

Crate | derive_more

■ derive_more

- ◆ Automatic implementation of Rust standard library traits for structs and enums
 - Arithmetic: Add, Sub, Mul, ...
 - From/Into, TryFrom/TryInto
 - IntoIterator
 - Display
 - Indexing
- ◆ Constructors

■ Display - Struct Example

```
use derive_more::Display;
```

```
#[derive(Display)]
```

```
#[display(fmt = "Item: {}, Quantity: {}", item, qty)]
```

```
struct Order {  
    item: String,  
    qty: usize,  
}
```

■ Display – Struct Output

```
let order = Order {  
    item: String::from("Apple"),  
    qty: 3,  
};
```

```
println!("{}", order);
```

Item: Apple, Quantity: 3

■ Display – Enum Example

```
use derive_more::Display;
```

```
#[derive(Display)]
```

```
enum GroceryItem {
```

```
    #[display(fmt = "Bread slices: {}", _0)]
```

```
    Bread(usize),
```

```
    #[display(fmt = "Fruit")]
```

```
    Fruit,
```

```
    #[display(fmt = "Ounces of meat: {}", _0)]
```

```
    Meat(usize),
```

```
}
```

■ Display – Enum Output

```
let bread = GroceryItem::Bread(10);  
let fruit = GroceryItem::Fruit;  
let meat = GroceryItem::Meat(6);  
println!("{}", bread);  
println!("{}", fruit);  
println!("{}", meat);
```

```
Bread slices: 10  
Fruit  
Ounces of meat: 6
```

■ Display – Enum Example 2

```
use derive_more::Display;

#[derive(Display)]
#[display(fmt = "Grocery item: {}")]
enum GroceryItem {
    #[display(fmt = "{} bread slices", _0)]
    Bread(usize),
    #[display(fmt = "fruit")]
    Fruit,
    #[display(fmt = "{} ounces of meat", _0)]
    Meat(usize),
}
```

■ Display – Enum Output 2

```
let bread = GroceryItem::Bread(10);  
let fruit = GroceryItem::Fruit;  
let meat = GroceryItem::Meat(6);  
println!("{}", bread);  
println!("{}", fruit);  
println!("{}", meat);
```

```
Grocery item: 10 bread slices  
Grocery item: fruit  
Grocery item: 6 ounces of meat
```


■ From – Tuple Struct

```
use derive_more::From;
```

```
#[derive(From)]
```

```
struct UserId(i32);
```

```
let user_id: UserId = 15.into();
```

```
let user_id = UserId::from(15);
```

■ From – Struct

```
use derive_more::From;
```

```
#[derive(From)]  
struct Coordinate {  
    x: i32,  
    y: i32,  
}
```

```
let coord: Coordinate = (3, 2).into();  
let coord = Coordinate::from((3, 2));
```

■ From – Enum

```
use derive_more::From;
```

```
#[derive(From)]  
enum Material {  
    Flooring(usize, usize),  
    Wood(usize),  
}
```

```
let floor: Material = (5, 5).into();  
let floor = Material::from((5, 5));  
let wood: Material = 10.into();  
let wood = Material::from(10);
```

IntoIterator

```
use derive_more::IntoIterator;
```

```
#[derive(IntoIterator)]
```

```
struct Passengers {
```

```
    #[into_iterator(owned, ref, ref_mut)]
```

```
    names: Vec<String>,
```

```
}
```

```
let passengers = Passengers { names: vec![] };
```

```
for passenger in &passengers { ...
```

```
}
```

■ Arithmetic

```
use derive_more::{Add, From, Sub};
```

```
#[derive(Add, Debug, From, Sub)]
```

```
struct Point {
```

```
    x: i32,
```

```
    y: i32,
```

```
}
```

```
let a: Point = (1, 1).into();
```

```
let b: Point = (2, 3).into();
```

```
let c = a + b;
```

```
println!("{:?}", c);
```

Point { x: 3, y: 4 }

■ Recap

- ◆ *derive_more* provides derive macros to implement common stdlib traits:
 - *Display* – `{}` formatting token
 - *From* – *From* & *Into*
 - *IntoIterator* – *IntoIterator* for:
 - ▶ Move, Borrow, Mutable Borrow
 - *Add*, *Sub*, ... – Corresponding stdlib trait
- ◆ Full list of derives is available as part of the *derive_more* docs