

Dominando o Change Tracking no CAP Java

O guia completo para criar trilhas de auditoria
inteligentes e integradas.

Preparando o Terreno: Dependências e Pré-requisitos

Antes de rastrear qualquer alteração, vamos garantir que seu projeto está configurado corretamente.

Dependência do Serviço (srv/pom.xml)

Adicione a dependência `cds-feature-change-tracking` ao seu `pom.xml`.

```
<dependency>
    <groupId>com.sap.cds</groupId>
    <artifactId>cds-feature-change-
tracking</artifactId>
    <scope>runtime</scope>
</dependency>
```

Configuração do Build

Seu POM também deve incluir o *goal* para resolver o modelo CDS fornecido pelo feature.

Nota: Para mais detalhes, consulte a documentação 'Reference the New CDS Model in an Existing CAP Java Project'.

Requisito de UI

Para usar a UI integrada com SAP Fiori elements, certifique-se de que sua versão do SAP UI5 é **1.121.2 ou superior**.



O Change Tracking captura operações de modificação via CQN. Alterações feitas por SQL nativo, JDBC ou outros meios que contornam o runtime do CAP Java não são rastreadas.

A Receita Essencial: Habilitando o Rastreamento em 2 Passos

Para o nosso exemplo, usaremos uma entidade de domínio simples `model.Books`.

Passo 1: Estender a Entidade de Domínio

Incorpore o aspect `changelog.changeTracked` à sua entidade. Isso adiciona a associação `changes`, que expõe o histórico de alterações.

`cds`

```
using {sap.changelog as changelog} from 'com.sap.cds/change-tracking';  
extend model.Books with changelog.changeTracked;
```

Passo 2: Anotar os Elementos a Serem Rastreados

Use a anotação `@changelog` nos campos específicos cujas alterações você deseja registrar.

`cds`

```
annotate Bookshop.Books {  
    title @changelog;  
    stock @changelog;  
};
```

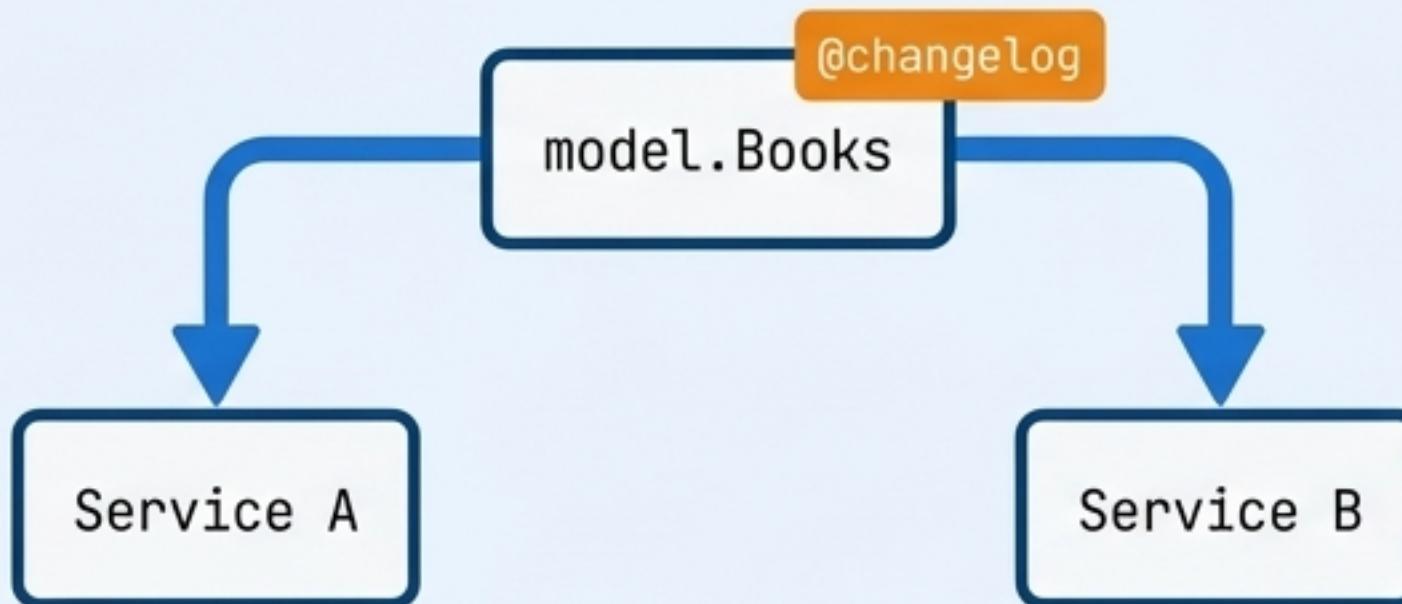


A Decisão Crítica: Anotar no Nível de Domínio ou de Serviço?

O local onde você aplica a anotação `@changelog` tem implicações diretas sobre o que é rastreado. A escolha é fundamental.

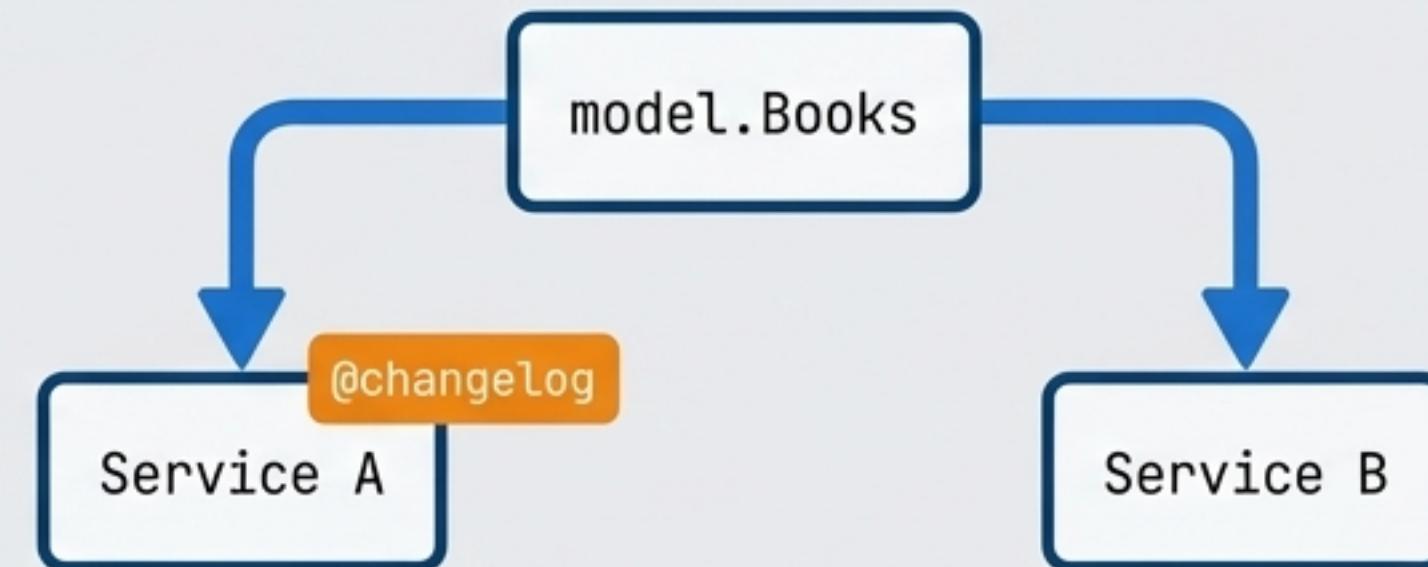
Anotação em Nível de Domínio (`model.Books`)

- O quê:** Rastreia alterações feitas através de **qualquer** projeção da entidade.
- Quando Usar:** Ideal para garantir uma trilha de auditoria completa e consistente em toda a aplicação.



Anotação em Nível de Serviço (`Bookshop.Books`)

- O quê:** Rastreia alterações feitas **apenas** através daquela projeção de serviço específica.
- Quando Usar:** Perfeito para cenários onde você quer evitar o rastreamento de operações de back-end, como replicação de dados ou alterações em massa, que podem ser custosas.



Nota Importante: Elementos anotados com `@PersonalData` (sujeitos a audit logging) são ignorados pelo Change Tracking.

Dando Sentido às Mudanças: Identificadores de Entidade

O Problema

Um log que diz "ID 123-456, campo 'stock' alterado de 10 para 9" é técnico, mas pouco informativo. Qual livro foi alterado?

A Solução

Anote a entidade com `@changelog` para definir um "identificador amigável". Esses valores serão armazenados junto com cada alteração.

Como Funciona

1. Quando stock ou outro campo anotado muda, o valor atual de title é capturado.
2. Esse valor é armazenado junto com o registro da alteração.

Dica de Mestre

Escolha campos que são "insert-only" ou que mudam com pouca frequência (como um número de pedido ou título) para serem seus identificadores.

```
cds
// Adiciona o título do livro a cada entrada de
log para essa entidade
annotate Bookshop.Book with @changelog: [
    title
];
```

Rastreando Estruturas Complexas: Composições

Como rastrear alterações em uma `Order` e seus `OrderItems` em uma única operação "deep update"?

A Regra de Ouro: Estenda a entidade **raiz** da composição com `changelog.changeTracked`. Não a entidade filha.

Exemplo de Modelo (CDS)

```
entity Orders : cuid {  
    customerName: String @changelog;  
    items: Composition of many OrderItems on items.parent = $self;  
}  
  
entity OrderItems : cuid {  
    parent: Association to Orders; ←  
    quantity: Integer @changelog;  
}
```

Composition

Implementação Correta

```
extend Orders with changelog.changeTracked;
```



Implementação Incorreta

```
extend OrderItems with changelog.changeTracked;
```



Resultado: Com a configuração correta, todas as alterações, incluindo as profundas nos `OrderItems`, são associadas corretamente à `Orders` raiz.

Substituindo Chaves Estrangeiras por Valores Legíveis

O Problema

Por padrão, ao alterar uma associação (ex: `customer`), o log armazena o valor da chave estrangeira (um UUID), o que não é útil para um usuário.

customer_ID: "a1b2c3d4-..."



A Solução

Use a anotação `@changelog` na associação para especificar quais campos da entidade associada devem ser armazenados.

customer.name: "ACME Corp"

Exemplo Prático

Para uma entidade `Orders` com uma associação `customer`, podemos armazenar o nome do cliente.

```
annotate Orders {  
    // Para a associação 'customer', armazene o valor do campo 'name'  
    // da entidade Customer associada.  
    customer @changelog: [ customer.name ]  
}
```

! Pré-requisito Crítico

A entidade alvo da associação deve existir. Se uma `Order` for atualizada com um `customer_ID` inválido, a entrada de log não será criada. Use validações como `@assert.target` para evitar isso.

Identificadores na Prática: Pontos de Atenção

Herança em Projeções

Projeções de uma entidade anotada herdam a anotação `@changelog`.



Atenção: Se você excluir ou renomear um campo usado no identificador da projeção, você **deve** reanotar a projeção com os nomes de elementos atualizados para evitar erros.

Valores "As-Is"

Os valores do identificador são armazenados como estão no momento da alteração.

Eles **não são traduzidos** e podem não ser formatados de acordo com a localidade do usuário (ex: moedas, unidades de medida).

Identificadores em Composições

Nenhuma anotação especial é necessária para composições. Os identificadores da entidade alvo (`OrderItems`, em nosso exemplo) são usados automaticamente.

Exemplo (CDS)

```
// Define o identificador para a raiz  
annotate Orders with @changelog: [OrderNo];  
  
// Define o identificador para o item, incluindo o da raiz  
annotate OrderItems with @changelog: [  
    parent.OrderNo,  
    supplierName  
];
```



Expondo o Histórico: Integrando com a UI do Fiori

Como Funciona

O aspect `changelog.changeTracked` expõe uma associação chamada `changes`. Você pode usar essa associação para exibir o log em uma Object Page.

Adicionando a Facet de Histórico

Adicione um `UI.ReferenceFacet` às anotações da sua entidade para criar uma nova aba ou seção para o histórico de alterações.

```
annotate Bookshop.Books with @(
    UI : {
        Facets : [
            {
                $Type : 'UI.ReferenceFacet',
                ID : 'ChangeHistoryFacet',
                Label : '{i18n>ChangeHistory}', -----
                Target : 'changes/@UI.PresentationVariant',
                ![@UI.PartOfPreview]: false
            }
        ]
    });
}
```

Book
O Senhor dos Anéis

Geral Detalhes Associações **Histórico de Alterações**

Campo	Valor Antigo	Valor Novo	Data
Preço	20.00 USD	25.00 USD	15/05/2024
Autor	J.R.R. Tolkien	J.R.R. Tolkien (Revisado)	10/05/2024
Título	O Senhor dos Anéis: A Sociedade	O Senhor dos Anéis	01/05/2024

Refinando a Experiência: Customizando a UI do Log

Você pode customizar a aparência e o comportamento da lista de alterações usando anotações `UI.PresentationVariant` e `UI.LineItem`.

Opção 1: Customização por Entidade

Anotar a associação `changes` diretamente na sua entidade (`Bookshop.Books.changes`).

Exemplo (CDS): Ordenar as mudanças pela data de criação, da mais recente para a mais antiga.

```
annotate Bookshop.Books.changes with @UI: {
    PresentationVariant: {
        Visualizations: ['@UI.LineItem'],
        SortOrder: [
            {
                Property: change.createdAt,
                Descending: true
            }
        ],
        LineItem: [...]
    });
}
```

Opção 2: Customização Global

Anotar a entidade `sap.changelog.ChangeLink` para aplicar a mesma customização a **todas** as entidades rastreadas.



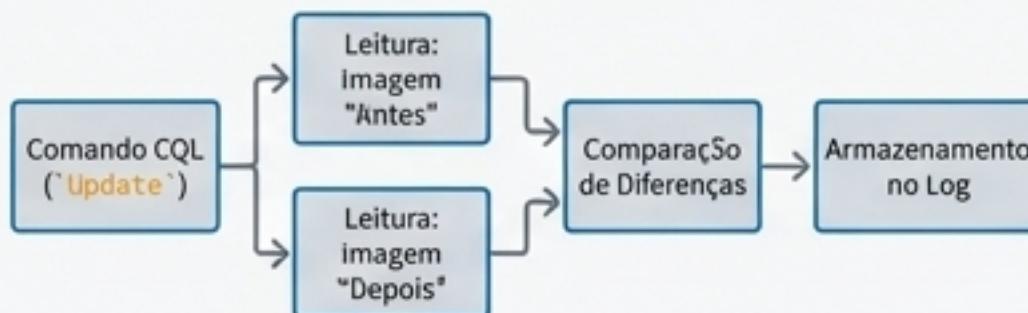
Dica

Use `@UI.Hidden` para ocultar colunas desnecessárias da UI.

Ex: `annotate Bookshop.Books.changes:up_ with @UI.Hidden;`

Por Baixo dos Panos: Como as Alterações São Armazenadas e Detectadas

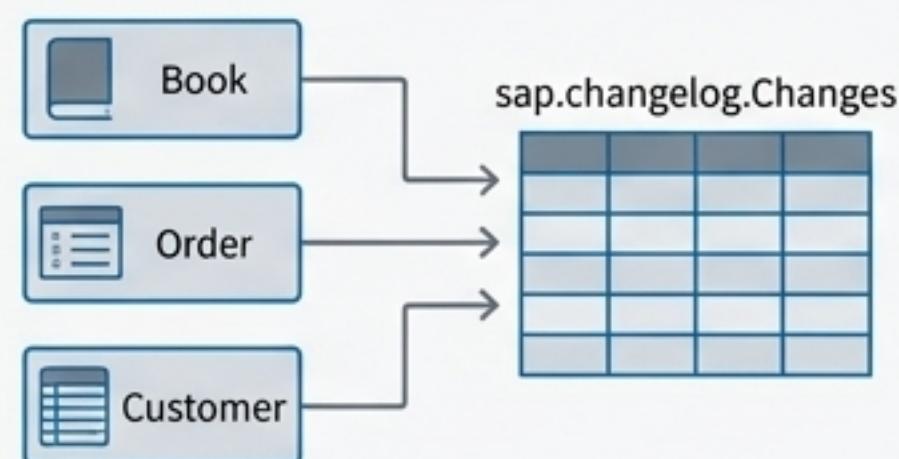
Mecanismo de Detecção



Estrutura de Armazenamento

As alterações de todas as entidades são armazenadas em uma única tabela "plana", representada pela entidade `sap.changelog.Changes`.

- **Cada entrada contém:** Natureza da mudança (add/mod/del), nome da entidade, atributo alterado, valor antigo/novo, usuário, timestamp, etc.



Impacto na Performance

As leituras adicionais podem impactar a performance de operações de `Update`, especialmente em massa.

Regra de Ouro: Sempre que possível, use chaves primárias para modificar entidades rastreadas. Evite "searched updates".

Gerenciando o Ciclo de Vida do Log: Exclusão em Cascata

Comportamento Padrão

Por padrão, os registros de log **permanecem** no banco de dados mesmo quando suas entidades correspondentes são excluídas.

O Risco

Com o tempo, isso pode levar a um grande volume de logs órfãos, consumindo espaço de armazenamento desnecessariamente.

A Solução

Para vincular o ciclo de vida do log ao da entidade, use a anotação `@cascade: { delete }`:



Implementação

Opção 1 (Por Entidade): Aplique a anotação na sua entidade de domínio.

`cds`

```
annotate model.Books.changes:change  
with @cascade: { delete };
```

Opção 2 (Global): Aplique a anotação na entidade `ChangeLink` para habilitar a exclusão em cascata para **todas** as entidades no modelo.

`cds`

```
annotate sap.changelog.ChangeLink:change  
with @cascade: { delete };
```

Indo Além da Auditoria: Reagindo Programaticamente às Mudanças

E se você precisar acionar uma notificação, invalidar um cache ou iniciar um workflow sempre que uma alteração for registrada?

Passo 1: Atualizar o Escopo da Dependência

Para acessar as classes do Change Tracking em seu código Java, mude o escopo da dependência de `runtime` para `compile` em seu `srv/pom.xml`.

```
<dependency>
    <groupId>com.sap.cds</groupId>
    <artifactId>cds-feature-change-tracking</artifactId>
    <scope>compile</scope>
</dependency>
```

Passo 2: Implementar um Event Handler

Crie um handler que escuta o evento `CREATE_CHANGES` no serviço `ChangeTrackingService`.

```
import cds.gen.sap.changelog.Changes;

@Component
@ServiceName(ChangeTrackingService.DEFAULT_NAME)
public class ChangeTrackingHandler implements EventHandler {

    @After(event = ChangeTrackingService.CREATE_CHANGES)
    void afterCreate(CreateChangesEventContext context) {
        context.getResult().listOf(Changes.class).forEach(c -> {
            // Faça algo com a entrada de log (c)
        });
    }
}
```

Atenção: Este evento é disparado para cada instrução e não está vinculado a uma transação ou operação em lote maior.

Tópico Avançado: Modificando Documentos Profundos via Java

Modificar itens dentro de uma composição requer uma abordagem específica para garantir que as alterações sejam rastreadas corretamente.

Cenário 1: "Deep Update" com Delta

Use `CdsList.delta()` para representar apenas os itens que estão sendo alterados.

```
Orders order = Orders.create("...");  
order.setOrderNo("N1");  
Order.setOrderNo("N1");  
  
OrderItems item = OrderItems.create("...");  
item.setQuantity(3);  
order.setItems(CdsList.delta(item));  
  
Update.entity(Orders_.class).entry(order);
```

Cenário 2: Modificação Direta de um Item (com Path Expression)

Evite modificações diretas (`Update.entity(OrderItems_.class)`), pois não são suportadas. Em vez disso, construa um caminho a partir da raiz.

```
// Atualiza um item específico  
Update.entity(Orders_.class, o ->  
    o.filter(f -> f.ID().eq("...").items().filter(...));  
  
// Deleta um item específico  
Delete.from(Orders_.class, o ->  
    o.filter(f -> f.ID().eq("...").items().filter(...));  
  
// Deleta todos os itens (bulk)  
Delete.from(Orders_.class, o ->  
    o.filter(f -> f.ID().eq("...").items()));
```

Checklist de Produção: Considerações Finais Essenciais

Antes de implantar, revise estes pontos críticos para garantir uma solução robusta, performática e segura.



Custos de Armazenamento

O log pode crescer rapidamente. Defina uma política de retenção e use a exclusão em cascata para gerenciar o tamanho.



Impacto na Performance

As leituras extras durante os `updates` têm um custo. Avalie o impacto em operações em massa e considere usar o rastreamento em nível de serviço para mitigar.



Onde e Como as Mudanças Ocorrem

Reavalie sua decisão de rastrear em nível de domínio vs. serviço. Rastreie apenas onde for necessário (ex: UIs de interação do usuário), não em processos de replicação de dados.



Implicações de Segurança

Se você expõe o log completo ao usuário, como as regras de acesso complexas da sua entidade se aplicam ao log? Planeje a segurança do acesso aos dados históricos.