Artículo
Alberto Fuentes · 15 abr, 2021 Lectura de 4 min

Open Exchange

AppS.REST: un nuevo framework REST para InterSystems IRIS

Hola a todos! Comentamos hoy una entrada de Timothy Leavitt cuyo equipo (*Application Services* en InterSystems - encargado de desarrollar y mantener muchas de nuestras aplicaciones internas, y proporcionar herramientas y prácticas recomendadas a otras aplicaciones departamentales), durante el último año, se embarcó en un viaje hacia el desarrollo de interfaces de usuario basadas en Angular/REST, para las aplicaciones existentes construidas originalmente con CSP y/o Zen. Esto ha planteado un interesante reto, que os puede resultar familiar a muchos de vosotros: desarrollar nuevas APIs REST para los modelos de datos y la lógica empresarial existentes.

Como parte de este proceso, creamos un nuevo framework para las APIs REST, que ha sido muy útil para nosotros mismos. Ya está disponible en Open Exchange en https://openexchange.intersystems.com/package/apps-rest. Creo que habrá más artículos sobre esto próximamente pero, mientras tanto, hay buenos tutoriales en la documentación del proyecto en GitHub (https://github.com/intersystems/apps-rest).

Como introducción, aquí se encuentran algunos de nuestros objetivos e intenciones de diseño. Todavía no se han realizado todos, ¡pero vamos por buen camino!

Desarrollo y despliegue rápidos

Nuestro enfoque REST debería proporcionar el mismo inicio rápido para el desarrollo de aplicaciones que Zen, ya que resuelve problemas comunes y al mismo tiempo proporciona flexibilidad para los casos de uso específicos de las aplicaciones.

- Exponer un nuevo recurso para el acceso a REST debería ser tan fácil como exponerlo a un DataModel de Zen.
- La incorporación/modificación de recursos REST debería involucrar cambios en el nivel al que se accede.
- La exposición de una clase persistente a través de REST debería llevarse a cabo por herencia y sobre-escrituras de métodos mínimos, pero también debería haber soporte para las funcionalidades equivalentes de programación manual (esto es similar a %ZEN.DataModel.Adaptor y a %ZEN.DataModel.ObjectDataModel).
- Los patrones comunes de gestión/notificación de errores, serialización/deserialización, validación, etc., no deberían necesitar ser re-implementados para cada recurso en cada aplicación.
- El soporte para las consultas, el filtrado y la clasificación de los datos de SQL, así como las funciones de búsqueda avanzada y la paginación, deberían estar incorporadas, en vez de que se vuelvan a implementar para cada aplicación.
- Debería ser fácil desarrollar APIs REST para los métodos de clase y consultas de clase de la API/librería existente, así como a nivel de objeto (CRUD).

Seguridad

La seguridad es una decisión tomada en consecuencia en el momento de diseñar/implementar en vez de una idea de última hora.

- Cuando las capacidades REST se obtienen por herencia de clases, el comportamiento por defecto debería ser NO proporcionar acceso al recurso, hasta que el desarrollador especifique activamente quién debe recibir acceso y bajo qué condiciones.

- Las implementaciones estandarizadas de las funciones relacionadas con SQL minimizan la superficie potencial para los ataques de inyección SQL.
- El diseño debe tener en cuenta el top 10 de la API OWASP (consulta: https://owasp.org/www-project-api-security)

Mantenimiento

La uniformidad del diseño de aplicaciones es una poderosa herramienta para el ecosistema de aplicaciones en una empresa.

- En vez de acumular un conjunto de API REST de diversa índole e implementaciones programadas de forma manual, deberíamos tener APIs REST de aspecto similar en toda nuestra cartera. Esta uniformidad debería llevar a:
    - Técnicas de depuración comunes.
    - Técnicas de prueba comunes.
    - Técnicas comunes de interfaz de usuario para conectarse a las APIs REST.
    - Facilidad para desarrollar aplicaciones compuestas que accedan a varias APIs.
- El conjunto de *endpoints* y el formato de las representaciones de objetos proporcionadas/aceptadas por medio de REST debería estar bien definido, de manera que podamos generar automáticamente la documentación de la API (por ejemplo, Swagger/OpenAPI) basada en estos endpoints.
- Con base en la documentación estándar de la API, deberíamos ser capaces de generar partes del código del cliente (por ejemplo, clases de typescript correspondientes a nuestras representaciones REST) con la ayuda de herramientas de terceros/estándares de la industria.

#API #API REST #Framework #JSON #Mejores prácticas #Modelo de datos #Seguridad #InterSystems IRIS #InterSystems IRIS for Health #Open Exchange
Ir a la aplicación en InterSystems Open Exchange

URL de fuente:https://es.community.intersystems.com/post/appsrest-un-nuevo-framework-rest-para-intersystems-iris