REST APIs con plumber: : GUÍA RÁPIDA

Introducción to REST APIs

APIs web usan **HTTP** para comunicarse entre **cliente** y **servidor**.

HTTP

HTTP se basa en una solicitud y una respuesta. Un cliente realiza una solicitud a un servidor, que gestiona la solicitud y proporciona una respuesta. Las solicitudes y respuestas son textos con un formato especial que contiene detalles y datos sobre el intercambio entre el cliente y el servidor.

SOLICITUD

Método HTTP · Ruta · Versión HTTP · Encabezados · Cuerpo del mensaje

```
curl -v "http://httpbin.org/get"

#> GET /get HTTP/1.1
#> Host: httpbin.org
#> User-Agent: curl/7.55.1
#> Accept: */*
#
# Request Body
```

RESPUESTA

Versión HTTP · Código de estado · Frase de motivo · Encabezados · Cuerpo del mensaje

```
#< HTTP/1.1 200 OK
#< Connection: keep-alive
#< Date: Thu, 02 Aug 2018 18:22:22 GMT
#
# Response Body
```

Plumber: Cree APIs con R

Plumber usa comentarios especiales para convertir cualquier código R arbitrario en puntos de conexión de API. En el ejemplo siguiente se define una función que toma el argumento msg y lo devuelve incrustado en texto adicional.

Los comentarios de plumber comienzan con #* · Los decoradores @ definen las características de la API · Método HTTP · /<path> se utiliza para definir la ubicación del punto de conexión

```
library(plumber)

#* @apiTitle Plumber Example API

#* Echo back the input
#* @param msg The message to echo
#* @get /echo
function(msg = "") {
  list(
    msg = paste0(
      "The message is: '", msg, "'")
  )
}
```

Plumber canalización

Los puntos de conexión de plumber contienen código R que se ejecuta en respuesta a una solicitud HTTP. Las solicitudes entrantes pasan a través de un conjunto de mecanismos antes de que se devuelva una respuesta al cliente.

FILTROS

Los filtros pueden reenviar solicitudes (después de mutarlas potencialmente), generar errores o devolver una respuesta sin reenviar la solicitud. Los filtros se definen de forma similar a los puntos de conexión mediante la etiqueta `@filter [name]`. De forma predeterminada, los filtros se aplican a todos los puntos de conexión. Los puntos de conexión pueden optar por no participar en los filtros mediante la etiqueta `@preempt`.

ANALIZADOR

Los analizadores determinan cómo Plumber analiza el cuerpo de la solicitud entrante. De forma predeterminada, Plumber analiza el cuerpo de la solicitud como notación de objetos JavaScript (JSON). Otros analizadores, incluidos los analizadores personalizados, se identifican mediante la etiqueta `@parser [parser name]`. Todos los analizadores registrados se pueden ver con `registered_parsers()`.

PUNTOS DE CONEXIÓN

Los puntos de conexión definen el código de R que se ejecuta en respuesta a las solicitudes entrantes. Estos puntos de conexión corresponden a métodos HTTP y responden a las solicitudes entrantes que coinciden con el método definido.

MÉTODOS

- `@get` - request a resource
- `@post` - send data in body
- `@put` - store / update data
- `@delete` - delete resource
- `@head` - no request body
- `@options` - describe options
- `@patch` - partial changes
- `@use` - use all methods

SERIALIZADOR

Los serializadores determinan cómo Plumber devuelve los resultados al cliente. De forma predeterminada, Plumber serializa el objeto de R devuelto en la notación de objetos JavaScript (JSON). Otros serializadores, incluidos los serializadores personalizados, se identifican mediante la etiqueta `@serializer [serializer name]`. Todos los serializadores registrados se pueden ver con `registered_serializers()`.

Identificar como filtro · Nombre del filtro · Adelante · Descripción del punto de conexión · Analizador · Método HTTP · Salida del registro · Ruta de acceso del punto de conexión · Serializer

```
library(plumber)

#* @filter log
function(req, res) {
  print(req$HTTP_USER_AGENT)
  forward()
}

#* Convert request body to uppercase
#* @preempt log
#* @parser json
#* @post /uppercase
#* @serializer json
function(req, res) {
  toupper(req$body)
}
```

Ejecución de las API de plumnber

Plumber APIs can be run programmatically from within an R session.

Ruta de acceso a la definición de API · Especificar el puerto de API

```
library(plumber)

plumb("plumber.R") %>%
  pr_run(port = 5762)
```

This runs the API on the host machine supported by the current R session.

INTEGRACIÓN DE IDE

Documentación

Las API de fontanero generan automáticamente un archivo de especificación de OpenAPI. Este archivo de especificación se puede interpretar para generar una interfaz de usuario dinámica para la API. La interfaz predeterminada se genera a través de Swagger.

Interactuar con la API

Una vez que la API se está ejecutando, se puede interactuar con ella mediante cualquier cliente HTTP. Tenga en cuenta que el uso de `httr` requiere el uso de una sesión de R independiente de la que sirve a la API.

```
(resp <- httr::GET("localhost:5762/echo?msg=Hello"))
#> Response [http://localhost:5762/echo?msg=Hello]
#>   Date: 2018-08-07 20:06
#>   Status: 200
#>   Content-Type: application/json
#>   Size: 35 B
httr::content(resp, as = "text")
#> [1] "{\"msg\":[\"The message is: 'Hello'\"]}"
```

Plumber programático

Plumber ordenado

Plumber es excepcionalmente personalizable. Además de usar comentarios especiales para crear API, las API se pueden crear de forma totalmente programática. Esto expone características y funcionalidades adicionales. Plumber tiene una interfaz conveniente y "ordenada" que permite que los enrutadores API se construyan pieza por pieza. El siguiente es un ejemplo parte de archivo `plumber.R` estándar.

```
library(plumber)

#* @plumber
function(pr) {
  pr |>
    pr_get(path = "/echo",
           handler = function(msg = "") {
             list(msg = paste0(
               "The message is: '",
               msg,
               "'")
             )
           }) |>
    pr_get(path = "/plot",
           handler = function() {
             rand <- rnorm(100)
             hist(rand)
           },
           serializer = serializer_png()) |>
    pr_post(path = "/sum",
            handler = function(a, b) {
              as.numeric(a) + as.numeric(b)
            })
}
```

Usar @plumber etiqueta

Función que acepta y modifica un enrutador de plumber

Función "Tidy" para crear la API de Plumber

OpenAPI

Plumber crea automáticamente un archivo de especificación de OpenAPI basado en los comentarios de Plumber. Este archivo se puede modificar aún más usando `pr_set_api_spec()` con una función que modifique la especificación existente o una ruta a un archivo de configuración `.yaml` o `.json`.

```
library(plumber)

#* @param msg The message to echo
#* @get /echo
function(msg = "") {
  list(
    msg = paste0(
      "The message is: '", msg, "'")
  )
}

#* @plumber
function(pr) {
  pr |>
    pr_set_api_spec(function(spec) {
      spec$paths[["/echo"]]$get$summary <-
        "Echo back the input"
      spec
    })
}
```

Función que recibe y modifica la especificación existente

Devolver la especificación actualizada

De forma predeterminada, Swagger se utiliza para interpretar el archivo de especificación de OpenAPI y generar la interfaz de usuario para la API. Se pueden utilizar otros intérpretes para ajustar la apariencia de la interfaz de usuario a través de `pr_set_docs()`.

Plumber avanzado

SOLICITUD y RESPUESTA

Plumber proporciona acceso a objetos especiales de req y res que se pueden pasar a las funciones de Plumber. Estos objetos proporcionan acceso a la solicitud enviada por el cliente y a la respuesta que se enviará al cliente. Cada objeto tiene varios componentes, de los cuales los más útiles se describen a continuación:

Nombre	Ejemplo	Descripción
req		
req$pr	plumber::pr()	El enrutador de Plumber que procesa la solicitud
req$body	list(a=1)	Normalmente es lo mismo que argsBody
req$argsBody	list(a=1)	La salida del cuerpo analizado
req$argsPath	list(c=3)	Los valores de los argumentos path
req$argsQuery	list(e=5)	La salida analizada de `req$QUERY_STRING`
req$cookies	list(cook = "a")	Una lista de cookies
req$REQUEST_METHOD	"GET"	El método utilizado para la solicitud HTTP
req$PATH_INFO	"/"	La ruta de acceso de la solicitud HTTP entrante
req$HTTP_*	"HTTP_USER_AGENT"	Todos los encabezados HTTP enviados con la solicitud
req$bodyRaw	charToRaw("a=1")	El contenido raw() del cuerpo de la solicitud
res		
res$headers	list(header = "abc")	Encabezados HTTP que se incluirán en la respuesta
res$setHeader()	setHeader("foo", "bar")	Establece un encabezado HTTP
res$setCookie()	setCookie("foo", "bar")	Establece una cookie HTTP en el cliente
res$removeCookie	removeCookie("foo")	Elimina una cookie HTTP
res$body	"{\"a\":[1]}"	Salida serializada
res$status	200	El código de estado de la respuesta HTTP
res$toResponse()	toResponse()	Una lista de estados, encabezados y cuerpo

PLUMBER ASINCRÓNICO

Plumber admite la ejecución asincrónica a través del paquete de R future. Este patrón permite a Plumber procesar simultáneamente varias solicitudes.

```
library(plumber)
future::plan("multisession")

#* @get /slow
function() {
  promises::future_promise({
    slow_calc()
  })
}
```

Establecer el plan de ejecución

Cálculo lento

MONTAJE DE ENRUTADORES

Los enrutadores de plumber se pueden combinar montando enrutadores en otros enrutadores. Esto puede ser beneficioso cuando se crean enrutadores que involucran varios puntos de conexión diferentes y desea dividir cada componente en un enrutador separado. Estos enrutadores separados pueden incluso ser archivos separados cargados usando `plumb()`.

```
library(plumber)

route <- pr() |>
  pr_get("/foo", function() "foo")

#* @plumber
function(pr) {
  pr |>
    pr_mount("/bar", route)
}
```

Crear un enrutador inicial

Montar un enrutador en otro

En el ejemplo anterior, la ruta final es /bar/foo.

EJEMPLOS DE EJECUCIÓN

Algunos paquetes, como el propio paquete Plumber, pueden incluir ejemplos de API de Plumber. Las API disponibles se pueden ver mediante `available_apis()`. Estas API de ejemplo se pueden ejecutar con `plumb_api()` combinado con `pr_run()`.

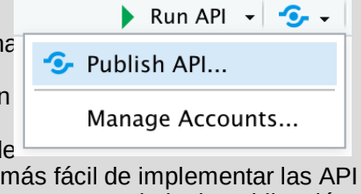
```
library(plumber)

plumb_api(package = "plumber",
          name = "01-append",
          edit = TRUE) |>
  pr_run()
```

Identifique el nombre del paquete y el nombre de la API

Opcionalmente, abra el archivo para editarlo

Ejecución de la API de ejemplo

Implementación de las API de Plumber



Una vez que las API de Plumber ha sido desarrolladas, a menudo necesitan ser desplegada en algún lugar para ser útil. Las API de plumber se pueden implementar de Varias formas. Una de las formas más fácil de implementar las API de Plumber es mediante Posit Connect, que admite la publicación con botón desde el IDE de RStudio.