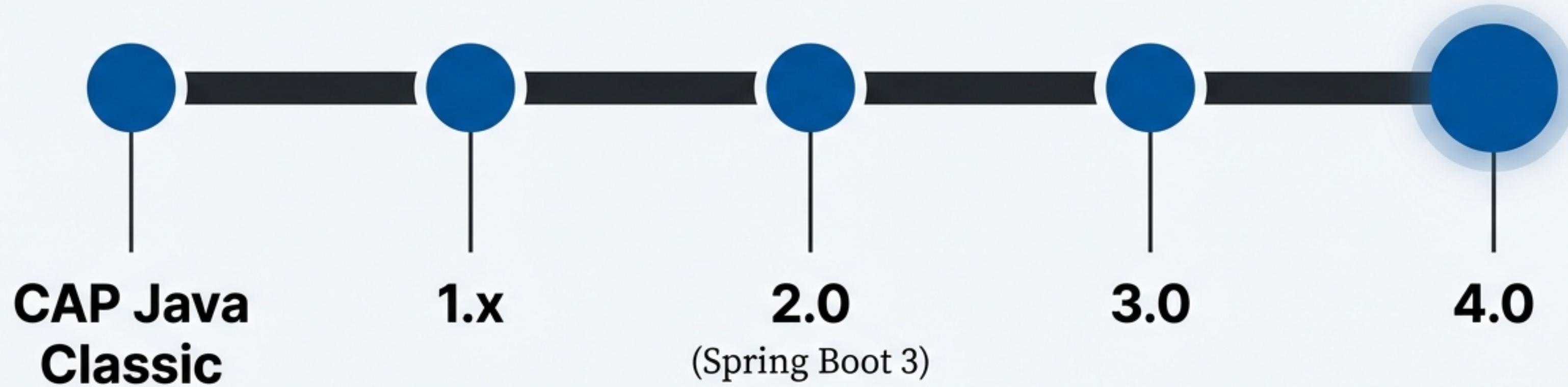


Navegando na Evolução do CAP Java

Seu Guia Definitivo para um Upgrade Tranquilo e Eficiente

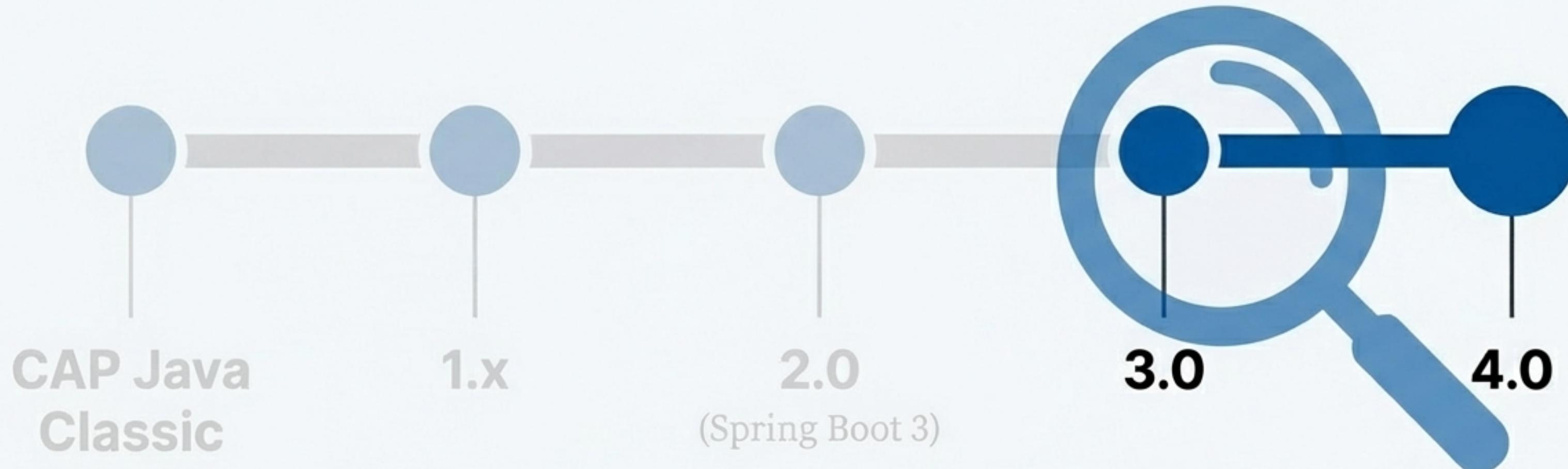
Este guia condensa os manuais de migração oficiais do CAP Java em um formato visual e prático. Criado para desenvolvedores experientes, foca nas mudanças críticas e fornece um caminho claro para cada etapa de atualização.

O Roteiro de Migração do CAP Java



Vamos navegar pelas atualizações em ordem cronológica reversa, começando pela mais recente. Use este roteiro para encontrar rapidamente a seção relevante para o seu projeto.

Missão 1: Atualizando da Versão 3.10 para a 4.0



Nesta seção, abordamos as mudanças essenciais que você precisa conhecer ao migrar para a versão mais recente do CAP Java. Foco em licença, dependências e remoção de features.

Mudanças Fundamentais na Versão 4.0



Nova Licença

A versão 4.0 é lançada sob o novo [*SAP Developer License Agreement for CAP*](#). Verifique os termos para garantir a conformidade.



Versões Mínimas

A versão mínima do @sap/cds-dk foi elevada para a ^8. O suporte para a versão ^7 foi descontinuado.



Remoção da Feature cds-feature-event-hub

Esta feature foi removida. O sucessor é o plugin open-source [*CDS Plugin for SAP Cloud Application Event Hub*](#), que começa na versão 4.0.0 com o mesmo *group* e *artifact ID*.

API, Propriedades e Comportamento na Versão 4.0

⚠ Remoção de Mensagens Não Estruturadas

O suporte para mensagens não estruturadas (plain strings) via `MessagingService` foi completamente removido. Todas as mensagens agora devem ser estruturadas.

```
// Em vez de:  
messagingService.emit("myTopic", "unstructured message");
```

```
// Use:  
messagingService.emit("myTopic",  
    Map.of("message", "unstructured message"));
```

APIs Removidas

- void MessagingService.emit(String, String)
- String TopicMessageEventContext.getData()

Outras Mudanças de Propriedade

 **Default Alterado:** `cds-maven-plugin` -> `generate` -> `excludes`: `\"localized.*\"` agora é null.

Propriedades Removidas:

- cds.messaging.services.<key>.structured
- cds.security.authorization.emptyAttribute
ValuesAreRestricted

(Agrupe outras de forma concisa) Propriedades relacionadas ao multitenancy clássico e ao serializador do ODataV4 também foram removidas. Consulte o guia oficial para a lista completa.

Missão 2: Atualizando da Versão 2.10 para a 3.0



A versão 3.0 introduziu mudanças significativas no perfil de produção, segurança e suporte a multitenancy. Vamos analisar os pontos mais críticos.

Mudanças Críticas na Versão 3.0

↑ Versões Mínimas

CAP Java 3.0 agora exige versões mínimas atualizadas:

Dependência	Versão Mínima
Cloud SDK	5.9.0
@sap/cds-dk	^7
Maven	3.6.3



Perfil de Produção `cloud`

O perfil de produção agora tem como padrão `cloud`, ativando configurações seguras para ambientes produtivos. Isso desativa a página de índice (`/`) por padrão. Se você usa essa rota para health checks, migre para o endpoint do Spring Boot Actuator: `/actuator/health`.



Remoção do Suporte ao MTX Clássico

O suporte ao MTX clássico (`@sap/cds-mtx`) foi removido em favor do MTX simplificado. A API `MtSubscriptionService` foi substituída pela `DeploymentService`.



⚠ **Breaking Change:** Com a nova `DeploymentService`, o conteúdo do tenant (como contêineres HDI) agora é excluído por padrão durante o `unsubscribe`.

Evolução da Segurança na Versão 3.0

🛡️ Remoção da Feature `cds-feature-xsuaa`

A feature `cds-feature-xsuaa` foi removida. O suporte para XSUAA e IAS foi unificado sob a `cds-feature-identity`, que utiliza a biblioteca `spring-security` da SAP em vez da antiga `spring-xsuaa`.

Se você customizou a configuração de segurança, será necessário adaptá-la para a nova biblioteca. Se tinha uma dependência direta, use os starters `cds-starter-cloudfoundry` ou `cds-starter-k8s`.

🛡️ Proof-of-Possession (PoP) para Autenticação IAS

Em cenários com IAS, a validação de Proof-of-Possession agora é obrigatória por padrão. Aplicações que chamam seu serviço CAP Java precisarão enviar um certificado de cliente válido junto com o token JWT.

Certifique-se de que a propriedade `forwardAuthCertificates: true` está configurada no destino do Approuter que aponta para o seu backend CAP.

Comportamento de Localização e Geração de Código na 3.0

Localização Lazy por Padrão

Recursos EDMX nos endpoints `/$m` agora são localizados de forma lazy, reduzindo o consumo de memória.

O build do CDS gera arquivos EDMX com templates e um `i18n.json`.

Para gerar EDMXs já localizados, use a flag `--opts contentLocalizedEdmx=true` no `cds build`.

Geração de Classes POJO Ajustada

Os valores padrão de `sharedInterfaces` e `uniqueEventContexts` mudaram para `true`, resultando em POJOs incompatíveis com versões anteriores.

Com `uniqueEventContexts: false`

```
// Geração Antiga
public interface DoSomethingContext
extends EventContext {
    // Interface não prefixada
    // Tipo anônimo gerado inline
    Collection<Values> getValues();
    interface Values extends CdsData { /*
    ... */ }
}
```

Com `uniqueEventContexts: true`

```
// Geração Nova
public interface
MyEntityDoSomethingContext extends
EventContext {
    // Interface prefixada com "MyEntity"
    // Usa tipo global MyArray.Item
    Collection<MyArray.Item> getValues();
}
```

O Grande Salto: Migrando para a Versão 2.0

A migração para a versão 2.0 não é um passo incremental. É uma mudança fundamental na stack tecnológica. Esta seção é a mais crítica para projetos em versões legadas.



Spotlight na Versão 2.0: A Mudança Tectônica



Adoção do Spring Boot 3

O CAP Java 2.0 é construído sobre o **Spring Boot 3**. Se sua aplicação utiliza APIs nativas do Spring, consulte o guia de migração oficial do Spring Boot 3.0.



Requisito Mínimo: JDK 17

A mudança para o Spring Boot 3 torna o **Java 17 a versão mínima obrigatória**. Todas as dependências não gerenciadas pelo CAP devem ser compatíveis.



`javax` para `jakarta`

A mudança mais impactante: Spring Boot 3 usa Jakarta EE 10. Todos os pacotes `javax.*` foram renomeados para `jakarta.*`. Isso exigirá mudanças em `imports` em todo o código.

```
// Antes: import javax.servlet.http  
// Depois: import jakarta.servlet.http
```

Preparando-se para o 2.0: Dependências e Comportamento

Versões Mínimas e Recomendadas

Dependência	Mínima	Recomendada
JDK	17	21
Maven	3.5.0	3.9.8
@sap/cds-dk	6	7
Spring Boot	3.0	latest

⚠️ Mudança na Estratégia de `upsert`

O comportamento do upsert mudou fundamentalmente.

Antes (<= 1.27)

Era um delete cascanteado seguido de um insert profundo. Dados parciais redefiniriam os elementos ausentes para seus valores iniciais.

Depois (>= 1.28)

É um update profundo que cria dados se não existirem. Dados parciais deixam os elementos ausentes intactos. A geração de UUID para chaves não ocorre mais.



A propriedade `cds.sql.upsert.strategy` para reverter ao comportamento legado foi removida. Se precisar do comportamento de substituição, implemente um `delete` seguido de um `insert`.

Limpeza de API na Versão 2.0: O Que Foi Removido

Muitas APIs, métodos e propriedades que estavam depreciados na versão 1.x foram removidos na 2.0. Corrija todos os usos de APIs depreciadas antes da migração.



Modificação de CQL

A interface `CqnModifier` foi removida. Utilize a nova classe `Modifier` para modificar statements CQL de forma mais eficiente.



Paginação

As interfaces `CqnLimit` e `Limit` foram removidas. Use os métodos `.limit(top, skip)` diretamente nas queries `Select` e `Expand`.



Propriedades CDS

Uma lista extensa de propriedades `cds.*` foi removida ou substituída.

Exemplo Chave:

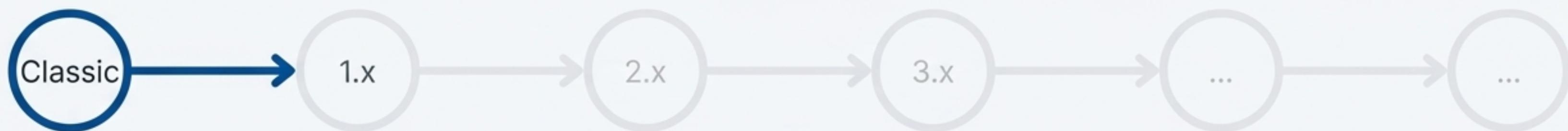
`cds.dataSource.serviceName` →
`cds.dataSource.binding`

Exemplo Chave:

`cds.security.xsuaa.serviceName` →
`cds.security.xsuaa.binding`

Esta é apenas uma amostra. Para uma lista exaustiva de todas as classes e métodos removidos por pacote, consulte o guia de migração oficial.

A Origem: Do Clássico ao CAP Java Moderno



Para projetos que ainda utilizam o CAP Java Runtime clássico, esta seção resume os passos essenciais para migrar para a arquitetura moderna baseada em Spring Boot, abrindo caminho para todas as atualizações futuras.

Guia Rápido: Migrando do CAP Java Classic para 1.x

A migração do runtime clássico para a nova SDK do CAP Java moderniza seu projeto, oferecendo inicialização mais rápida, melhores APIs, integração nativa com Spring e suporte a SQLite para desenvolvimento local.



1 Estrutura do Projeto

Crie um novo projeto CAP Java usando o arquétipo Maven e copie manualmente seus modelos CDS (.cds) e dados (.csv) para as novas pastas db/ e srv/ .



2 Lógica de Negócios (Handlers)

- Anote suas classes de handler com `@Component` e `@ServiceName` .
- Substitua as anotações de evento antigas (ex: `@BeforeCreate`) pelas novas, baseadas em fases (`@Before` , `@On` , `@After`).
- Adapte as assinaturas dos métodos, substituindo tipos como `CreateRequest` por `CdsCreateEventContext` .



3 Configuração de Segurança

A nova SDK autoconfigura a segurança com base nas anotações `@requires` e `@restrict` no seu modelo CDS, simplificando drasticamente a configuração.



4 Acesso a Dados

Substitua o `CDSDataSourceHandler` pelo `CqnService` (para acesso via camada de serviço) ou `PersistenceService` (para acesso direto ao banco).

Esta migração é a base para alinhar seu projeto com a evolução contínua do CAP Java.