Devolución de respuestas personalizadas de Servicios Web y Servicios REST utilizando Node.js y EWD 3

Artículo
Nancy Martínez · Nov 21, 2019 Lectura de 5 min

Devolución de respuestas personalizadas de Servicios Web y Servicios REST utilizando Node.js y EWD 3

¡Hola Comunidad!

Como explicó Rob en un artículo anterior, la interfaz Node.js de Caché permite crear Servicios Web y Servicios REST utilizando el framework modular framework EWD 3.

Por defecto, estos servicios devuelven una respuesta JSON con Content-Type: application / json y el cuerpo de respuesta contiene el JSON que devuelve utilizando el método finished(), por lo que:

finished({ test: 'test response' });

devuelve

{ "test": "test response" }

con un content-type HTTP de application/json

Para otros servicios y herramientas externas como, por ejemplo, motores de generadores de informes o EDI, deberá devolver las respuestas formateadas según las especificaciones de estos servicios y también con diferentes tipos de contenido como XML. Sin embargo, debido a que, por defecto, ewd-qoper8-express hace todo el trabajo necesario para que luego tu puedas para devolver y enviar la respuesta JSON de forma predeterminada, se deberá modificar y formatear la respuesta en otro formato según las especificaciones del servicio externo antes de que sea enviada. Con el módulo ewd-qoper8-express, ahora es posible modificar la respuesta HTTP según sus necesidades.

Como se describe en la publicación anterior de Rob cómo crear servicios, usted sabe que el módulo Express le proporciona un patrón muy claro para definir cada servicio en su archivo de inicio ewd-xpress.js usando:

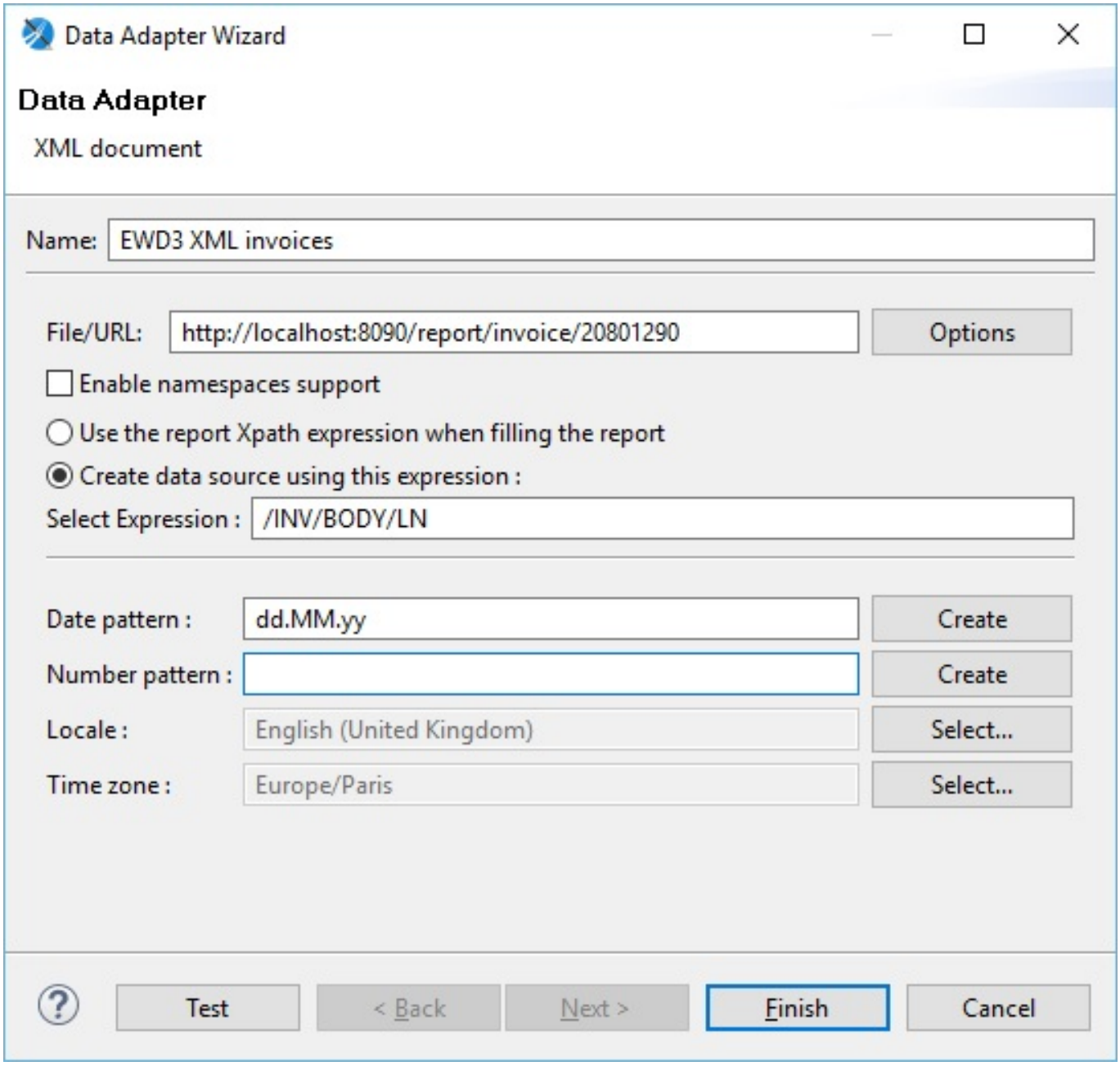
Como se describe en el artículo de Rob sobre cómo crear servicios, el módulo Express proporciona un patrón muy claro para definir cada servicio en su archivo de inicio ewd-xpress.js usando:

```
var config = {
  managementPassword: 'keepThisSecret!',
  serverName: 'My EWD 3 Server',
  port: 8080,
  poolSize: 1,
  database: {
    type: 'cache'
  }
};
var ewdXpress = require('ewd-xpress').master;
var xp = ewdXpress.intercept();
xp.app.use('/test', xp.qx.router());
```

```
xp.app.use('/report', xp.qx.router());
ewdXpress.start(config);
```

Esto devolverá por defecto una respuesta HTTP en JSON para el servicio <test> y <report>.

Por ejemplo, si se desea utilizar una herramienta de informes externa como JasperReports, si usa clases de Caché y SQL, puede utilizar su JBDC Data Connection para proporcionar los datos para tus informes, pero si tus datos están en globales o se quiere usar el enfoque orientado a documentos, el XML data adapter puede ser una opción mejor para proporcionar los datos para tu informe. En este caso, se deberá crear un servicio REST que devuelva los datos del informe en formato XML.



Para personalizar la respuesta y devolverla en XML en lugar de JSON, podemos utilizar el mismo patrón que utiliza Express para añadir middleware personalizado. La clave para decirle al módulo ewd-qoper8-express que tu mismo devolverás una respuesta personalizada es añadir una opción {nextCallback: true} a la llamada de función del enrutador en xp.app.use() y agregar tu propia devolución de llamada a la cadena de llamada de devolución de llamada que enviará la respuesta de acuerdo con el formato necesario. Por ejemplo, cuando se necesita devolver una respuesta a un XML Data Adapter para JasperReports, tu archivo de inicio ewd-xpress.js se convierte en:

```
var config = {
  managementPassword: 'keepThisSecret!',
  serverName: 'My EWD 3 Server',
  port: 8080,
  poolSize: 1,
  database: {
    type: 'cache'
  }
};
var ewdXpress = require('ewd-xpress').master;
var xp = ewdXpress.intercept();
var js2xmlparser = require('js2xmlparser');
xp.app.use('/report', xp.qx.router({ nextCallback: true }), function(req, res) {
 var message = res.locals.message;
 console.log('##### converting response to XML ... #####');
 console.dir(message, {depth:6});
 res.set('Content-Type', 'application/xml');
 if (message.error)
  res.send(js2xmlparser('error', message));
 else
  res.send(js2xmlparser(message.json.root || 'xmlRoot', message.json.data || {}));
});
```

Esto funciona exactamente igual que en el ejemplo anterior que devuelve una respuesta JSON: los controladores en tu módulo report.js serán llamados para cada tipo de informe, pero ahora convertiremos los JSON a XML usando la siguiente devolución de llamada en la cadena Express. En este caso, necesita enviar la respuesta en tu código ya que la lógica de tratamiento/envío de respuesta predeterminada se omite en este caso. La respuesta JSON de tu módulo de aplicación se devuelve ahora en res.locals.message y puede ser modificada antes de ser enviada.

Como se puede ver, aquí se utiliza otro módulo npm (js2xmlparser) además del módulo Express para convertir la respuesta JSON devuelta por tu módulo de aplicación (report.js en este ejemplo) a XML. Esta es una de las características más poderosas de Node.js porque puede utilizar todos los módulos listos para usar (¡actualmente 250.000!) en sus aplicaciones Caché.

Tu módulo de aplicación report.js puede devolver ahora una respuesta como esta (consultar el ejemplo de JasperReports):

```
{
  json: {
    data: {
      'category': [
        {
          '@': {
            'name': 'home'
          },
          'person': [
            {
              '@': {
                'id': 1
              },
              'lastname': 'Davolio',
              'firstname': 'Nancy'
            },
            ...
          ]
        },
        ...
```

```
      ]
    },
    root: 'addressbook'
  },
  error: ''
}
```

Como verás, el signo "@" se usa para devolver atributos de una etiqueta XML. Finalmente, para devolver la respuesta en XML, se debe configurar el Content-Type como application/xml y el módulo js2xmlparser convertirá su respuesta de JSON a XML:

```
<addressbook>
  <category name="home">
    <person id="1">
      <lastname>Davolio</lastname>
      <firstname>Nancy</firstname>
    </person>
    ...
  </category>
  ...
</addressbook>
```

Otro ejemplo es la creación de un servicio REST para crear documentos utilizados para EDI. Como estos documentos necesitan ser intercambiados con otras partes, están en formato XML.

Un consejo más: si estás depurando tu módulo de aplicación utilizando, por ejemplo, el cliente Advanced REST de Chrome y quieres devolver/ver la respuesta JSON de tu módulo de aplicación, puedes cambiar {nextCallback:true} a {nextCallback:false} y tu función de devolución de llamada adicional no será llamada (y no se convertirá a XML). Es recomendable inspeccionar el JSON si el XML no es el esperado.

Ahora ya puedes personalizar completamente las respuestas HTTP según tus propias necesidades ¡y puede utilizar los patrones estándar proporcionados por el módulo Express!

#API REST #JSON #Node.js #SOAP #Caché
00 2 0 0 345

Log in or sign up to continue
Añade la respuesta

URL de fuente: https://es.community.intersystems.com/post/devoluci%C3%B3n-de-respuestas-personalizadas-de-servicios-web-y-servicios-rest-utilizando-nodejs-y