Spring Professional Exam Tutorial v5.0

Question 30

Question 30 - What is Spring Expression Language (SpEL for short)?

Spring Expression Language (SpEL) is an expression language that allows you to query and manipulate objects graphs during the runtime. SpEL is used in different products across Spring portfolio.

SpEL can be used independently with usage of `ExpressionParser` and `EvaluationContext` or can be used on top of fields, method parameters, constructor arguments via `@Value` annotation `@Value("#{ … }")`.

SpEL supported features:

- Literal expressions
- Boolean and relational operators
- Regular expressions
- Class expressions
- Accessing properties, arrays, lists, and maps
- Method invocation
- Relational operators
- Assignment
- Calling constructors
- Bean references
- Array construction
- Inline lists
- Inline maps
- Ternary operator
- Variables
- User-defined functions
- Collection projection
- Collection selection
- Templated expressions

Language Reference - https://docs.spring.io/spring/docs/current/spring-framework-reference/core.html#expressions-language-ref

Question 30 - What is Spring Expression Language (SpEL for short)? (cont.)

SpEL expressions are usually interpreted during runtime, this is good since it provides a lot of dynamic features. However, in some cases performance is more important then number of features available, for those cases Spring Framework 4.1 introduced possibility to compile expressions.

Compilation of Spring Expression is done by creating real Java Class that embodies expression, this results in much faster Expression Evaluation. Because during compilation, reference types of properties are unknown, Compiled Expressions are best to use when types of referenced types are not changing.

Compiler is turned off by default, you can turn it on by:

- Parser Configuration
- System Property - `spring.expression.compiler.mode`

Compiler can operation in three modes (`SpelCompilerMode`)

- Off – default
- Immediate – compile upon first expression interpretation
- Mixed – compiler dynamically switched between interpreted and compiled mode, compiled form is generated after few invocations, if exception will be thrown during compiled form evaluation, then fallback to interpreted form will occur, and then after few invocation compiler will switch to compiled mode again

Compiled Mode does not support following expressions:

- Expressions involving assignment
- Expressions relying on the conversion service
- Expressions using custom resolvers or accessors
- Expressions using selection or projection