Artículo
Joel Espinoza · Oct 28, 2019 Lectura de 4 min

Cómo reenviar solicitudes en un servicio REST

¡ Hola Comunidad!

Una función útil de nuestra estructura REST es la capacidad que tienen las clases de Dispatch para **identificar los prefijos de una solicitud y redireccionarlos** a otra clase de Dispatch. Este enfoque permite mejorar el orden y la lectura del código, permite mantener separadas las versiones de una interfaz fácilmente y ofrece una forma de proteger llamadas a APIs a las que solo ciertos usuarios podrán acceder.

Resumen

Para configurar un servicio REST en su instancia de Caché o IRIS Database, necesita definir una aplicación CSP dedicada y crear una **clase de Dispatch** asociada que gestione las solicitudes que entran. La clase de Dispatch se extiende desde %CSP.REST e incluirá un bloque XData que contiene su mapa URL. Esto indica al sistema qué método deberá llamar cuando reciba una solicitud en particular.

Por ejemplo:

```
XData UrlMap [ XMLNamespace = "http://www.intersystems.com/urlmap" ]
{
<Routes>
  <Route Url="/orders" Method="GET" Call="GetOrders"/>
  <Route Url="/orders" Method="POST" Call="NewOrder"/>
</Routes>
}
```

Los elementos <Route> determinan las diferentes solicitudes que administrará el servicio. Una solicitud GET para el recurso "/orders" llamará al método de clase "GetOrders". Una solicitud POST que se realiza al mismo recurso llamará, en su lugar, al método "NewOrder".

Es importante tener en cuenta que el nombre de la aplicación CSP no se considera parte del nombre del recurso solicitado en nuestro mapa URL. Analice una solicitud realizada en la dirección:

```
http://localhost:57772/csp/demo/orders
```

Si nuestra aplicación CSP se llama "/csp/demo", entonces el único segmento de la solicitud que es administrado por la clase de Dispatch es el que se encuentra después del nombre de la aplicación. En este caso es "/orders".

Reenvío de solicitudes

En lugar de llamar a un método desde dentro de la clase de Dispatch, la otra opción para su mapa URL es reenviar todas las solicitudes que coincidan con un prefijo en particular hacia una clase de Dispatch diferente.

Esto se realiza mediante el elemento <Map> que se encuentra en la sección UrlMap. El elemento contiene dos atributos, **Prefix** y **Forward**. Si la solicitud de la URL coincide con alguno de los prefijos, entonces, enviamos la solicitud a la clase de Dispatch especificada, para que se procese posteriormente.

Por ejemplo:

```
XData UrlMap [ XMLNamespace = "http://www.intersystems.com/urlmap" ]
{
<Routes>
  <Map Prefix="/shipping" Forward="Demo.Service.Shipping"/>
  <Route Url="/orders" Method="GET" Call="GetOrders"/>
  <Route Url="/orders" Method="POST" Call="NewOrder"/>
</Routes>
}
```

Una solicitud GET o POST para "/orders" será administrada directamente por esta clase. Sin embargo, las solicitudes que coincidan con el prefijo "/shipping" se redireccionarán a la clase de Dispatch Demo.Service.Shipping, que tiene su propio mapa URL:

```
XData UrlMap [ XMLNamespace = "http://www.intersystems.com/urlmap" ]
{
<Routes>
  <Route Url="/track/:id" Method="GET" Call="TrackShipment"/>
</Routes>
}
```

Fallo en el enrutamiento de la URL

Para demostrar cómo influye cada componente solicitado por la URL en el método que se llama al final, analizaremos cada elemento de una solicitud para la siguiente dirección:

```
http://localhost:57772/csp/demo/shipping/track/123
```

http://	Es el protocolo que se usó para la solicitud.
localhost:57772 (52773 en IRIS)	Es el servidor al que nos conectamos.
/csp/demo/shipping/track/123	Es el recurso que se está solicitando.
/csp/demo	Es el nombre de la aplicación CSP. Una clase de Dispatch se define para la aplicación, y enruta la solicitud hacia allí.
/shipping/track/123	Es el segmento del recurso que se enviará a la primera clase de Dispatch.
/shipping	Es el prefijo que coincide con el elemento <Map> en el mapa URL. Lo redireccionará a la clase Demo.Service.Shipping.
/track/123	Es el segmento del recurso que se envió a la segunda clase de Dispatch. Coincide con la ruta "/track/:id". Llama al método TrackShipment(123).

Resultados

- ***Control del código fuente*** — Separar su API REST en varias clases reducirá el tamaño total de cada clase, lo cual le ayudará a mantener el historial del código fuente bajo control, conciso y legible.
- ***Versiones*** — Una manera sencilla para lograr que varias versiones de una API sean compatibles simultáneamente es utilizar el reenvío. Una sola clase de Dispatch podría reenviar solicitudes que coincidan con los prefijos /v1 o /v2 a una clase de Dispatch que implemente esa versión de la API. La API

REST, que es un elemento central de Atelier, nuestro nuevo IDE, utiliza el mismo esquema de versiones.
- ***Seguridad*** — Si su API necesita tener rutas que estén restringidas para ciertos usuarios - por ejemplo, para permitir que únicamente los administradores realicen ciertos tipos de solicitudes- tiene sentido aislar estas rutas en su propia clase y después reenviar las solicitudes mediante un prefijo en particular. Si la segunda clase de Dispatch define un método **OnPreDispatch**, su código se ejecutará antes del procesamiento de cada solicitud. El servicio puede utilizar esto para comprobar los privilegios de un usuario y decidir si continúa con el procesamiento o cancela su solicitud.

#API REST #CSP #JSON #Mejores prácticas #XML #Caché #InterSystems IRIS

URL de fuente: https://es.community.intersystems.com/post/c%C3%B3mo-reenviar-solicitudes-en-un-servicio-rest