Project Kepler

Eugene Burmako

EPFL, LAMP

17 January 2012

Kepler

How to empower developers so that they can extend the compiler, but stay sane and not mess it up?

Why?

Compile-time programming enables:

- language virtualization
- program reification
- self-optimization
- algorithmic program construction

How?

Macros make extending the compiler possible.

Quasiquotes make extending the compiler bearable.

But!

Q: Scala has enough advanced features for its creator to think about introducing feature switches. Why bother?

But!

A: Macros advance us in several areas that are hot for the community:

- code lifting for better DSLs,
- domain-specific optimization for high performance,
- (speculation) type-level computations for principled type hackery.

Zen

```
class Queryable[T, Repr](query: Query) {
  macro def filter(p: T => Boolean): Repr = scala"""
    val b = $newBuilder
    b.query = Filter($query, ${reify(p)})
    b.result
  """
}
```

Now

Prototypes: http://github.com/scalamacros/kepler

- macro defs
- quasiquotes
- splicing
- pattern matching

Now

Documentation: http://scalamacros.org

- use cases
- talks
- walkthroughs
- alpha proposals

Use

- Slick language integrated connector kit
- Lenses
- Shapeless
- Domain-specific inlining

Next?

- SIP (Scala 2.10)
- Stabilization
- Macro types
- Macro annotations

Thanks!

eugene.burmako@epfl.ch