🔷 Sistema de Tipado Ruby 3

RBS, TypeProf y Steep - Guía Completa

Ruby 3.3.8 | Agosto 2025

1. Introducción al Sistema de Tipado Ruby 3

Filosofía del Tipado en Ruby 3

Ruby 3 introduce un sistema de tipado **gradual y opcional** que permite agregar información de tipos sin cambiar la naturaleza dinámica del lenguaje. El sistema está compuesto por tres herramientas principales:

- **RBS:** Lenguaje de definición de tipos
- **TypeProf:** Inferencia automática de tipos
- **Steep:** Verificador de tipos

1.1 Ventajas del Sistema de Tipado

Sin Tipado

```
# user.rb class User def initialize(name, age) @name = name @age = age end def adult? @age >= 18 end def greet(message) "#{message}, #{@name}!" end end # Posibles errores en runtime user = User.new("John", "30") # age como string user.adult? # Error: no se puede comparar user.greet(123) # Funciona pero no es semánticamente correcto
```

Con Tipado RBS

```
# user.rbs class User @name: String @age: Integer def initialize: (String, Integer) -> void def adult?: -> bool def greet: (String) -> String end # user.rb (sin cambios) class User def initialize(name, age) @name = name @age = age end def adult? @age >= 18 end def greet(message) "#{message}, #{@name}!" end end # Verificación con Steep detecta errores antes de runtime
```

2. RBS - Ruby Signature

¿Qué es RBS?

RBS es un lenguaje para describir la estructura de programas Ruby. Define tipos de métodos, variables de instancia, constantes y más, sin modificar el código Ruby original.

2.1 Sintaxis Básica de RBS

Tipos Primitivos

```
# tipos_primitivos.rbs # Tipos básicos String Integer Float bool # true o false nil void # sin valor de retorno # Tipos literales "literal_string" 42 true false # Tipos de colección Array[String] # Array de strings Hash[String, Integer] # Hash con keys String y values Integer Array[Integer | String] # Array que puede contener Integer o String
```

Definición de Clases

```
# calculator.rbs class Calculator # Variables de instancia @memory: Float @history: Array[String] # Métodos de instancia def initialize: () -> void def add: (Float, Float) -> Float def subtract: (Float, Float) -> Float def multiply: (Float, Float) -> Float def divide: (Float, Float) -> Float def clear_memory: () -> void def recall: () -> Float def store: (Float) -> void # Métodos de clase def self.scientific_mode: () -> Calculator # Constantes PI: Float E: Float end
```

2.2 Tipos Avanzados

```
# tipos_avanzados.rbs # Union Types (tipos unión) def process_id: (Integer | String) -> String # Optional Types (tipos opcionales) def find_user: (Integer) -> User? # Generics (tipos genéricos) class Container[T] @value: T def initialize: (T) -> void def get: () -> T def set: (T) -> void end # Proc types def map_values: [T, U] (Array[T], ^(T) -> U) -> Array[U] # Record types (estructuras) type UserRecord = { name: String, age: Integer, email: String? } # Alias de tipos type ID = Integer | String type UserCollection = Array[User] | Hash[ID, User] # Interface types interface _Serializable def serialize: () -> String def deserialize: (String) -> self end
```

2.3 Módulos y Mixins

```
# authentication.rbs module Authentication # Métodos de módulo def self.hash_password: (String) -> String def self.verify_password: (String, String) -> bool # Métodos que se incluyen cuando se hace include def authenticate: (String, String) -> bool def logout: () -> void def current_user: () -> User? end # Clase que incluye el módulo class UserController include Authentication @current_user: User? def login: (String, String) -> bool def dashboard: () -> String end
```

3. TypeProf - Inferencia de Tipos

TypeProf: El Analizador Inteligente

TypeProf analiza código Ruby existente y genera automáticamente definiciones RBS, infiriendo tipos basándose en el uso real del código.

3.1 Instalación y Uso de TypeProf

```
# Instalación gem install typeprof # Uso básico - analizar archivo individual typeprof user.rb # Analizar múltiples archivos typeprof app/models/*.rb # Generar archivo RBS typeprof user.rb > sig/user.rbs # Análisis con configuración typeprof --config=typeprof.yaml
```

3.2 Ejemplo de Inferencia Automática

Código Ruby Original

```
# shopping_cart.rb class ShoppingCart def initialize @items = [] @total = 0.0 end def add_item(name, price, quantity = 1) item = { name: name, price: price, quantity: quantity, subtotal: price * quantity } @items << item @total += item[:subtotal] end def remove_item(index) return unless @items[index] removed = @items.delete_at(index) @total -= removed[:subtotal] removed end def calculate_discount(percentage) discount = @total * (percentage / 100.0) @total - discount end def checkout { items: @items, total: @total, timestamp: Time.now } end end # Uso del código para ayudar a la inferencia cart = ShoppingCart.new cart.add_item("Laptop", 999.99, 1) cart.add_item("Mouse", 29.99, 2) cart.remove_item(0) final_total = cart.calculate_discount(10.0) receipt = cart.checkout
```

RBS Generado por TypeProf

```
# shopping_cart.rbs (generado automáticamente) class ShoppingCart @items: Array[Hash[Symbol, (String | Float | Integer)]] @total: Float def initialize: () -> void def add_item: (String, Float, ?Integer) -> Float def remove_item: (Integer) -> (Hash[Symbol, (String | Float | Integer)] | nil) def calculate_discount: (Float) -> Float def checkout: () -> Hash[Symbol, (Array[Hash[Symbol, (String | Float | Integer)]] | Float | Time)] end
```

3.3 Configuración de TypeProf

```
# typeprof.yaml target_files: - "app/models/**/*.rb" - "lib/**/*.rb" output_dir: "sig" options: # Incluir métodos privados en el análisis show_untyped: false # Nivel de detalle en la inferencia verbose: 1 # Ignorar ciertos patrones ignore: - "test/**/*.rb" - "spec/**/*.rb" # Configuraciones específicas por archivo per_file_options: "app/models/user.rb": show_untyped: true libraries: - name: "rails" version: "7.0"
```

4. Steep - Verificación de Tipos

Steep: El Verificador de Tipos

Steep utiliza las definiciones RBS para verificar la corrección de tipos en tiempo de desarrollo, detectando errores antes de que lleguen a producción.

4.1 Instalación y Configuración

```
# Instalación gem install steep # Inicializar proyecto con Steep steep init # Esto crea Steepfile con configuración básica target :main do signature "sig" check "app" library "pathname" end
```

4.2 Verificación de Tipos

```
# Verificar todos los archivos steep check # Verificar archivo específico steep check app/models/user.rb # Modo watch (verificación continua) steep watch # Verificar con más detalle steep check --verbose # Generar reporte en formato JSON steep check --format=json > type_errors.json
```

4.3 Ejemplos de Detección de Errores

```
# user.rb class User def initialize(name, age) @name = name @age = age end def birthday! @age += 1 end def adult? @age >= 18 end end # user.rbs class User @name: String @age: Integer def initialize: (String, Integer) -> void def birthday!: () -> void def adult?: () -> bool end # test_user.rb - Errores que Steep detecta user = User.new("John", "30") # Error: Expected Integer, got String user.birthday! result = user.adult? puts result + 1 # Error: No method + for bool # Steep output: # test_user.rb:1:20: [error] Type mismatch: # expected: Integer # actual: String # test_user.rb:4:5: [error] Type mismatch: # expected: No method `+` for `bool`
```

5. Integración Práctica

5.1 Flujo de Trabajo Recomendado

> **Proceso de Adopción Gradual**
>
> 1. **Análisis inicial:** Usar TypeProf para generar RBS básico
> 2. **Refinamiento:** Editar manualmente los archivos RBS
> 3. **Verificación:** Usar Steep para detectar inconsistencias
> 4. **Iteración:** Corregir código o tipos según sea necesario

```
# Makefile para automatizar el proceso .PHONY: typecheck generate-types check-types # Generar tipos automáticamente generate-types: typeprof app/models/*.rb > sig/models.rbs typeprof app/controllers/*.rb > sig/controllers.rbs typeprof lib/*.rb > sig/lib.rbs # Verificar tipos check-types: steep check # Verificar con reporte detallado typecheck: generate-types steep check --verbose # Verificación continua durante desarrollo watch: steep watch # Limpiar archivos de tipos generados clean-types: rm -f sig/*.rbs
```

5.2 Integración con CI/CD

```
# .github/workflows/type_check.yml name: Type Check on: [push, pull_request] jobs: typecheck: runs-on: ubuntu-latest steps: - uses: actions/checkout@v3 - name: Set up Ruby uses: ruby/setup-ruby@v1 with: ruby-version: 3.3.8 bundler-cache: true - name: Install type checking tools run: | gem install steep typeprof rbs - name: Generate type signatures run: | mkdir -p sig typeprof app/models/*.rb > sig/models.rbs typeprof app/controllers/*.rb > sig/controllers.rbs - name: Type check run: steep check - name: Upload type errors if: failure() uses: actions/upload-artifact@v3 with: name: type-errors path: type_errors.json
```

5.3 Configuración Avanzada

```
# Steepfile avanzado target :main do signature "sig" # Directorios a verificar check "app" check "lib" # Bibliotecas RBS incluidas library "pathname" library "logger" library "json" library "time" # Configuraciones específicas configure_code_diagnostics do |hash| # Ignorar ciertos tipos de errores hash[Steep::Diagnostic::Ruby::MethodDefinitionMissing] = :information hash[Steep::Diagnostic::Ruby::UnresolvedOverloading] = :warning end end # Target separado para tests target :test do signature "sig" check "test" check "spec" library "minitest" library "rspec" end
```

6. Casos de Uso Avanzados

6.1 APIs y Servicios Web

```
# api_service.rbs class ApiService @base_url: String @headers: Hash[String, String] def initialize: (String, ?Hash[String, String]) -> void # Métodos HTTP tipados def get: [T] (String, **untyped) -> ApiResponse[T] def post: [T] (String, Hash[String, untyped], **untyped) -> ApiResponse[T] def put: [T] (String, Hash[String, untyped], **untyped) -> ApiResponse[T] def delete: (String, **untyped) -> ApiResponse[nil] private def make_request: (String, String, ?Hash[String, untyped]) -> Net::HTTPResponse def parse_response: [T] (Net::HTTPResponse) -> ApiResponse[T] end # Respuesta tipada class ApiResponse[T] @status: Integer @data: T @errors: Array[String] def initialize: (Integer, T, Array[String]) -> void def success?: () -> bool def error?: () -> bool def data: () -> T def errors: () -> Array[String] end # Modelos de datos type UserData = { id: Integer, name: String, email: String, created_at: String } type ProductData = { id: Integer, name: String, price: Float, category: String, in_stock: bool }
```

6.2 Metaprogramación Tipada

```
# dynamic_model.rbs class DynamicModel @attributes: Hash[Symbol, untyped] def initialize: (Hash[Symbol, untyped]) -> void # Métodos dinámicos def method_missing: (Symbol, *untyped) -> untyped def respond_to_missing?: (Symbol, bool) -> bool # Define_method dinámico def self.attr_typed: [T] (Symbol, Class) -> void # Validaciones tipadas def validate: () -> Array[String] def valid?: () -> bool # Serialización def to_h: () -> Hash[Symbol, untyped] def to_json: () -> String # Factory methods tipados def self.from_hash: (Hash[Symbol, untyped]) -> instance def self.from_json: (String) -> instance end # Uso específico con tipos class User < DynamicModel # Definir atributos con tipos attr_typed :name, String attr_typed :age, Integer attr_typed :email, String # Métodos específicos tipados def adult?: () -> bool def full_profile: () -> Hash[Symbol, (String | Integer | bool)] end
```

7. Mejores Prácticas

Recomendaciones para Adopción Exitosa

- **Comienza gradualmente:** Empieza por módulos críticos
- **Usa TypeProf como base:** Genera tipos automáticamente primero
- **Refina manualmente:** Los tipos generados son un punto de partida
- **Integra en CI/CD:** Haz la verificación parte del proceso
- **Documenta decisiones:** Explica por qué ciertos tipos se eligieron
- **Mantén simplicidad:** No sobre-especifiques tipos complejos al inicio

Limitaciones y Consideraciones

- **Metaprogramación:** Difícil de tipar dinámicamente
- **Overhead:** Requiere mantenimiento adicional
- **Curva de aprendizaje:** Sintaxis RBS requiere tiempo
- **Herramientas en evolución:** Ecosistema aún madurando

7.1 Ejemplo Completo de Proyecto Tipado

```
# Estructura de proyecto con tipos my_app/ ├── app/ │ ├── models/ │ │ ├── user.rb │ │ └── product.rb │ └── services/ │ └── payment_service.rb ├── sig/ │ ├── models/ │ │ ├── user.rbs │ │ └── product.rbs │ └── services/ │ └── payment_service.rbs ├── Steepfile ├── typeprof.yaml └── Gemfile # Gemfile gem 'rbs' gem 'typeprof' gem 'steep' # Comandos para mantener tipos actualizados rake typecheck:generate # Genera RBS con TypeProf rake typecheck:verify # Verifica con Steep rake typecheck:clean # Limpia archivos temporales
```