/ JAVA DB SCHOOL
Annotations

Java annotations

- Tags that represent metadata
- Can be attached to classes, interfaces, methods or fields

/ Built-in annotations

Built-in annotations

- @Override – assures the subclass method is overriding the parent class method
- @SuppressedWarnings – used to supress warnings issued by the compiler
- @Deprecated – marks a method as deprecated so the compiler prints a warning informing the user the method may be removed in the future version

/ Custom annotations

Custom annotations

- Can be marker, single-value or multi-value

- Restrictions:
  - Annotation should not have any throws clauses
  - Return values: primitive data types, String, Class, enum or array of these data types
  - Should not have any parameters
  - Can assign default values

Marker annotation

- Has no method

@interface MyAnnotation{}

- Example of marker annotations: @Override, @Deprecated

Single-value annotation

- Has one method
- Can be provided a default value

```
@interface MyAnnotation {
    int value() default 0;
}
```

- Usage

```
@MyAnnotation(value=10)
```

Multi-value annotation

- Definition

```
@interface MyAnnotation {
    int value1() default 1;
    String value2() default "";
    String value3() default "xyz";
}
```

- Usage

```
@MyAnnotation(value1=10,value2=“abc",value3=“q1w2e3")
```

Built-in annotations to use in custom annotations

- @Target
  - Used to specify at which type the annotation is used
  - java.lang.annotation.ElementType – TYPE (class, interface or enumeration), FILED, METHOD, CONSTRUCTOR, PARAMETER etc.
- @Retention
  - Used to specify at which level the annotation is available
  - RetentionPolicy – SOURCE (only for the source code, it is discarded during compilation), CLASS (available for the Java compiler, but not for the JVM), RUNTIME (available for the Java compiler and the JVM)
- @Inherited – marks the annotation to be inherited in subclasses
- @Documented – marks the annotation to be included in the documentation

/ Practice, practice, practice

Practice

- Implement ActiveRecordEntity to annotate Customer class
- The annotation should receive the table name and the primary key name
- Implement an ActiveRecord class that uses the information from ActiveRecordEntity annotation to select / update / insert / delete any type of entry (customers, products, orders etc.)

/ Q&A


MOBILE/ACADEMY