Certainly! Here are two code examples demonstrating the use of custom Java annotations:

Example 1: Creating a Custom Annotation

In this example, we'll create a custom annotation called `MethodInfo` and use it to annotate a method. We'll also use reflection to access and process the annotation at runtime.

```java
import java.lang.annotation.*;

@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
@interface MethodInfo {
    String author() default "John Doe";
    String date();
    int revision() default 1;
    String comments();
}

public class AnnotationExample {

    @MethodInfo(author = "Alice", date = "2023-10-01", comments = "Initial version")
    public void myMethod() {
        // Method implementation
    }

    public static void main(String[] args) {
        AnnotationExample example = new AnnotationExample();

        // Use reflection to access and process the annotation
        Class<?> clazz = example.getClass();
        try {
            Method method = clazz.getMethod("myMethod");
            MethodInfo methodInfo = method.getAnnotation(MethodInfo.class);

            System.out.println("Author: " + methodInfo.author());
            System.out.println("Date: " + methodInfo.date());
            System.out.println("Comments: " + methodInfo.comments());
        } catch (NoSuchMethodException e) {
            e.printStackTrace();
        }
    }
}
```

In this example:

- We define a custom annotation `@MethodInfo` with attributes like `author`, `date`, `revision`, and `comments`.
- We annotate the `myMethod` method with `@MethodInfo`.
- In the `main` method, we use reflection to access the `MethodInfo` annotation on the `myMethod` and print its values.

Example 2: Using Annotations in a Framework (Spring Framework)

The Spring Framework extensively uses annotations for configuration. Here's an example of a Spring component class annotated with `@Component`:

```java
import org.springframework.stereotype.Component;

@Component
public class MyService {
    public String getMessage() {
        return "Hello, from MyService!";
    }
}
```

In this example:

- The `MyService` class is annotated with `@Component`. This annotation tells Spring to treat `MyService` as a Spring bean and manage its lifecycle.
- Spring will automatically create an instance of `MyService` and make it available for dependency injection or other Spring features.

These examples illustrate how you can create and use custom annotations in Java and how annotations are widely used in frameworks like Spring to simplify configuration and enhance the functionality of Java applications.