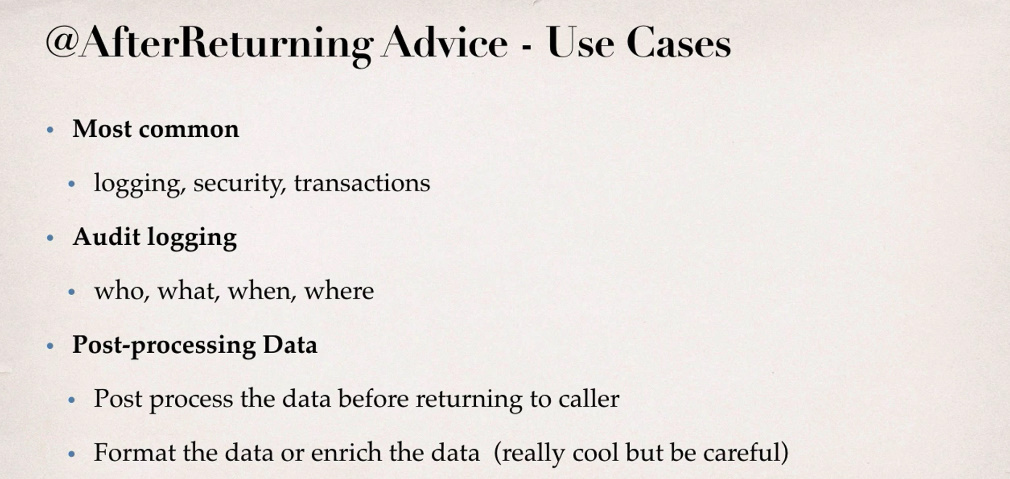
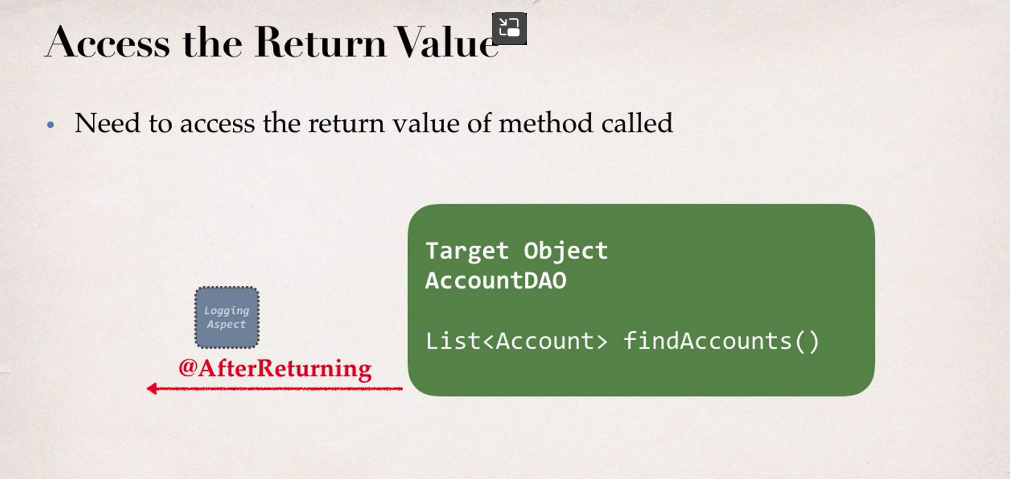
**@AfterReturning**

* Asigura executarea unui Advice doar dupa ce metoda se executa cu succes, adica cand trece de return sau ,daca are void, ajuge la final cu bine. Cu bine se are in vedere fara exceptii. Asta inseamna ca desi metoda va executa return, acel obiect returnat intai va trece prin Aspect, si abea apoi va fi returnat acolo unde a fost apelat.

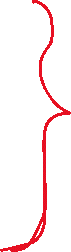


* Pote accesa si ceea ce metoda returneaza.

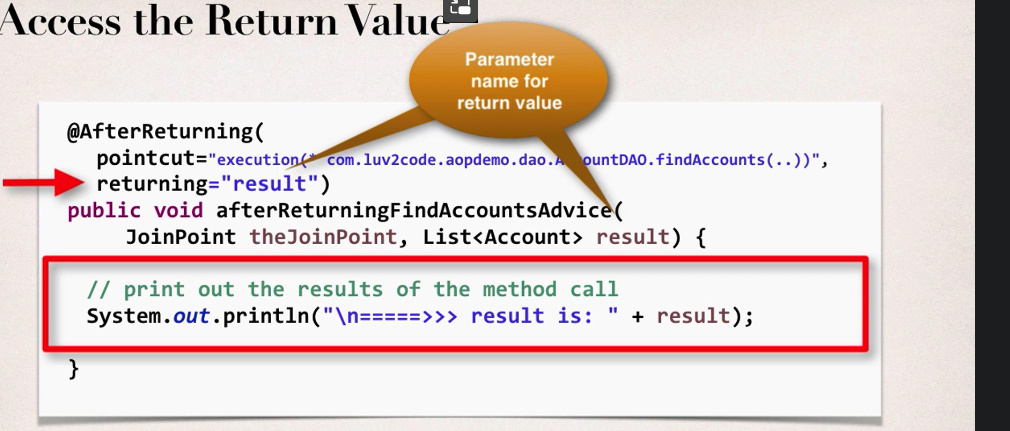


* **AfterReturning nu se mai executa daca apare vreo exceptie, asa cum la return nu se ajunge!**

public List<Account> findAccounts(boolean tripWire){  
 if(tripWire){  
 throw new RuntimeException("Something's wrong");  
 }  
  
  
 accountList.add(new Account("Jhon","Gold"));  
 accountList.add(new Account("Ela","Diamante"));  
 accountList.add(new Account("Bree","Iron"));  
  
 return accountList;  
}



**Implementare**



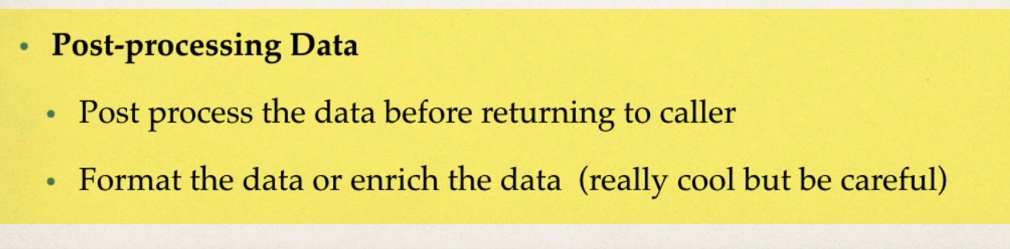
1. Adaugam deja in anotatia @AfterReturning 2 values, una pentru PointCut expression, si alta pentru parametrul ce va stoca ceea ce returneaza metoda.
2. In parametrii metodei din Advice, cream tot un param de tip JoinPoint, ca sa putem accesa parametrii, dar si unul de tipul care stim ca va returna metoda
3. Numele parametrului din returning=”” trebuie sa coincida cu cel din metoda din Advice

Putem obtine datele despre metoda si sub forma de String:

String method = joinPoint.getSignature().toShortString();

**Modificarea returnului**

* @AfterReturning poate sa modifice si datele care urmeaza sa fie returnate. El doar returneaza o referinta la obiectul ce urmeaza sa fie returnat.Asa dar, anotatia data asigura executia la Advice inainte de return.



* Dupa ce se executa acest Advice, adica dupa @AfterReturning, metoda care a trecut prin el va returna deja ce va fi in param din returning=””, adica obiectul sau din return va fi modificat de Advice si apoi va fi returnat.