# Visão Geral

Imagine que você está organizando uma biblioteca real. Livros sobre diferentes temas podem ter títulos iguais, mas são organizados em seções diferentes, como "Ciência", "História", etc. Da mesma forma, em um projeto de software, você pode ter classes ou funções com o mesmo nome, mas que fazem coisas diferentes ou pertencem a diferentes partes do sistema. É aqui que entram os namespaces, que permitem organizar seu código para evitar conflitos.

No mundo PHP, os namespaces são projetados para resolver dois problemas que os autores de bibliotecas e aplicações encontram ao criar elementos de código reutilizáveis, como classes ou funções:

1. Colisões de nomes entre o código criado e classes/funções/constantes internas do PHP ou classes/funções/constantes de terceiros.
2. Capacidade de criar apelidos (ou encurtar) Nomes\_Extralongos projetados para aliviar o primeiro problema, melhorando a legibilidade do código-fonte.

Os Namespaces do PHP fornecem uma maneira de agrupar classes, interfaces, funções e constantes relacionadas.

# Namespaces

**Como Funciona um Namespace?**

Um namespace é como um sobrenome para classes, funções ou constantes. Ele agrupa esses elementos para evitar confusão. Veja o exemplo

Texto

Descrição gerada automaticamente com confiança médiaAqui temos 2 pastas, as 2 tem o mesma classe:

Se nos formos incluir esses arquivos e depois instanciar um objeto para chamar um método, certamente daria um erro. Mas podemos evitar isso usando nasmespaces.

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Nomeamos os namespaces com o nome da pasta só por motivos didáticos, mas pode ser qualquer nome.

Agora, como podemos instanciar um desses 2 objetos e especificar qual queremos utilizar ?

Texto

Descrição gerada automaticamenteFazemos colocando o nome do nasmespace logo depois de “new”, colocamos entre barras invertidas.

Mas afinal, se em uma aplicação real, é sempre recomendado utilizar nomes diferentes para as classes, por que então utilizar namespaces ?

- Por causa de bibliotecas e frameworks que podem ter suas próprias classes e com nomes iguais.

As bibliotecas e frameworks já utilizam namespaces próprios, por isso que eles não conflitam um com outro. Mas mesmo assim, em aplicações de grande porte, sempre será utilizado os namespaces.

**Outra forma:**

Interface gráfica do usuário, Texto

Descrição gerada automaticamentePodemos também chamar os namespaces que queremos da seguinte forma:

Podemos utilizar o “use + nomeDoNamespace \ Classe” que queremos.

Mas quando utilizamos o “use”, nós informamos que queremos chamar apenas 1 classe, mas isso não faz sentido, visto que queremos incluir diversas classes com nomes iguais, então podemos agrupar esses namespaces e dar “apelidos” para eles, assim atribuindo um nome único que já vai fazer referência namespace que queremos.

Texto

Descrição gerada automaticamentePara isso utilizamos o “as + nomeQueQueremosDarAoNamespace”:

**Nota:** Nomes de namespace não diferenciam maiúsculas de minúsculas.

**Nota:** O nome do namespace PHP e os nomes compostos que começam com esse nome (como PHP\Classes) são reservados para uso interno da linguagem e não devem ser usados no código criado pelo usuário.

Embora qualquer código PHP válido possa estar contido em um namespace, apenas os seguintes tipos de código são afetados por namespaces: classes, interfaces, funções e constantes.

**Regras Importantes para Usar Namespaces em PHP**

1. **Nomes Totalmente Qualificados**:
   * Nomes totalmente qualificados são aqueles que começam com uma barra invertida \, como \MeuProjeto\MinhaClasse.
   * **Importante**: Você **não pode** usar nomes totalmente qualificados ao declarar um namespace. Isso ocorre porque o PHP trata esses nomes como caminhos absolutos de namespaces, enquanto a declaração de namespace deve ser relativa ao arquivo onde está sendo definida.

Tela de celular com aplicativo aberto

Descrição gerada automaticamente

1. **Posição da Declaração de Namespace**:

* A declaração de namespace deve ser a **primeira instrução no script PHP**. Isso significa que não pode haver código HTML, espaço em branco ou qualquer outro conteúdo antes dela, com exceção da construção declare.
* A construção declare pode ser usada antes da declaração de namespace para definir configurações específicas, como a codificação de caracteres do arquivo-fonte.

Texto

Descrição gerada automaticamente

**Dividindo um Namespace em Múltiplos Arquivos**

Texto

Descrição gerada automaticamenteExistem algumas forma de agrupar classes a um namespace, e a forma de fazer isso é simplesmente usando o mesmo nome de namespace em arquivos diferentes.

Dessa forma, você pode ter vários arquivos para o mesmo namespace.

**Agrupando importações com USE**

Texto

Descrição gerada automaticamentePodemos agrupar as classes que queremos ao usar conchetes : “ { } ”

# Sub-namespaces

Os subnamespaces são uma forma de organizar os namespaces parecido de uma parte do sistema para que fique facil de utilizar, assim criando uma espécie de “mini-framework”.

Texto

Descrição gerada automaticamente**Por exemplo:**

Texto

Descrição gerada automaticamente

**O Que São Realmente os Sub-Namespaces?**

Sub-namespaces são uma **extensão do conceito de namespaces em PHP**, criados para organizar o código de maneira hierárquica e lógica. Eles não são apenas uma convenção de nomenclatura, mas um mecanismo essencial que traz uma organização semântica de namespaces para o interpretador do PHP.

**Por Que o PHP Escolheu a Barra Invertida \?**

Mesmo que o conceito de sub-namespaces seja principalmente semântico e organizacional, a barra invertida \ é fundamental para o PHP entender a hierarquia. Sem essa consistência, o PHP não seria capaz de distinguir corretamente entre nomes de classes e a estrutura hierárquica.

A escolha da barra invertida \ como separador de namespaces tem raízes históricas e técnicas:

1. **Diferença de Outras Linguagens**: Linguagens como Java, C#, e C++ usam o ponto . para separar namespaces. No PHP, o ponto já é usado para **concatenação de strings**, então foi necessário escolher outro símbolo.
2. **Compatibilidade com o Sistema de Arquivos**: A barra invertida é um símbolo que lembra a separação de diretórios, especialmente em sistemas Windows. Isso ajuda a manter uma certa consistência, mesmo que o PHP seja multiplataforma.
3. **Autoloading e Carregamento Automático de Classes**: Como mencionado, o uso de sub-namespaces com \ permite que ferramentas como o Composer carreguem classes automaticamente com base na estrutura de diretórios. Sem essa hierarquia, a eficiência do autoloading seria perdida.
4. **Exemplo**: Se você usar sub-namespaces, pode mapear automaticamente classes para arquivos em uma estrutura de pastas correspondente, simplificando a manutenção e o desenvolvimento.
5. **Hierarquia Lógica no Projeto**: Embora pareça uma questão semântica, essa hierarquia tem um impacto funcional em como o código é estruturado, compreendido e mantido.
6. **Integração com Frameworks e Padrões PSR**: Muitos frameworks e bibliotecas esperam que o código siga a especificação PSR-4 de autoloading, que depende de uma estrutura hierárquica de namespaces com \. Sem isso, você não poderia aproveitar ao máximo esses frameworks.

Sub-namespaces são **mais do que apenas organização semântica**. Eles têm implicações técnicas reais que afetam como o código é carregado, estruturado e integrado com ferramentas e frameworks. Portanto, o uso de \ não é arbitrário; ele faz parte de um design cuidadoso para melhorar o desenvolvimento e a manutenção de projetos PHP.

# Definindo vários namespaces no mesmo arquivo

O PHP permite declarar mais de um namespace em um único arquivo. No entanto, há maneiras recomendadas e não recomendadas de fazer isso, dependendo da clareza e da organização do código.

**1. Sintaxe de Combinação Simples (Não Recomendada)**

Texto

Descrição gerada automaticamenteA sintaxe de combinação simples declara vários namespaces no mesmo arquivo sem usar colchetes {}. Embora essa forma seja válida, ela **não é recomendada**, pois pode tornar o código confuso e mais difícil de entender.

**Por Que Não Usar Esta Sintaxe?**

* **Legibilidade**: É difícil perceber onde um namespace termina e o outro começa, especialmente em arquivos grandes.
* **Organização**: Fica mais difícil organizar o código de forma clara e lógica.

**2. Sintaxe com Colchetes (Recomendada)**

Texto

Descrição gerada automaticamenteA sintaxe com colchetes {} é a forma recomendada de declarar vários namespaces no mesmo arquivo. Ela torna a separação entre namespaces mais evidente.

**Vantagens desta Sintaxe**

* **Clareza**: Os colchetes delimitam claramente onde cada namespace começa e termina.
* **Organização**: Facilita a leitura e a manutenção do código.

### Combinar Código Global sem Namespace

Texto

Descrição gerada automaticamenteSe você precisar misturar código com namespaces e código global (código que não pertence a nenhum namespace), você deve usar a sintaxe de colchetes. O código global deve ser encapsulado em um namespace sem nome, como mostrado abaixo:

# Fundamentos

Existem três maneiras de acessar classes em um sistema de namespaces:

**Nome Não Qualificado**: Use apenas o nome sem prefixo. O PHP resolve dentro do namespace atual.

**Nome Qualificado**: Use um caminho relativo ao namespace atual.

Texto

Descrição gerada automaticamente**Nome Totalmente Qualificado**: Use \ antes do nome para especificar o caminho completo.

Texto

Descrição gerada automaticamente

**Analogia:**  
**Nome Não Qualificado**: Imagine estar em uma prateleira específica de uma biblioteca e apenas dizer "quero o livro X". O bibliotecário sabe que você está falando de algo daquela prateleira.

**Nome Qualificado**: Você diz "quero o livro X da seção de literatura". Isso dá mais detalhes, mas ainda não é o endereço completo.

**Nome Totalmente Qualificado**: Você dá o endereço completo: "Quero o livro X, na biblioteca central, seção de literatura".

**Como Acessar Elementos Globais em um Namespace**

Texto

Descrição gerada automaticamenteSe você precisar acessar funções ou classes globais, como strlen() ou Exception, dentro de um namespace, use \ para especificar que está acessando o elemento global.

**Por Que Usar \ para Acessar Elementos Globais?**

Quando você cria um namespace no PHP, tudo o que está dentro dele é **isolado** em seu próprio "espaço de nomes". Isso significa que o PHP procura funções, classes ou constantes primeiro dentro do namespace atual. Se você chamar algo sem especificar que é global, o PHP pensa que você está se referindo a um elemento **definido dentro do namespace**.

# Namespaces e recursos de linguagem dinâmica

Para entendermos o que são os recursos dinâmicos com namespaces, primeiro precisamos entender somente com classes.

Sempre que nós vamos instanciar um objeto, usamos a palavra reservada new e depois o nome da classe, certo ? Então, se antes disso, passarmos o nome da classe para uma variável, e usarmos new + variável que armazena o nome da classe, iremos criar uma forma dinâmica de instânciar objetos.

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamenteAgora, com namespaces, a forma é semelhante, mas invés de passarmos o nome da classe, passamos o endereço dela no namespace:

# Palavra-chave namespace e \_\_NAMESPACE\_\_

\_\_NAMESPACE\_\_ é uma **constante mágica** que o PHP fornece para nos dizer o nome do namespace atual em que o código está sendo executado. O valor de \_\_NAMESPACE\_\_ é uma **string** que contém o nome do namespace. Se o código não estiver em nenhum namespace, ela será uma string vazia.

1. **Dentro de um Namespace**:

Uma imagem contendo objeto, relógio, bola, medidor

Descrição gerada automaticamente

* + Aqui, \_\_NAMESPACE\_\_ retorna "MeuProjeto" porque estamos dentro do namespace MeuProjeto.

1. **No Código Global (Sem Namespace)**:



* + Neste caso, \_\_NAMESPACE\_\_ retorna uma string vazia "" porque o código não está dentro de um namespace.

**----------- Como Usar \_\_NAMESPACE\_\_ em Aplicações Reais -----------**

Interface gráfica do usuário, Texto

Descrição gerada automaticamenteA constante \_\_NAMESPACE\_\_ é útil quando você quer construir nomes de classes de forma dinâmica dentro de um namespace. Por exemplo:

* + **Explicação**: \_\_NAMESPACE\_\_ é usado para construir dinamicamente o nome completo da classe, incluindo o namespace.

# Espaço global

**O Que é o Espaço Global no PHP?**

No PHP, o **espaço global** é onde todas as classes, funções e constantes estão quando você não usa namespaces. Antes de o PHP suportar namespaces, todos os elementos eram definidos no espaço global por padrão.

* **Espaço Global**: É como a "pasta principal" onde tudo é armazenado se você não usa namespaces.

**Como Funciona o Espaço Global com Namespaces?**

Quando você usa namespaces, pode precisar acessar classes, funções ou constantes que ainda estão no espaço global. Para fazer isso, você prefixa o nome com \. Isso diz ao PHP: "Vá para o espaço global".

Texto

Descrição gerada automaticamenteImagine que você está em um namespace, mas quer usar uma função ou classe que pertence ao espaço global:

**Prioridades na Resolução de Nomes em Namespaces**

1. **Classes**: Se você usa um nome de classe sem \ no início, o PHP procura a classe no namespace atual. Se não a encontrar, dá um erro.
2. **Funções e Constantes**: Se o PHP não encontrar uma função ou constante no namespace atual, ele vai procurar no espaço global.