記事
Toshihiko Minamoto · 2021年6月29日 5m read

RESTサービスでのリクエストの転送

RESTフレームワークの有用な機能の1つに、
ディスパッチクラスが
リクエストのプレフィックスを識別して別のディスパッチクラスに転送するという機能があります。 URLマップをモジュール化するこの手法により、コードの可読性が向上し、インターフェースの個別のバージョンが管理しやすくなります。また、特定のユーザーのみがアクセスできるように、API呼び出しを保護する手段も得ることができます。

概要

Caché インスタンスにRESTサービスをセットアップするには、専用のSCPアプリケーションを定義して、それに関連付けられた、受信リクエストを処理する**ディスパッチクラス**を作成する必要があります。
ディスパッチクラスは、%CSP.RESTを拡張し、URLマップを含むXDataブロックを含めます。
こうすることで、システムに、特定のリクエストを受信したときにどのメソッドを呼び出すのかを指示します。

以下に、例を示します。

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

<Route> 要素は、サービスが処理するさまざまなリクエストを定義しています。
「/orders」リソースのGETリクエストは、クラスに「GetOrders」メソッドを呼び出しています。
同じリソースに対して行われるPOSTリクエストは、代わりに「NewOrder」メソッドを呼び出しています。

CSPアプリケーション名は、URLマップでリクエストされるリソース名の一部としてみなされないことに注意しておくことが重要です。 次のアドレスに対して行われるリクエストについて考えてみましょう。

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

CSPアプリケーションを「/csp/demo」とした場合、
ディスパッチクラスが処理するリクエストのセグメントは、アプリケーション名の後に続くものだけになります。
つまり、この場合は「/orders」のみということになります。

リクエストの転送

URLマップで利用できる、ディスパッチクラス内のメソッドを呼び出す以外のオプションは、特定のプレフィックスに一致するすべてのリクエストを別のディスパッチクラスに転送する方法です。

これは、UrlMapセクションの<Map> 要素を使って行います。
この要素には、**Prefix** と**Forward**の2つの属性があります。 リクエストURLがプレフィックスの1つに一致する場合、そのリクエストを特定のディスパッチクラスに送信して処理を続けることができます。

以下に、例を示します。

```
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>
}
```

「/orders」のGETリクエストまたはPOSTリクエストは、このクラスによって直接処理されますが、
「/shipping
」プレフィックスに一致するリクエストは
、独自のURLマップを持つ「/shipping」ディスパッチクラスにリダイレクトされます。

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

URLルーティングの詳細

リクエストされたURLの各コンポーネントが、最終的に呼び出されるメソッドにどのような影響があるのかを示すために、次のアドレスのリクエストを分解して説明します。

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

http://	The protocol used for the request.
localhost:57772	The server that we connect to.
/csp/demo/shipping/track/123	The resource being requested.
/csp/demo	The CSP application name. A dispatch class is defined for the application, route the request there.
/shipping/track/123	The resource segment sent to the first dispatch class.
/shipping	The prefix that matches the <Map> element in the URL map. Redirect to the Demo.Service.Shipping class.
/track/123	The resource segment sent to the second dispatch class. Matches the route "/track/:id". Call the method TrackShipment(123).

使用時のメリット

- ***ソース管理*** — REST APIを複数のクラスに分離すると、各クラスの全体的なサイズが小さくなるため、ソースの管理履歴を明確かつ読みやすく維持することができるようになります。
- ***バージョン管理***
  — 転送を使用すると、複数のバージョンのAPIを簡単に同時にサポートすることができます。 1つのディスパッチクラスは、「/v1」または「/v2」プレフィックスに一致するリクエストを、そのバージョンのAPIを実装するディスパッチクラスに転送できます。 私たちの新しいIDEであるAtelierの中心にあるREST APIでは、これと同じバージョン管理スキームが使用されています。

- ***セキュリティ***

  —

  特定の種類のリクエストを管理者だけが実行できるようにする場合など、特定のユーザーに制限されたルーティングをAPIで使用する必要がある場合、独自のクラスにルートを分離すると、特定のプレフィックスを使用して合理的にリクエストを転送することができます。
  2つ目のディスパッチクラスで**OnPreDispatch**
  メソッドが定義されている場合、各リクエストを処理する前に、そのコードが実行されます。 サービスはこれを利用して、ユーザーの権限を確認し、処理を続行するか、リクエストをキャンセルするかを決定できます。

#CSP #JSON #REST API #XML #ベストプラクティス #Caché #InterSystems IRIS

ソースURL:
https://jp.community.intersystems.com/post/rest%E3%82%B5%E3%83%BC%E3%83%93%E3%82%B9%E3%81%A7%E3%81%AE%E3%83%AA%E3%82%AF%E3%82%A8%E3%82%B9%E3%83%88%E3%81%AE%E8%BB%A2%E9%80%81