# Oshi & Spring

# Revisando alguns itens...

1. Antes de mais nada, faça um git pull no repositório do projeto;
2. Abra o projeto prove3dapp no netbeans;
3. Acesse a classe Config que está dentro do projeto;

   3.1. Bom, o que está classe faz, como o próprio nome já diz, é utilizada para configuração. No caso, o Spring utiliza essa classe para adicionar novas depências ao projeto. Acontece que o Spring não reconhece o oshi como uma classe, porque não faz parte do próprio spring, então temos que setar isto na mão. Quando você olha o metódo oshi(), percebe que ele retorna 'new SystemInfo()', ou seja, quando chamamos está função em qualquer outra classe, nós instânciamos a classe SystemInfo do oshi, e aí o Spring começa a reconhecer como parte do projeto mesmo.
4. Agora abra a Tela em Swing que você precisa alterar;
5. Nesta classe, entre no código fonte e crie um atributo da própria classe desta maneira:

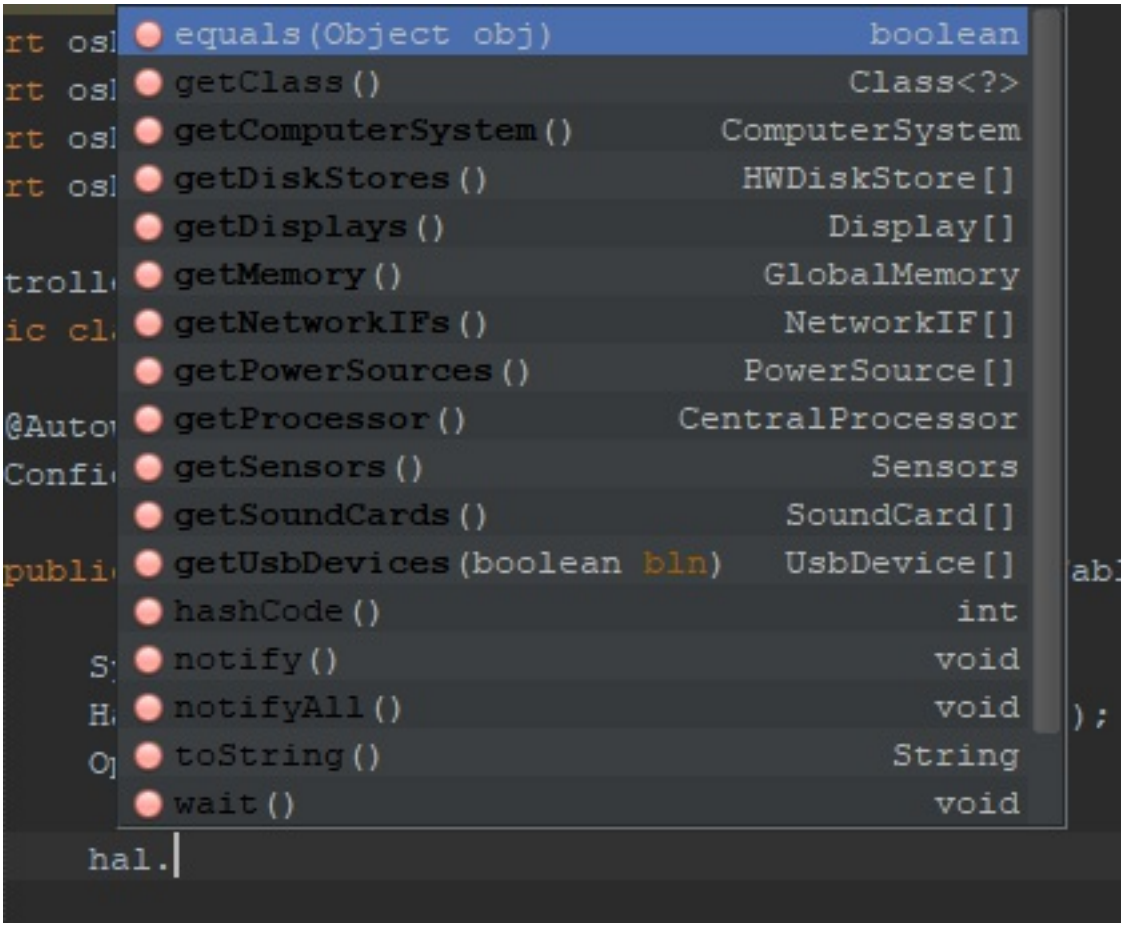
   ```
   @Autowired
   private Config config;
   ```

7. O @Autowired é uma anotação para injetar classes no spring, ele dá a certeza para a gente que o Spring vai reconhecer a classe que estamos criando.

8. Agora para utilizarmos as classes do oshi, fazemos assim(dentro de algum metódo):

```
 // o metódo oshi dentro de config retorna new Syste
mInfo(), ou seja,
 //substitui o código, deixando assim em tempo de ex
ecução: SystemInfo si = new SystemInfo();
SystemInfo si = config.oshi();

 //Deste atributo utilizaremos as duas classes mais
importantes do Oshi, que são:

 //Informações de Hardware
 HardwareAbstractionLayer hal = si.getHardware();

 //Informações do sistema operacional
 OperatingSystem os = si.getOperatingSystem();

```

10. Feito isso, podemos utilizar ‘os’ e ‘hal’ para chamar os metódos de suas respectivas classes.
11. Teste aí, experimente escrever ‘hal.get’, e utilize o atalho CTRL + Espaço. Repare na quantidade de informações que nos é trazida:




12. Tudo o que estiver mais escuro, quer dizer que são metódos exclusivos desta classe.
13. Repare que logo depois dos metódos, a IDE nos mostra o tipo de retorno daquele metódo.

> *Obs: Digamos que você queria armazenar hal.getMemory() dentro de uma variável, você vai saber qual tipo de variável utilizar verificando o tipo de retorno daquele metódo, que no caso é: GlobalMemory. Exemplificando:*

```
        /*Assim funciona porque armazena dentro de um tipo
de retorno válido,
        que no caso é GlobalMemory*/
        GlobalMemory memoria = hal.getMemory();
```

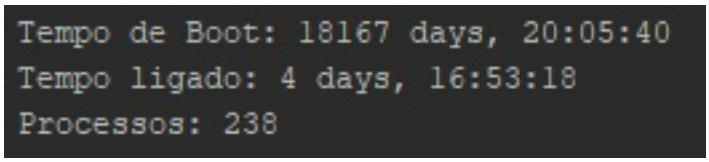
```
        //Você não poderia fazer assim, por exemplo:
        String memoria = hal.getMemory();
        /*Porque você está tentando armazenar GlobalMemory
dentro de uma String,
        e isso é inválido*/

        /* Repare que alguns tipos de retorno possuem colch
etes([]) na frente
        (exemplo: HWDiskStore[]), isto porque o tipo de ret
orno dele é um Array,
        ou seja, irá possuir mais de um valor. Neste caso,
fazemos assim: */
        HWDiskStore[] memoriaDisco = hal.getDiskStores();
```

14. Tendo isto em mente, armazene hal.getMemory() dentro de uma variável memória;
15. Após isto, pule uma linha e escreva memória.get, e utilize as teclas de atalho CTRL + Espaço. Novamente... os metódos em negrito, são aqueles que são exclusivos daquela Classe(No caso, GlobalMemory).
16. Repare no metódo getgetAvailable(), ele tem o tipo de retorno long, então não poderiamos armazená-lo em um GlobalMemory ou uma String(A não ser que formatemos o retorno), por exemplo.
17. Pois bem, armazene este metódo dentro de uma variavel long;
18. Após isto, mostre isto no console do netbeas;

19. Note que veio um número muito grande, isto porque não está formatado e nem convertido. Para isto podemos utilizar um metódo do próprio java, basta fazer o seguinte: `FormatUtil.formatBytes(nomedasuavariavel);` e mostre isto do console do Netbeans, repare na diferença;
20. Essas regras explicadas até aqui se apliam tanto nos metódos de Hardware, como nos de sistema operacional.
21. Mas para eu provar isto para você, vamos fazer um teste, agora chamando uma função de sistema operacional, ou melhor, três funções.
22. Estas funções/metódos são: 1- Mostrar o tempo que o computador está ligado. 2- Tempo de Boot dele. 3- Total de processos rodando no computador.
23. Para o primeiro metódo devemos escrever `os.getSystemUpTime();`, este também retorna em long, e para formatarmos isto, basta utilizar a mesma classe FormatUtil, mas desta vez, utilizando o metodo formatElapsedSecs();
24. Deixando assim `FormatUtil.formatElapsedSecs(os.getSystemUpTime)`, basta mostrar isto no console do seu projeto... e prontinho, você sabe a quanto seu computador está ligado.
25. Para o segundo metódo devemos utilizar o `os.getSystemBootTime()`, e mais uma vez, é um retorno em long, então devemos utilizar literalmente o mesmo metódo de cima e mostrar no console (lembre-se de substituir a função que será formatada);
26. Para o terceiro, e último metódo de hoje, utilizamos o comando `os.getProcessCount()`, este retorna com inteiro, então não

precisamos formatá-lo, mas lembre-se de converte-lo para String ao mostrar no log.

27. Estes são os dados que coletei com o meu PC, utilizandos estes 3 metódos:




> *E sim, por incrível que pareça, meu computador está ligado a 4 dias direto.*

Bom, por hoje é só pe-pe-pessoal. Qualquer dúvida é só chamar, estarei a disposição.