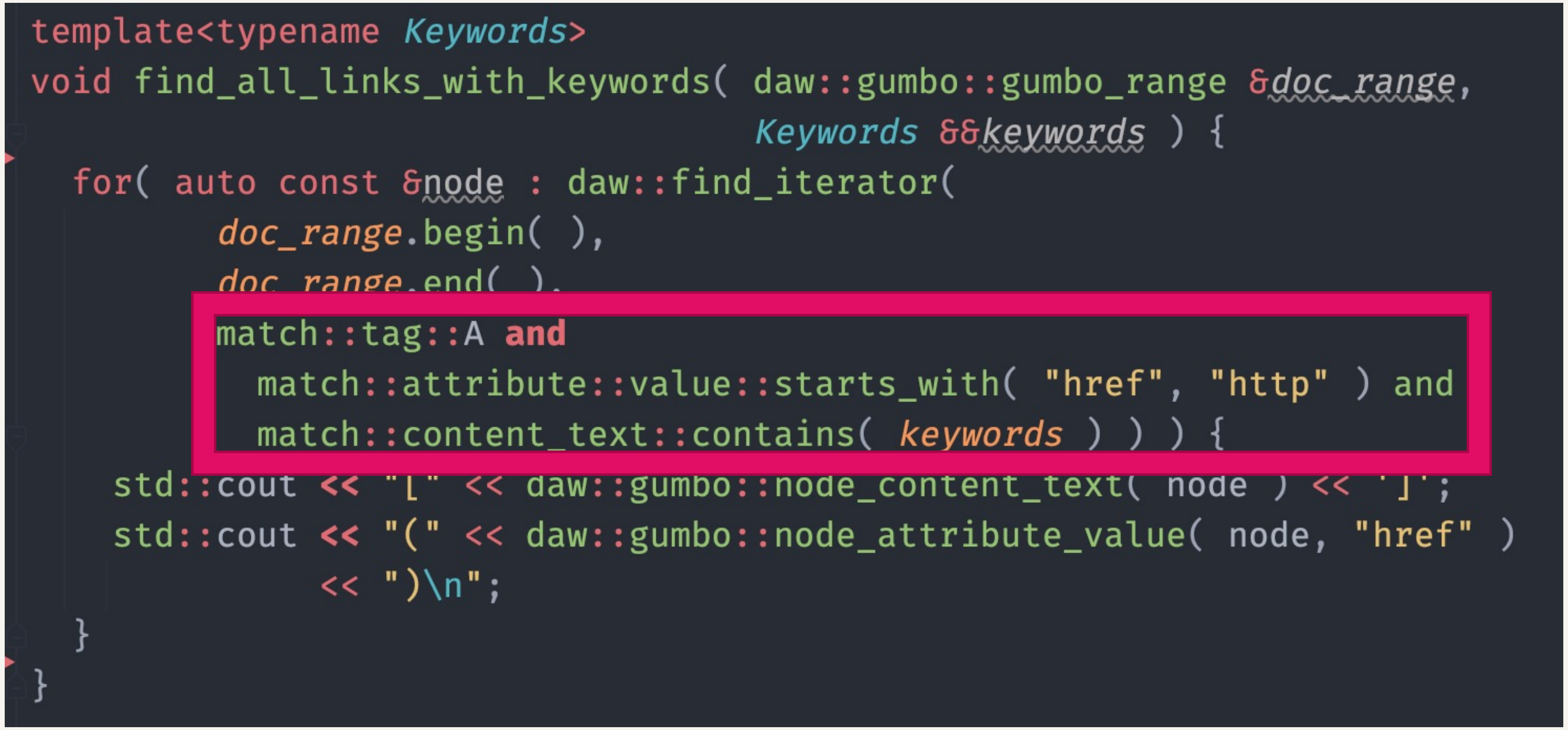
- Enumerate all div tags



```
void enumerate_all_div_tags( daw::gumbo::gumbo_range &doc_range,
                             daw::string_view html_doc ) {
  for( auto const &node : daw::find_iterator( doc_range.begin( ),
                                              doc_range.end( ),
                                              match::tag::DIV ) ) {
    std::cout << daw::gumbo::node_inner_text( node, html_doc ) << '\n';
  }
}
```

Show me the Code

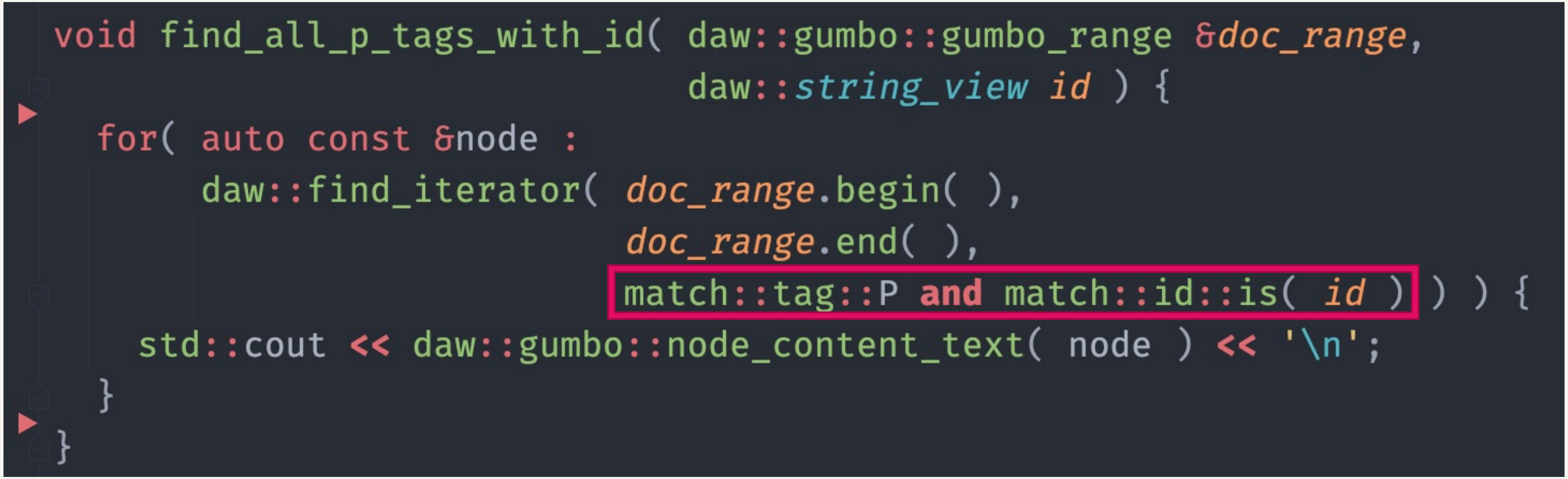
- Find all links that contain a keyword



```cpp
template<typename Keywords>
void find_all_links_with_keywords( daw::gumbo::gumbo_range &doc_range,
                                   Keywords &&keywords ) {
  for( auto const &node : daw::find_iterator(
         doc_range.begin( ),
         doc_range.end( ),
         match::tag::A and
           match::attribute::value::starts_with( "href", "http" ) and
           match::content_text::contains( keywords ) ) ) {
    std::cout << "[" << daw::gumbo::node_content_text( node ) << ']';
    std::cout << "(" << daw::gumbo::node_attribute_value( node, "href" )
              << ")\n";
  }
}
```

Show me the Code

- Find all Paragraph's with matching matching



```
void find_all_p_tags_with_id( daw::gumbo::gumbo_range &doc_range,
                              daw::string_view id ) {
  for( auto const &node :
       daw::find_iterator( doc_range.begin( ),
                           doc_range.end( ),
                           match::tag::P and match::id::is( id ) ) ) {
    std::cout << daw::gumbo::node_content_text( node ) << '\n';
  }
}
```

Examples

- Code from slides and slides

  https://github.com/beached/denver_cug_web_scraping

- Full example web service

  https://github.com/beached/climate_change_api_example

What’s Next

- Add a full Node type with accessors/matcher methods
- Use/write a library like Puppeteer/Selenium that uses headless Chrome/Firefox

Questions