

Consumindo Serviços Remotos com CAP Java

O Caminho Recomendado para Integração de APIs OData V2 e V4



Um deck de referência para desenvolvedores Java.

O Objetivo: Simplicidade e Poder com CQN

```
// Injeção do serviço remoto configurado  
@Autowired  
@Qualifier(ApiBusinessPartner_.CDS_NAME)  
private CqnService bupa;  
  
// Execução de uma query CQN de forma nativa  
CqnSelect select = Select.from(ABusinessPartnerAddress_.class)  
    .where(a -> a.BusinessPartner().eq("4711"));  
  
ABusinessPartnerAddress address = bupa.run(select)  
    .single(ABusinessPartnerAddress.class);
```



Sua query CQN é traduzida para HTTP automaticamente. Aspectos como segurança e propagação de usuário são gerenciados por configuração, não por código.

Por que Remote Services são a Abordagem Padrão?

Os Remote Services são clientes baseados em CQN que abstraem a complexidade da comunicação com APIs remotas.

-  **Abstração CQN:** Libera os desenvolvedores de terem que mapear CQN para OData (V2/V4) manualmente.
-  **Segurança por Configuração:** Aspectos como autenticação, resiliência e propagação de tenant/usuário são tratados pelo framework.
-  **Integração com Cloud SDK:** Aproveita nativamente os `destinations` e `service bindings` do SAP Cloud SDK para conectividade.
-  **Extensibilidade Multitenant:** Mantém as aplicações extensíveis e simplifica o mocking para testes.

A recomendação clara do CAP é usar Remote Services em vez de usar diretamente o SAP Cloud SDK.

A Anatomia de um Remote Service

Definição do Serviço (O 'Contrato')

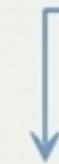
Uma definição de serviço do modelo CDS que especifica la API remota.



Conectividade (O 'Endereço')

A configuração de um `destination` (BTP ou programático) ou `service binding` que define o endpoint e a autenticação.

 Use o comando `cds import` para gerar esta definição a partir de uma especificação EDMX.



No seu application.yaml, você conecta o “Contrato” ao “Endereço” para habilitar o serviço.

Passo 1: Habilitando a Funcionalidade no seu Projeto

Para habilitar os Remote Services para APIs OData V2 ou V4, adicione a seguinte dependência Maven ao seu pom.xml.

```
<dependency>
    <groupId>com.sap.cds</groupId>
    <artifactId>cds-feature-remote-odata</artifactId>
    <scope>runtime</scope>
</dependency>
```

Observação: Esta dependência inclui o mínimo necessário. Dependências adicionais do Cloud SDK podem ser necessárias para funcionalidades avançadas como integração com o Destination Service do BTP.

Passo 2: Configuração Essencial no `application.yaml`

Os Remote Services devem ser configurados explicitamente. A configuração define o protocolo e o link para a definição CDS.

```
cds:  
  remote.services:  
    # Nome único do serviço, usado para lookup em Java  
    API_BUSINESS_PARTNER:  
      1 # (1) Define o protocolo. Padrão: "odata-v4"  
      type: "odata-v2"  
      2 # (2) Define o nome da definição do serviço no modelo CDS.  
      # Opcional, se o nome do serviço (API_BUSINESS_PARTNER) já for o mesmo.  
      model: "NomeDaDefinicaoNoCDS"  
      3 # (3) A configuração de conectividade virá aqui...  
      destination:  
        name: "s4-business-partner-api"
```

1. **`type`**: Define o protocolo. Valores suportados: odata-v4 (padrão) e odata-v2.
2. **`model`**: Permite usar um nome de serviço Java único (bupa-abc) que aponta para uma definição CDS compartilhada (API_BUSINESS_PARTNER).
3. **Conectividade**: A próxima decisão é **como** conectar: via binding ou destination.

O Ponto de Decisão: Como Você se Conecta?



O Caminho Integrado (BTP)

Ideal para APIs rodando no SAP BTP. O `service binding` abstrai a URL e os detalhes de autenticação. É a forma mais simples e recomendada quando disponível.

****When to Use****

Quando você possui um `service binding` para um serviço de reuso ou para uma aplicação CAP na mesma subconta com identidade compartilhada.



O Caminho Flexível

Necessário quando um `service binding` não está disponível. Você armazena a URL e as credenciais em um `destination` (no BTP Destination Service ou programaticamente).

****When to Use****

Para APIs externas, on-premise, ou quando a configuração de conectividade precisa ser gerenciada separadamente da aplicação.

Conectando com Service Bindings

Cenário 1: Binding para um Serviço de Reuso

Se um broker de serviço fornece uma instância, você só precisa especificar o nome do binding. O CAP extrai a URL e a autenticação automaticamente.

```
cds:  
  remote.services:  
    SomeReuseService:  
      binding:  
        name: "some-service-binding" # Nome do seu binding
```

Cenário 2: Binding para um Serviço com Identidade Compartilhada (XSUAA/IAS)

Para comunicação entre apps CAP que compartilham a mesma instância de identidade, o binding não contém a URL. Você precisa configurá-la explicitamente.

```
cds:  
  remote.services:  
    OtherCapService:  
      binding:  
        name: "shared-xsuaa" # Nome do binding do serviço de identidade  
        options:  
          url: "${CDS_REMOTE_SERVICES_OTHERCAPSERVICE_BINDING_OPTIONS_URL}"
```

Definindo a Estratégia de Propagação de Usuário

A propriedade `onBehalfOf` na seção de `binding` define qual identidade será usada para chamar o serviço remoto.

```
cds:  
  remote.services:  
    SomeReuseService:  
      binding:  
        name: "some-service-binding"  
        onBehalfOf: "currentUser" # Escolha a sua estratégia
```

Opções Disponíveis

- `currentUser` (padrão): Propaga o usuário nomeado do contexto. Se não houver, usa um usuário técnico (específico do tenant).
- `systemUser`: Sempre usa um usuário técnico (específico do tenant).
- `systemUserProvider`: Usa um usuário técnico do tenant provedor. Útil para comunicação interna não autorizada por tenant.

Conectando com Destinations

Quando um `service binding` não é uma opção, use um `destination`. O CAP Java usa o Cloud SDK `DestinationAccessor` para encontrar o `destination` configurado pelo nome.

```
cds:  
  remote.services:  
    API_BUSINESS_PARTNER:  
      type: "odata-v2"  
      destination:  
        # O nome do destination a ser buscado no BTP Destination Service  
        # ou registrado programaticamente.  
        name: "s4-business-partner-api"
```

Observação Importante: Para que a busca no BTP Destination Service funcione, dependências adicionais do Cloud SDK podem ser necessárias no seu `pom.xml`. Em cenários multitenant, a busca ocorre na subconta do tenant atual.

Detalhes da Configuração de Destinations

Tópico 1: Construindo a URL do Serviço

A URL final é uma combinação de 3 partes:

URL base do `destination` + `http.suffix` (Opcional) + Nome do Serviço CDS = Final URL

```
# destination URL: https://s4.sap.com
cds:
  remote.services:
    API_BUSINESS_PARTNER:
      destination:
        name: "s4-business-partner-api"
        http:
          suffix: "/sap/opu/odata/sap"
# URL final da request: https://s4.sap.com/sap/opu/odata/sap/API_BUSINESS_PARTNER
```

Tópico 2: Consumindo APIs de Outras Aplicações IAS

Para chamadas entre aplicações que confiam no mesmo tenant IAS, configure um `destination` com `NoAuthentication` e uma propriedade adicional.

Configuração do Destination (no BTP)

- URL: <url-da-api-remota>
- Authentication: NoAuthentication
- Additional Properties: **cloudsdk.ias-dependency-name**: <nome-da-dependencia-ias>

Tópico Avançado: Manipulando Mídia (Imagens, Documentos)

Contexto (Modelo CDS)

```
entity Media {  
    key ID: UUID;  
    @Core.MediaType: 'image/png'  
    image: LargeBinary;  
}
```

Leitura de Mídia

- Regra:** Use uma query `Select` que busca uma **única instância** pela chave e seleciona o campo de mídia.
- Resultado:** O conteúdo é retornado como um `InputStream` ou `Reader` que deve ser consumido para liberar a conexão HTTP.

Escrita e Deleção de Mídia

- Regra:** Use uma query `Update` em uma **única instância**.
- Para Escrever:** Forneça o valor do campo como um `InputStream` ou `Reader`.
- Para Deletar:** Defina o valor do campo como `null`. Isso gera uma request `DELETE` para o elemento de mídia.

Aviso: Updates em lote ou misturando campos de mídia e não-mídia na mesma query são ignorados.

Tópico Avançado: Integração com o Cloud SDK

Tópico 1: Gerenciamento de Dependências

Recomendação

Para usar funcionalidades avançadas, adicione o SAP Cloud SDK BOM ao seu pom.xml.

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>com.sap.cloud.sdk</groupId>
            <artifactId>sdk-bom</artifactId>
            <version><!-- use-latest-version --></version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

Tópico 2: Registro Programático de Destinations

Caso de Uso

Útil para criar destinations dinamicamente durante a inicialização da aplicação, por exemplo, para injetar uma APIKey de uma variável de ambiente.

```
@Component
public class DestinationConfiguration implements EventHandler {
    @Value("${api-hub.api-key}")
    private String apiKey;

    @Before(event = ApplicationLifecycleService.EVENT_APPLICATION_PREPARED)
    public void initializeDestinations() {
        DefaultHttpDestination httpDestination = DefaultHttpDestination
            .builder("https://sandbox.api.sap.com/s4hanacloud")
            .header("APIKey", apiKey)
            .name("s4-business-partner-api").build();
        DestinationAccessor.prependDestinationLoader(
            new DefaultDestinationLoader().registerDestination(httpDestination));
    }
}
```

A Rota de Fuga: Consumo Nativo com o Cloud SDK

Quando Usar: Se você precisa chamar um endpoint que não pode ser consumido como um Remote Service (ex: não-OData), você pode recorrer às APIs nativas do Cloud SDK.

O Que Envolve:

- Uso direto do `HttpClientAccessor` para obter um `HttpClient`.
- Operações de baixo nível como serialização e deserialização do payload.
- Maior verbosidade no código.

Exemplo de Código Java (usando um Destination existente)

```
// Obtenha o destination (do BTP ou programático)
HttpDestination destination = DestinationAccessor.getDestination("<destination-name>");

// Obtenha um HttpClient pré-configurado com autenticação e resiliência
HttpClient httpClient = HttpClientAccessor.getHttpClient(destination);

// Execute requests HTTP...
```

 Esta abordagem oferece flexibilidade máxima, mas perde as abstrações do CAP. Use os Remote Services sempre que possível.

Resumo: O Caminho Recomendado para Consumo de Serviços



Adote os Remote Services como sua abordagem padrão. Eles fornecem uma abstração robusta e segura, permitindo que você se concentre na lógica de negócios em vez da complexidade da integração.



Next Steps: Para um guia completo de ponta a ponta com exemplos práticos, consulte o cookbook **Consuming Services** na documentação oficial do CAP.