decltrait

Tian Liao

January 2, 2024

Document number:
Date: January 2, 2024
Audience:
Authors: Tian Liao, Mingxin Wang
Reply-to: Tian Liao <tilia@microsoft.com>

1 Introduction

2 Proposal

2.1 Language feature

decltrait-specifier:
decltrait (*class-name*, *id-expression*)
where, *id-expression* requires to be evaluated to a pointer.
decltrait with the same *class-name*s deduce reference to a unique unamed trait-type.

Example:

```
struct Drawable { void print(); };
struct Rectangle { void print() { std::println("Rectangle."); } };
struct Circle { void print() { std::println("Circle."); } };

void foo() {
  Rectangle rectangle;
  auto& drawable = decltrait(Drawable, &rectangle);
  drawable.print(); // prints "Rectangle.".

  Circle circle;
  decltrait(Drawable, &circle).print(); // prints "Circle.".

  static_assert(std::is_lvalue_reference_v<
    decltrait(Drawable, &rectangle)>); // true.
  static_assert(!std::is_same_v<
    Drawable,
    std::remove_reference_t<
      decltype(
        decltrait(Drawable, &rectangle))>>); // true (not the same).
  static_assert(std::is_same_v<
    decltype(decltrait(Drawable, &rectangle)),
    decltype(decltrait(Drawable, &circle))>); // true.
}
```

In the sample code, `decltrait(Drawable, &rectangle)` asking compiler to do some magic like:

```
struct __UNIQUE_TRAIT_Drawable {
  virtual void print() = 0;
};

struct __UNIQUE_DISPATCH_Drawable_Rectangle
  : __UNIQUE_TRAIT_Drawable {
  void print() override {
    reinterpret_cast<Rectangle*>(this)->print();
```

```
  }
};

__UNIQUE_TRAIT_Drawable& __UNIQUE_DECLTRAIT_Drawable(Rectangle* src) {
  return
    *reinterpret_cast<__UNIQUE_DISPATCH_Drawable_Rectangle*>(src);
}

void foo(){
  Rectangle rectangle;
  decltrait(Drawable, &rectangle).print(); // is equvalent to below:
  __UNIQUE_DECLTRAIT_Drawable(&rectangle).print();
}
```

3 Motivation