SUMÁRIO

1. SOBRE SERIALIZAÇÃO

Como define a ciência da computação, segundo esse artigo no Wikipedia, no campo de estudo do armazenamento e transmissão de dados, serialização é o processo de "tradução de estruturas de dados ou estado de objeto em um formato que possa ser armazenado e reconstruído posteriormente no mesmo ou em outro ambiente computacional". Assim, por conseguinte, quando a série de dados resultante é relida de acordo com o formato de serialização utilizado, ela pode ser usada para criar um clone da estrutura original.

Esta abordagem é o ponto de partida e de fato o que torna possível a comunicação de sistemas. Por exemplo, para que uma API seja funcional, dados devem ser serializados em um sistema de origem para serem enviados, desserializados e utilizados em um sistema de destino.

Neste sentido surge a demanda, também, por métodos de serialização que permitam a comunicação não apenas entre sistemas semelhantes, mas entre aqueles que são codificados em diferentes linguagens, ou seja, que não são diretamente compatíveis. Para estes problemas, de incompatibilidade entre sistemas que devem trocar dados, foram criados procedimentos de codificações universais, como o JSON e o XML.

2. SERIALIZAÇÃO NO PHP

Como dito anteriormente, a serialização é utilizada para permitir o envio de estruturas de dados, ou dados propriamente ditos, entre sistemas semelhantes ou diferentes, e neste caso surgem métodos universais, como JSON e o XML.

Mas, no ambiente de desenvolvimento PHP, existem métodos de serialização próprios para o contexto, e que atendem, em diversas circunstâncias, melhor as demandas do que a serialização por outras vias.

2.1. A FUNÇÃO NATIVA SERIALIZE

Nativamente, o PHP dispõe de uma função para serialização chamada "serialize", que como descrito na documentação do PHP "é útil para armazenar ou passar valores PHP sem perder seu tipo e estrutura".

Pois bem: portanto, pode haver uma preferência pela serialização utilizando a função nativa do PHP para envio de dados entre os segmentos de uma aplicação PHP. E a que essa preferência possível se deve? Ao fato de que, exatamente como elucida a documentação, a função preserva os aspectos dos valores e da estrutura original, o que é imprescindível em

diversas circunstâncias. Por exemplo, na orientação de objetos com encapsulamento, **os atributos privados de classes quando serializados com esta função nativa, continuam privados e secretos para outras partes da aplicação** – que é exatamente o propósito de um atributo privado.

No entanto, e em nítido contraste, **quando utilizado o JSON** para serializar os dados, como, por exemplo, os valores privados dos atributos de uma classe, para serem utilizados em outra parte da aplicação, como no Index**, a privacidade dos valores é ignorada** e os valores se tornam visíveis.

FONTES

https://www.php.net/manual/pt_BR/function.serialize.php

https://stackoverflow.com/questions/804045/preferred-method-to-store-php-arrays-json-encode-vs-serialize

https://stackoverflow.com/questions/2574728/serialize-or-json-in-php

https://pt.stackoverflow.com/questions/13087/o-que-%C3%A9-serializa%C3%A7%C3%A3o-quando-usar-como-implementar-no-c