HIE files in GHC 8.8

Zubin Duggal Matthew Pickering

Introduction

.hie files - why do they exist?

- Tooling like haddock and HIE need to recompile source to get access to semantic information
- Semantic information is completely unavailable for code that doesn't compile.
- No reasonable way to get information about code in dependencies
- Setting up a GHC session that can load what you want it to load is hard

Contents of .hie files

- The original source of the file we compiled
- An interval tree, where each interval corresponds to a span in the source

```
Node { nodeInfo :: NodeInfo type
     , nodeSpan :: RealSrcSpan
     , nodeChildren :: [HieAST type]
     }
```

Information captured in nodes

- The type(s) assigned by GHC to this Node, if any
- The name of the GHC AST constructors that this Node corresponds to
- The identifiers that occur at this span
- Information about the identifiers, like:
  - Are they names or modules
  - Their type, if any
  - The context in which they occur: Are they being defined or used, imported, exported, declared etc..
  - If they are being defined, the full span of the definition, and an approximated scope over which their definition can be used.
- Types (stored in a hash consed representation)

Generating .hie files

- Pass the option -fwrite-ide-info to ghc to generate them next to .hi/.o files
- Can control path with -hiedir
- Cabal/Stack and other build tools need to learn to manage these.

Consuming .hie files

```
readHieFile :: NameCache -> FilePath
            -> IO (HieFileResult, NameCache)
-- ^ Takes and returns a NameCache so it can
-- play nice with existing GHC sessions
generateReferencesMap
  :: HieASTs a
  -> M.Map Identifier [(Span, IdentifierDetails a)]
selectSmallestContaining
  :: Span -> HieAST a -> Maybe (HieAST a)
selectLargestContainedBy
  :: Span -> HieAST a -> Maybe (HieAST a)
```

Tools that make use of .hie files

Haddock hyperlinked-source

- Now with type information!
- Can be extended to support richer code navigation

Demo

HieDb

- Index .hie files to get reference information for your whole source tree!
- Extremely fast searching and indexing thanks to SQLite
- Supports many queries on .hie files and suitable for a lightweight IDE interface

Demo

hie-lsp

- An LSP server that uses .hie files and HieDb
- Fast, low resource usage
- Type information on hover
- References
- Go to definition
- Browse all symbols
- All of this works with your entire dependency tree
- Testing ground for features that will make their way into haskell-ide-engine

Demo

hie-lsif

- Generate LSIF files from .hie files
- To be used for things like Github's online code navigation and review for PRs

Language Server Index Format

- Language agnostic, project wide cache/database for LSP requests
- Contains cached responses to a subset of LSP requests(hover, definition etc.) for mostly static files like dependencies.

Links

- hiedb: https://github.com/wz1000/HieDb
- hie-lsp: https://github.com/wz1000/hie-lsp
- hie-lsif: https://github.com/mpickering/hie-lsif
- More information on .hie files:
  https://gitlab.haskell.org/ghc/ghc/wikis/hie-files