HAL - Linguagem de Aplicação de Hipertexto

Um tipo de hipermídia enxuta

- **Autor:** Mike Kelly (http://stateless.co/) < mike@stateless.co (mailto:mike@stateless.co) >

- **Criado:** 2011-06-13
- **Atualizado:** 18/09/2013 (Atualizado)

Resumo

HAL é um formato simples que oferece uma maneira consistente e fácil de hiperlink entre recursos em sua API.

Adotar o HAL tornará sua API explorável e sua documentação facilmente detectável de dentro da própria API. Em suma, tornará sua API mais fácil de trabalhar e, portanto, mais atraente para os desenvolvedores de clientes.

APIs que adotam HAL podem ser facilmente servidas e consumidas usando bibliotecas de código aberto disponíveis para a maioria das principais linguagens de programação. Também é simples o suficiente para que você possa lidar com isso como faria com qualquer outro JSON.

Sobre o autor

Mike Kelly é um engenheiro de software do Reino Unido. Ele dirige uma consultoria de API (http://stateless.co/) que ajuda as empresas a projetar e construir belas APIs que os desenvolvedores adoram.

Links Rápidos

- Uma API de demonstração usando HAL chamada HAL Talk (http://haltalk.herokuapp.com/)
- Uma lista de bibliotecas para trabalhar com HAL (Obj-C, Ruby, JS, PHP, C#, etc.) (https://github.com/mikekelly/hal_specification/wiki/Libraries)
- Uma lista de APIs de hipermídia públicas usando HAL (https://github.com/mikekelly/hal_specification/wiki/APIs)
- Grupo de discussão (perguntas, feedback, etc) (http://groups.google.com/group/hal-discuss)

Descrição geral

HAL fornece um conjunto de convenções para expressar hiperlinks em JSON ou XML.

O resto de um documento HAL é simplesmente JSON ou XML antigo.

Em vez de usar estruturas ad-hoc ou gastar um tempo valioso projetando seu próprio formato; voc pode adotar as convenções da HAL e se concentrar em construir e documentar os dados e transições que compõem sua API.

HAL é um pouco como HTML para máquinas, pois é genérico e projetado para conduzir muitos tipos diferentes de aplicativos por meio de hiperlinks. A diferença é que o HTML tem recursos para ajudar os 'atores humanos' a se moverem por meio de um aplicativo da Web para atingir seus objetivos, enquanto o HAL destina-se a ajudar os 'atores automatizados' a se moverem por uma A da Web para atingir seus objetivos.

Dito isto, **HAL é realmente muito amigável também** . Suas convenções tornam a documentação de uma API detectável a partir das próprias mensagens da API. Isso possibilita que os desenvolvedores saltem direto para uma API baseada em HAL e explorem seus recursos, sem a sobrecarga cognitiva de ter que mapear alguma documentação fora de banda em sua jornada.

Exemplos

O exemplo abaixo é como você pode representar uma coleção de pedidos com hal+json. Coisas para procurar:

- O URI do recurso principal que está sendo representado ('/orders') expresso com um link próprio
- O link 'próximo' apontando para a próxima página de pedidos
- Um link de modelo chamado 'ea:find' para pesquisar pedidos por id
- Os vários objetos de link 'ea:admin' contidos em uma matriz
- Duas propriedades da coleção de pedidos; 'atualmenteProcessando' e 'enviado hoje'
- Recursos de pedidos incorporados com seus próprios links e propriedades
- O URI compacto (curie) chamado 'ea' para expandir o nome dos links para sua URL de documentação

aplicativo/hal+json

```
{
    "_links": {
        "self": { "href": "/orders" },
        "curies": [{ "name": "ea", "href": "http://example.com/docs/rels/{rel}",
"templated": true }],
        "next": { "href": "/orders?page=2" },
        "ea:find": {
            "href": "/orders{?id}",
            "templated": true
        },
        "ea:admin": [{
            "href": "/admins/2",
            "title": "Fred"
        }, {
            "href": "/admins/5",
            "title": "Kate"
        }]
    },
    "currentlyProcessing": 14,
    "shippedToday": 20,
    "_embedded": {
        "ea:order": [{
            "_links": {
                "self": { "href": "/orders/123" },
                "ea:basket": { "href": "/baskets/98712" },
                "ea:customer": { "href": "/customers/7809" }
            },
            "total": 30.00,
            "currency": "USD",
            "status": "shipped"
        }, {
            "_links": {
                "self": { "href": "/orders/124" },
                "ea:basket": { "href": "/baskets/97213" },
                "ea:customer": { "href": "/customers/12369" }
            },
            "total": 20.00,
            "currency": "USD",
            "status": "processing"
        }]
    }
}
```

O modelo HAL

As convenções HAL giram em torno da representação de dois conceitos simples: *Recursos* e *Link*

Recursos

Os recursos têm:

- Links (para URIs)
- Recursos incorporados (ou seja, outros recursos contidos neles)

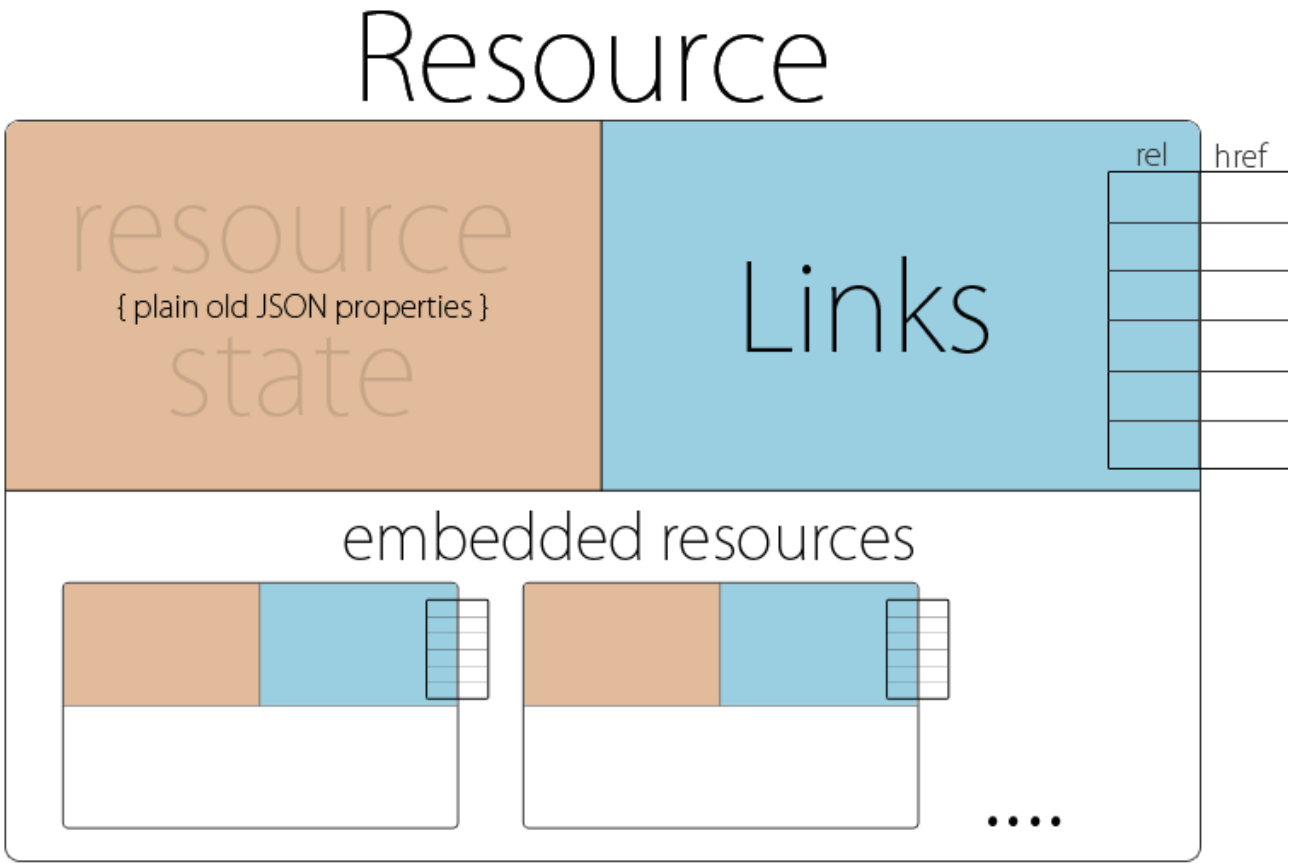
- Estado (seus dados JSON ou XML padrão do pântano)

Links

Os links têm:

- Um destino (um URI)
- Uma relação aka. 'rel' (o nome do link)
- Algumas outras propriedades opcionais para ajudar na descontinuação, negociação de conteúdo etc.

Abaixo está uma imagem que ilustra aproximadamente como uma representação HAL é estruturada:



Como o HAL é usado em APIs

O HAL foi projetado para criar APIs nas quais os clientes navegam pelos recursos seguindo os links.

Os links são identificados por relações de link. As relações de link são a força vital de uma API de hipermídia: são como você informa aos desenvolvedores clientes sobre quais recursos estão disponíveis e como eles podem interagir, e são como o código que eles escrevem selecionará qua link percorrer.

As relações de link não são apenas uma string de identificação no HAL. Na verdade, são URLs, qu os desenvolvedores podem seguir para ler a documentação de um determinado link. Isso é o que conhecido como "descoberta". A ideia é que um desenvolvedor possa entrar em sua API, ler a documentação dos links disponíveis e, em seguida, seguir seu nariz pela API.

A HAL incentiva o uso de relações de link para:

- Identifique links e recursos incorporados na representação
- Inferir a estrutura e o significado esperados dos recursos de destino
- Sinalizando quais solicitações e representações podem ser enviadas aos recursos de destino

Como servir HAL

HAL tem um tipo de mídia para as variantes JSON e XML, cujos nomes são `application/hal+json`e `application/hal+xml`respectivamente.

Ao servir HAL sobre HTTP, `Content-Type`a resposta deve conter o nome do tipo de mídia relevante.

A estrutura de um documento HAL

Documento mínimo válido

Um documento HAL deve conter pelo menos um recurso vazio.

Um objeto JSON vazio:

```
{}
```

Recursos

Na maioria dos casos, os recursos devem ter um URI próprio

Representado através de um link 'self':

```
{
    "_links": {
        "self": { "href": "/example_resource" }
    }
}
```

Links

Os links devem estar contidos diretamente em um recurso:

Os links são representados como objetos JSON contidos em um `_links`hash que deve ser uma propriedade direta de um objeto de recurso:

```
{
    "_links": {
        "next": { "href": "/page=2" }
    }
}
```

Relações de link

Os links têm uma relação (também conhecida como 'rel'). Isso indica a semântica - o significado - de um link específico.

Os links rel são a principal maneira de distinguir os links de um recurso.

É basicamente apenas uma chave dentro do `_links`hash, associando o significado do link (o 'rel') ao objeto de link que contém dados como o valor 'href' real:

```
{
    "_links": {
        "next": { "href": "/page=2" }
    }
}
```

Descoberta da API

Os links rels devem ser URLs que revelam a documentação sobre o link fornecido, tornando-os "detectáveis". Os URLs são geralmente bastante longos e um pouco desagradáveis para serem usados como chaves. Para contornar isso, o HAL fornece "CURIEs" que são basicamente tokens nomeados que você pode definir no documento e usar para expressar URIs de relação de link de

maneira mais amigável e compacta, ou seja, `ex:widget`em vez de `http://example.com/rels/widget`. Os detalhes estão disponíveis na seção sobre CURIEs u pouco mais abaixo.

Representando vários links com a mesma relação

Um recurso pode ter vários links que compartilham a mesma relação de link.

Para relações de link que podem ter vários links, usamos uma matriz de links.

```
{
    "_links": {
      "items": [{
          "href": "/first_item"
      },{
          "href": "/second_item"
      }]
    }
}
```

Observação: se você não tiver certeza se o link deve ser singular, suponha que será múltiplo. Se você escolher singular e achar que precisa alterá-lo, precisará criar uma nova relação de link ou enfrentar a quebra de clientes existentes.

CURIE

"CURIE"s ajudam fornecendo links para documentação de recursos.

HAL lhe dá uma relação de link reservada 'curies' que você pode usar para indicar a localização d documentação do recurso.

```
"_links": {
  "curies": [
    {
      "name": "doc",
      "href": "http://haltalk.herokuapp.com/docs/{rel}",
      "templated": true
    }
  ],

  "doc:latest-posts": {
    "href": "/posts/latest"
  }
}
```

Pode haver vários links na seção 'curies'. Eles vêm com um 'nome' e um modelo 'href' que deve conter o `{rel}`espaço reservado.

Os links, por sua vez, podem prefixar seu 'rel' com um nome CURIE. Associar o `latest-posts`lir com a `doc` documentação CURIE resulta em um link 'rel' definido como `doc:latest-posts`.

Para recuperar a documentação sobre o `latest-posts`recurso, o cliente expandirá o link CURIE associado com o 'rel' do link real. Isso resultaria em uma URL `http://haltalk.herokuapp.com/docs/latest-posts`que deve retornar a documentação sobre esse recurso.

Continua...

Esta especificação relativamente informal de HAL está incompleta e ainda em andamento. Por enquanto, se você quiser ter um entendimento completo, leia a especificação formal (http://tools.ietf.org/html/draft-kelly-json-hal) .

RFC

A variante JSON de HAL (application/hal+json) agora foi publicada como um rascunho da Internet: draft-kelly-json-hal (http://tools.ietf.org/html/draft-kelly-json-hal) .

Reconhecimentos

- Darrel Miller
- Mike Amundsen
- Mark Derricutt
- Herman Radtke
- Will Hartung
- Steve Klabnik
- todos no hal-discussão

Obrigado pela ajuda :)

Notas/tarefas