Refactoring C++ with Clang

Let's make C++ more fun than Java

Performance

Productivity

Fun

Performance

Productivity

Fun

Performance **Excellent!**

Productivity

Fun

Performance **Excellent!**

Productivity **… mixed**

Fun

Performance **Excellent!**

Productivity **… mixed**

Fun **… hmmm**

From: PedanticGuy
Subject: [PATCH] Awesome new feature

On Wed, March 7, 2011 NewContributor wrote:
> + void DoSomethingAwesome(Cats cats);

ALL your methods are capitalized incorrectly...
Please fix. Thx.

Tooling & Automation

Tooling & Automation

- Code Formatting
- Renaming
- Boilerplate

Tooling & Automation is Hard!

- Cross-TU
- Build systems & editors
- SPEED!

Clang

Tooling

Clang

Tooling

Refactoring

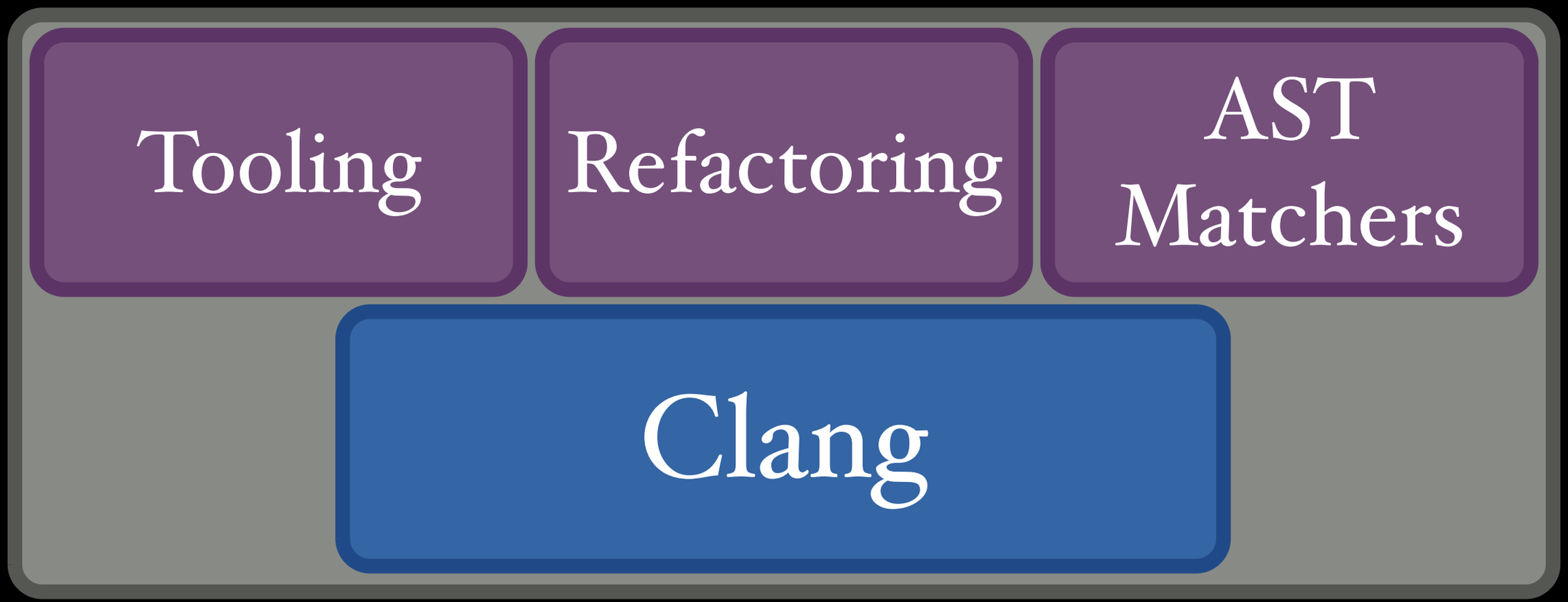
Clang

Tooling

Refactoring

AST Matchers

Clang


Tooling
Refactoring
AST Matchers
Clang

Plugins

Tooling

Refactoring

AST Matchers

Clang

Plugins

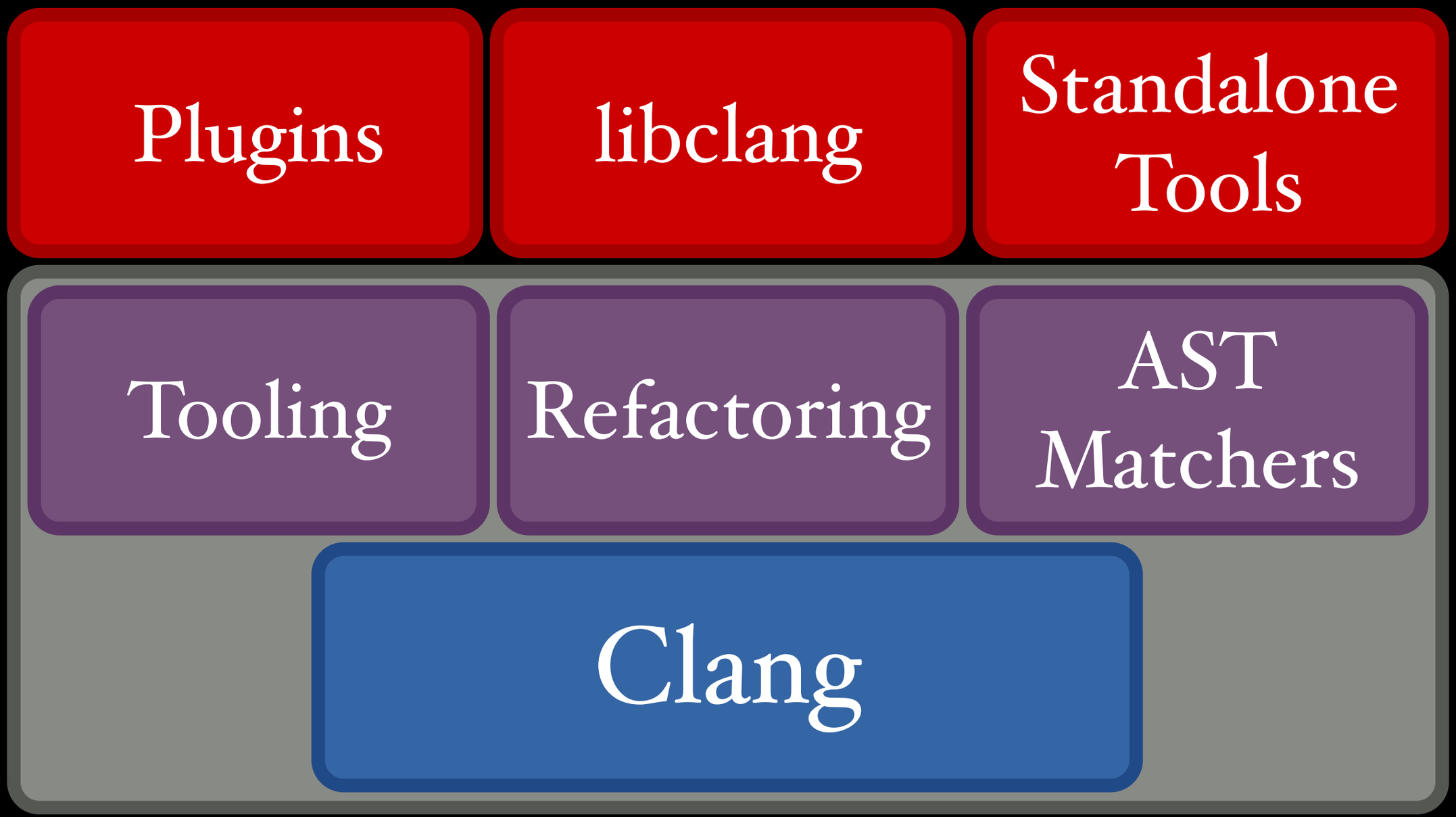
libclang

Tooling

Refactoring

AST Matchers

Clang


Plugins
libclang
Standalone Tools
Tooling
Refactoring
AST Matchers
Clang

Standalone Refactoring Tools

A Case Study:
Refactoring APIs

```
class MyElements
      : public ElementsBase {
  virtual const Element &Get();
};

Element f(const MyElements &E) {
  // ...
  return E.Get();
}
```

```
class MyElements
      : public ElementsBase {
   virtual const Element &Get();
};
```

```
Element f(const MyElements &E) {
   // ...
   return E.Get();
}
```

```
class MyElements
      : public ElementsBase {
  virtual const Element &Front();
};
```

```
Element f(const MyElements &E) {
  // ...
  return E.Front();
}
```

First, we need a tool to run over our code...

```
// Declare global command line flags.

cl::opt<std::string> BuildPath(
    cl::Positional,
    cl::desc("<build-path>"));

cl::list<std::string> SourcePaths(
    cl::Positional,
    cl::desc("<source0> [... <sourceN>]"),
    cl::OneOrMore);
```

```
int main(int argc, char **argv) {
  cl::ParseCommandLineOptions(argc, argv);
  string ErrorMessage;
  OwningPtr<CompilationDatabase> Compilations(
    CompilationDatabase::loadFromDirectory(
      BuildPath, ErrorMessage));
  if (!Compilations)
    report_fatal_error(ErrorMessage);
  RefactoringTool Tool(*Compilations,
                       SourcePaths);
  // ... magic ...
  return Tool.run(
    newFrontendActionFactory(...));
}
```

```
[
{
  "directory": "/work/project",
  "command":
    "/usr/bin/c++ -o code.cc.o -c code.cc",
  "file": "/work/project/code.cc"
},
{
  "directory": "/work/project",
  "command":
    "/usr/bin/c++ -o code2.cc.o -c code2.cc",
  "file": "/work/project/code2.cc"
},
...
```

```
int main(int argc, char **argv) {
  cl::ParseCommandLineOptions(argc, argv);
  string ErrorMessage;
  OwningPtr<CompilationDatabase> Compilations(
    CompilationDatabase::loadFromDirectory(
      BuildPath, ErrorMessage));
  if (!Compilations)
    report_fatal_error(ErrorMessage);
  RefactoringTool Tool(*Compilations,
                       SourcePaths);
  // ... magic ...
  return Tool.run(
    newFrontendActionFactory(...));
}
```

Magic you ask?

AST Matchers: A predicate library for Clang ASTs

Code snippet:

```
e = elements.Get(42);
f = fish->Get(23);
f.Cook();
feed();
```

Code snippet:

```
e = elements.Get(42);
f = fish->Get(23);
f.Cook();
feed();
```

Matcher:

```
Call()
```

Code snippet:

```
e = elements.Get(42);
f = fish->Get(23);
f.Cook();
feed();
```

Matcher:

```
Call(
  Callee(...))
```

Code snippet:

```
e = elements.Get(42);
f = fish->Get(23);
f.Cook();
feed();
```

Matcher:

```
Call(
  Callee(Method()))
```

Code snippet:

```
e = elements.Get(42);
f = fish->Get(23);
f.Cook();
feed();
```

Matcher:

```
Call(
  Callee(Method(HasName("Get"))))
```

Code snippet:

```
e = elements.Get(42);
f = fish->Get(23);
f.Cook();
feed();
```

Matcher:

```
Call(
  Callee(Method(HasName("Get"))),
  ThisPointerType(...))
```

Code snippet:

```
e = elements.Get(42);
f = fish->Get(23);
f.Cook();
feed();
```

Matcher:

```
Call(
  Callee(Method(HasName("Get"))),
  ThisPointerType(Class(
    IsDerivedFrom("ElementsBase"))))
```

Ok, let’s use magic!

```
  // Magic:
  MatchFinder Finder;
  CallRenamer CallCallback(
    &Tool.getReplacements());
  Finder.addMatcher(
    Call(
      Callee(Method(HasName("Get"))),
        ThisPointerType(Class(
            IsDerivedFrom("ElementsBase")))),
      &CallCallback);

  return Tool.run(
    newFrontendActionFactory(&Finder));
}
```

```
class CallRenamer
    : public MatchFinder::MatchCallback {
  Replacements *Replace;

 public:
  CallRenamer(Replacements *Replace)
    : Replace(Replace) {}

  virtual void run(
    const MatchFinder::MatchResult &Result);
};
```

```
MatchFinder Finder;
Renamer Callback(&Tool.getReplacements());
Finder.addMatcher(
  Call(
    Callee(Method(HasName("Get"))),
    ThisPointerType(Class(
        IsDerivedFrom("ElementsBase")))),
    Callee(Id("member",
              MemberExpression()))),
&Callback);
```

```
void CallRenamer::run(
    const MatchFinder::MatchResult &Result) {

  const MemberExpr *M =
    Result.Nodes.getStmtAs<MemberExpr>(
      "member");

  Replace->insert(Replacement(
    *Result.SourceManager,
    CharSourceRange::getTokenRange(
      SourceRange(M->getMemberLoc()))),
    "Front"));
}
```

```
// More magic:
DeclRenamer DeclCallback(
  &Tool.getReplacements());
Finder.addMatcher(Id("method",
    Method(
      HasName("Get"),
      OfClass(
        IsDerivedFrom(
          "ElementsBase")))),
&DeclCallback);
```

```
void DeclRenamer::run(
    const MatchFinder::MatchResult &Result) {

  const CXXMethodDecl *D =
    Result.Nodes.getDeclAs<CXXMethodDecl>(
      "method");

  Replace->insert(
    Replacement(
      *Result.SourceManager,
      CharSourceRange::getTokenRange(
        SourceRange(D->getLocation())),
      "Front"));
}
```

There’s more magic!

```
Finder.addMatcher(
  ConstructorCall(
    HasDeclaration(Method(
        HasName(StringConstructor))),
    ArgumentCountIs(2),
    HasArgument(
      0,
      Id("call", Call(
          Callee(Id("member",
                    MemberExpression()))),
          Callee(Method(
              HasName(StringCStrMethod))),
          On(Id("arg", Expression()))))),
  HasArgument(
    1,
    DefaultArgument())),
&Callback);
```

Go write tools!

Source Code

Open Source branch of Clang:

http://llvm.org/svn/llvm-project/cfe/branches/tooling

Rename example code:

.../examples/rename-interface/RenameInterface.cpp

Remove string::c_str calls code:

.../tools/remove-cstr-calls/RemoveCStrCalls.cpp