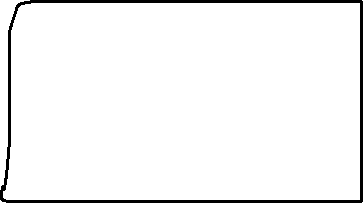
1. Around
2. Before
3. After
4. AfterThrowing
5. AfterReturning

Atentie! Incepand cu Spring 5.2.7, ordinea e alta:

1. Around
2. Before
3. AfterThrowing
4. AfterReturning
5. After

**Ordinea de executie(Spring mai vechi)**

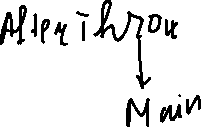
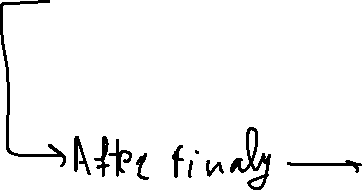
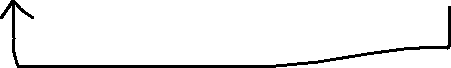
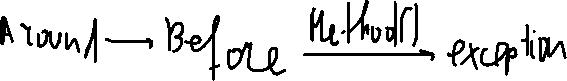
Ordinea de executie e mai dificila cand avem toate aceste 5 Advices, caci anume generarea unei exceptii e mai neclara. **Around mereu are prioritate!**



1. Intai mereu se executa Around, caci metoda nu ruleaza pana around nu o va executa.
2. Acum, metoda se executa in around, cand alegem noi, sa zicem ca intr-un bloc try catch. Metoda ruleaza, deci logic ca urmeaza Before.
3. Acum, daca apare o exceptie, @AfterReturning nu o sa fie executat pana cand @Around nu returneaza ceva, daca o face, dar ideea e ca dupa ce exceptia apare, ea nu se duce la @AftherThrowing!!! Ea revine la @Around, care se va asigura sa execute totul ca deobicei, adica executa ce are ea in catch,apoi After Finally, apoi ea o arunca, si abea atunci ea se va duce prin @AfterThrowing si va ajunge la Main. Asa cum merge ordinea. After inaintea lui AfterThrowing. Daca @Around nu arunca exceptia mai departe, dar returneaza ceva, se va executa @AfterReturning

**Deci, daca apare o exceptie, ordinea de executie va fi asa:**

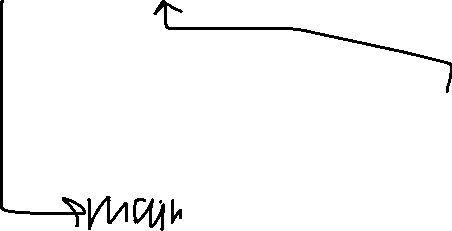
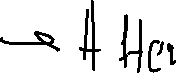
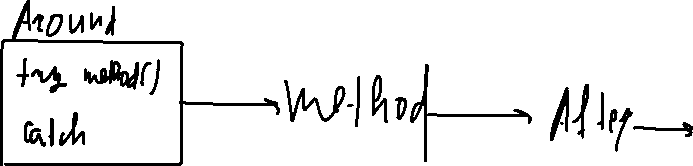
1. Around
2. Before
3. Around iar
4. After Finally
5. AfterThrowing



Daca @Around nu arunca exceptia mai departe, ea nici nu ajunge la @AfterThrowing.

In Spring 5.2.7+ AfterFinally e dupa AfterThrown!!!

Daca nu apare vreo exceptie:



**Ordinea generalizata:**

* Daca apare exception, si Around o arunca mai departe:

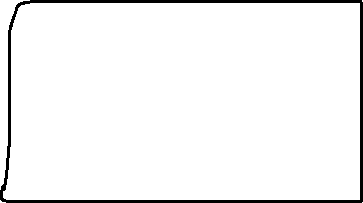
1. Around
2. Before
3. Metoda
4. Around
5. After finally
6. After Throwing
7. Main

* Daca apare exception, dar Around nu o arunca mai departe, ci returneaza ceva setat de el:

1. Around
2. Before
3. Metoda
4. Around
5. After Finally
6. After Returning
7. After Throwing
8. Main

**Spring Nou**

In spring nou, lucrurile sunt mai logice. Daca apare vreo exceptie, atunci totul e mut mai logic:



1. Intai mereu se executa Around, caci metoda nu ruleaza pana around nu o va executa.

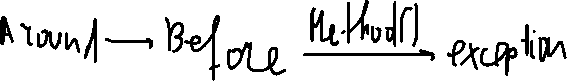
2. Acum, metoda se executa in around, cand alegem noi, sa zicem ca intr-un bloc

try catch. Metoda ruleaza, deci logic ca urmeaza Before.

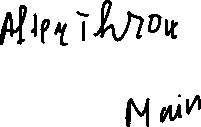
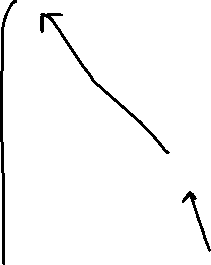
3. Acum, daca apare o exceptie, @AfterReturning nu o sa fie executat, dar ideea e ca dupa ce exceptia apare, ea **se duce** la @AftherThrowing nu la @Around!!! Dupa, se duce la @After Finaly si in cele din urma, EXCEPTIA revine la @Around, care se va asigura sa execute ce are ea in catch, apoi ea o arunca(daca asa noi decidem), si va ajunge la Main. Daca nu arunca exceptia ci returneaza ceva, nu se mai executa @AfterReturning ca in versiunea veche de Spring, caci nu are sens, asa cum @AfterReturning nu e pentru ea.

**Deci, daca apare o exceptie, ordinea de executie va fi asa:**

1. Around
2. Before
3. AfterThrowing
4. After finally
5. Around iar







@Around face apoi ce vrea cu exceptia, o arunca la Main sau nu, ea oricum deja a **Trecut prin @Throwing si @After final**

**Ordine Generalizata**

* Daca apare exception, si Around o arunca mai departe:

1. Around
2. Before
3. Metoda
4. After Throwing
5. After Finally
6. Around
7. Main

* Daca apare exception, dar Around nu o arunca mai departe, ci returneaz ceva setat de el:

1. Around
2. Before
3. Metoda
4. After Throwing
5. After Finally
6. Around
7. Main

**Vezi in projects!Modifica versiunile la spring-context in pom.xml pentru a vedea diferite ordini de executie in diferite versiuni!**