

Ruby Mapping for Structures

A Slice [structure](#) maps to a Ruby class with the [same name](#). For each Slice data member, the Ruby class contains a corresponding instance variable as well as accessors to read and write its value. For example, here is our [Employee](#) structure once more:

```
Slice

struct Employee {
    long number;
    string firstName;
    string lastName;
};
```

The Ruby mapping generates the following definition for this structure:

```
Ruby

class Employee
    def initialize(number=0, firstName='', lastName='')
        @number = number
        @firstName = firstName
        @lastName = lastName
    end

    def hash
        # ...
    end

    def ==
        # ...
    end

    def inspect
        # ...
    end

    attr_accessor :number, :firstName, :lastName
end
```

The constructor initializes each of the instance variables to a default value appropriate for its type. You can also declare different [default values](#) for members of primitive and enumerated types.

The compiler generates a definition for the `hash` method, which allows instances to be used as keys in a hash collection. The `hash` method returns a hash value for the structure based on the value of its data members.

The `==` method returns true if all members of two structures are (recursively) equal.

The `inspect` method returns a string representation of the structure.

See Also

- [Structures](#)
- [Ruby Mapping for Identifiers](#)
- [Ruby Mapping for Modules](#)
- [Ruby Mapping for Built-In Types](#)
- [Ruby Mapping for Enumerations](#)
- [Ruby Mapping for Sequences](#)
- [Ruby Mapping for Dictionaries](#)
- [Ruby Mapping for Constants](#)
- [Ruby Mapping for Exceptions](#)
- [Ruby Mapping for Interfaces](#)
- [Ruby Mapping for Operations](#)