Six steps from zero to IDE

@olafurpg

Watch this talk on YouTube
“The Scala IDE situation is quite possibly the worst out of all the industrial languages in existence. They’re all terrible, especially VS Code or Atom. Just use IntelliJ and hope for the best.”

— r/scala, 10 points
Let’s build an IDE!
How hard can it be?
Pretty basic features

- Work with my editor
  - Vim, Sublime Text, VS Code, Atom, Emacs, …
- Work with my build tool
  - sbt, gradle, maven, pants, bazel, mill, cbt, …
- Diagnostics
  - No spurious red squiggles!
Pretty basic features

- Goto definition
- Project sources, Scala deps, Java deps, classfiles
- Auto completions
  - Scope, auto-import, type members, extension methods
- Refactorings
  - Rename, organize imports, insert type annotation, …
override def references(
  request: TextDocumentReferencesRequest
): Task[ReferencesResult] = Task {
  ReferencesProvider.references(
    symbolIndex,
    Uri(request.params.textDocument.uri),
    request.params.position,
    request.params
  )

  p params : langserver.messages.ReferenceParams

  override m productArity (x$1: Int): Any
  override m productElement (x$1: Int): Any

  override m productFix

  SignatureHelpProvider.empty

Rest of the talk

- Language Server Protocol
- “Zoo” of Scala language servers
- Build Server Protocol
- Conclusion
<table>
<thead>
<tr>
<th></th>
<th>Scala</th>
<th>Java</th>
<th>Rust</th>
<th>Python</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vim</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sublime</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VS Code</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Atom</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Emacs</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IntelliJ</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Language Server Protocol

<table>
<thead>
<tr>
<th></th>
<th>Vim</th>
<th>Sublime</th>
<th>VS Code</th>
<th>Atom</th>
<th>Emacs</th>
<th>IntelliJ</th>
<th>Scala</th>
<th>Java</th>
<th>Rust</th>
<th>Python</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
</tr>
</tbody>
</table>
JSON-RPC

```json
-->
{
    "jsonrpc": "2.0",
    "method": "subtract",
    "params": [42, 23],
    "id": 1
}
<--
{
    "jsonrpc": "2.0",
    "result": 19,
    "id": 1
}
-->
{
    "jsonrpc": "2.0",
    "method": "subtract",
    "params": [23, 42],
    "id": 2
}
<--
{
    "jsonrpc": "2.0",
    "result": -19,
    "id": 2
}
```
Wait, JSON? On every keystroke?
Xi

an editor for the next 20 years with uncompromising performance
Frame trace breakdown (scroll at 165Hz)

- JSON parse
- xi-core
- OpenGL draw
- Shaping + OpenGL draw
- NSApplication.shared.orderedDocuments
- NSDictionary as [String : Any]
- EditView render
- blockCast
- handleDraw
- applyUpdate
- dfvi
- update
LGTM, where do I sign up?
How to build an IDE

1. Configure LSP

2. Implement the rest of the IDE
Language Features
package a

object A extends App {
  def foo: String = {
    "foo"
  }
}

No problems have been detected in the workspace so far.
textDocument/publishDiagnostics
textDocument/definition
textDocument/definition
 SemanticDB

Symbol: spire.math.Complex#real().
Range: 10:18-10:22
Role: REFERENCE
package scala.meta.languageserver

import scala.meta._

object Main extends App {
  val code = "object Code".parse[Source]
}
textDocument/completions
import com.typesafe.scalalogging.LazyLogging
import org.langmeta.inputs.Input
import org.langmeta.internal.semanticdb.schema.Database
import org.langmeta.io.AbsolutePath
import com.sun.crypto.provider.AESKeyGenerator
import scala.concurrent.{Future, Await, Awaitable => Danger}

object Semanticdbs extends LazyLogging {
  Future.successful(1)

  def toSemanticdb(
    input: Input.VirtualFile,
    scalacProvider: ScalacProvider
  ): Option[semanticdb.Database] =
textDocument/codeActions
The Zoo
<table>
<thead>
<tr>
<th></th>
<th>ENSIME</th>
<th>Dotty</th>
<th>sbt</th>
<th>Metals</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Navigation</strong></td>
<td>Graphpocolypse</td>
<td>TASTY*</td>
<td>zinc analyses*</td>
<td>SemanticDB</td>
</tr>
<tr>
<td><strong>Diagnostics</strong></td>
<td>scalac</td>
<td>dotc</td>
<td>sbt server</td>
<td>sbt server/scalac</td>
</tr>
<tr>
<td><strong>Completions</strong></td>
<td>scalac</td>
<td>dotc</td>
<td></td>
<td>scalac</td>
</tr>
<tr>
<td><strong>Refactoring</strong></td>
<td>scala-refactoring</td>
<td></td>
<td></td>
<td>scalafix</td>
</tr>
</tbody>
</table>

*partial support*
Limitations
Pretty Basic Features

- Work with my build tool
  - sbt, gradle, maven, pants, bazel, mill, cbt, ...
<table>
<thead>
<tr>
<th></th>
<th>IntelliJ</th>
<th>ENSIME</th>
<th>Dotty</th>
<th>Metals</th>
</tr>
</thead>
<tbody>
<tr>
<td>sbt</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>maven</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>gradle</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>pants</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>bazel</td>
<td>✔️</td>
<td></td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>mill</td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bloop</td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Build Server Protocol

<table>
<thead>
<tr>
<th>sbt</th>
<th>IntelliJ</th>
</tr>
</thead>
<tbody>
<tr>
<td>maven</td>
<td>ENSIME</td>
</tr>
<tr>
<td>gradle</td>
<td>Dotty</td>
</tr>
<tr>
<td>pants</td>
<td>Metals</td>
</tr>
<tr>
<td>bazel</td>
<td></td>
</tr>
<tr>
<td>mill</td>
<td></td>
</tr>
<tr>
<td>bloop</td>
<td>✓</td>
</tr>
</tbody>
</table>
Build Server Protocol and fresh IDEAs

— @ebenwert, @jvican at ScalaSphere April 2018
Conclusion

• Work with my editor
• Work with my build tool
• Diagnostics
• Navigation
• Completions
• Refactoring
Conclusion

- Work with my editor → LSP!
- Work with my build tool
- Diagnostics
- Navigation
- Completions
- Refactoring
Conclusion

• Work with my editor → LSP!

• Work with my build tool → BSP!

• Diagnostics → BSP+LSP!

• Navigation

• Completions

• Refactoring
Conclusion

- Work with my editor → LSP!
- Work with my build tool → BSP!
- Diagnostics → BSP+LSP!
- Navigation → SemanticDB!
- Completions
- Refactoring
Conclusion

• Work with my editor → LSP!

• Work with my build tool → BSP!

• Diagnostics → BSP+LSP!

• Navigation → SemanticDB!

• Completions → ???!

• Refactoring
Conclusion

- Work with my editor → LSP!
- Work with my build tool → BSP!
- Diagnostics → BSP+LSP!
- Navigation → SemanticDB!
- Completions → ???!
- Refactoring → Scalafix!
Thank You