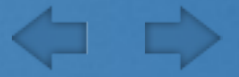




# ConfigureMe

Composite attributes



# Flat POJO

## flatBean.json

```
{  
  foo: "string",  
  bar: true  
}
```



## FlatBean class

```
@ConfigureMe(name="flatBean")  
public class FlatBean {  
    @Configure  
    public String foo;  
  
    @Configure  
    public Boolean bar;  
  
    @Override  
    public String toString() {  
        return "FlatPojo [foo=" + foo  
            + ", bar=" + bar + "];"  
    }  
}
```



## State

```
FlatPojo [foo=string, bar=true]
```



# Nested Objects

compositeBean.json

```
{
  foo: "string",
  inner: {
    bar: true,
    baz: 42
  }
}
```



CompositeBean class

```
@ConfigureMe(name="CompositeBean")
public class CompositeBean {
    @Configure
    public String foo;

    @Configure
    public InnerBean inner;

    public static class InnerBean {
        @Configure
        public boolean bar;

        @Configure
        public int baz;
    }
}
```



# Nested Objects

compositeBean.json

```
{
  foo: "string",
  inner: {
    bar: true,
    baz: 42
  }
}
```



CompositeBean class

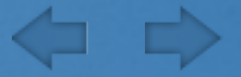
```
@ConfigureMe(name="CompositeBean")
public class CompositeBean {
    @Configure
    public String foo;

    @Configure
    public InnerBean inner;

    public static class InnerBean {
        @Configure
        public boolean bar;

        @Configure
        public int baz;
    }
}
```

Nested JSON objects denotes conditional inclusions within flat configuration



# Nested Objects

compositeBean.json

```
{
  foo: "string",
  @inner: {
    bar: true,
    baz: 42
  }
}
```

CompositeBean class

```
@ConfigureMe(name="CompositeBean")
public class CompositeBean {
  @Configure
  public String foo;

  @Configure
  public InnerBean inner;

  public static class InnerBean {
    @Configure
    public boolean bar;

    @Configure
    public int baz;
  }
}
```



# Nested Objects

## compositeBean.json

```
{
  foo: "string",
  @inner: {
    bar: true,
    baz: 42
  }
}
```



## CompositeBean class

```
@ConfigureMe(name="CompositeBean")
public class CompositeBean {
  @Configure
  public String foo;

  @Configure
  public InnerBean inner;

  public static class InnerBean {
    @Configure
    public boolean bar;

    @Configure
    public int baz;
  }
}
```



## State

```
CompositeBean [foo=string, inner=InnerBean [bar=true, baz=42]]
```



# Arbitrary nesting depth

## a.json

```
{
  foo: "string",
  @innerB: [
    {
      bar: true,
      @innerC: {
        baz: 42.5,
        foobar: [1, 2, 3, 4]
      }
    },
    {
      bar: false,
      @innerC: {
        baz: -42.5,
        foobar: [3, 2, 1]
      }
    }
  ]
}
```

## Top level class

```
@ConfigureMe(name="a", allfields=true)
public class A {
    public String foo;
    public B[] innerB;
}
```

## Inner classes

```
public class B {
    @Configure
    public boolean bar;
    @Configure
    public C innerC;
}

public class C {
    @Configure
    public float baz;
    @Configure
    public int[] foobar;
}
```



# Arbitrary nesting depth

a.json

```
{
  foo: "string",
  @innerB: [
    {
      bar: true,
      @innerC: {
        baz: 42.5,
        foobar: [1, 2, 3, 4]
      }
    },
    {
      bar: false,
      @innerC: {
        baz: -42.5,
        foobar: [3, 2, 1]
      }
    }
  ]
}
```

Top level class

```
@ConfigureMe(name="a", allfields=true)
public class A {
  public String foo;
  public B[] innerB;
}
```

**Annotated class**

Inner classes

```
public class B {
  @Configure
  public boolean bar;
  @Configure
  public C innerC;
}

public class C {
  @Configure
  public float baz;
  @Configure
  public int[] foobar;
}
```

**Not annotated classes**





# Arbitrary nesting depth

a.json

```
{
  foo: "string",
  @innerB: [
    {
      bar: true,
      @innerC: {
        baz: 42.5,
        foobar: [1, 2, 3, 4]
      }
    },
    {
      bar: false,
      @innerC: {
        baz: -42.5,
        foobar: [3, 2, 1]
      }
    }
  ]
}
```

Top level class

```
@ConfigureMe(name="a", allfields=true)
public class A {
  public String foo;
  public B[] innerB;
}
```

← Implicitly configured fields

Inner classes

```
public class B {
  @Configure
  public boolean bar;
  @Configure
  public C innerC;
}

public class C {
  @Configure
  public float baz;
  @Configure
  public int[] foobar;
}
```

← Explicitly configured fields



# Method annotations

d.json

```
{
  @inner: {
    attr: 42
  }
}
```



D class

```
@ConfigureMe(name="d")
public class D {
  @Configure
  public E inner;

  public static class E {
    @Configure
    public int attr;

    @BeforeConfiguration
    public void before() {
      System.out.println("before: " + attr);
    }

    @AfterConfiguration
    public void after() {
      System.out.println("after: " + attr);
    }
  }
}
```

Output

```
before: 0
after: 42
```





# String representation

f.json

```
{
  @inner: {
    attr: 42
  }
}
```



F class

```
@ConfigureMe(name="f")
public class F {
    public G inner;

    public static class G {
        @Configure
        public int attr;
    }

    @SetIf(condition=SetIfCondition.matches,
           value="inner")
    public void after(String name, String value) {
        System.out.println(value);
    }
}
```

Output

```
{"attr": "42"}
```





# Value resolution

f.json

```
{
  @inner: {
    attr: 42
  }
}
```



F class

```
@ConfigureMe(name="f")
public class F {
  public G inner;

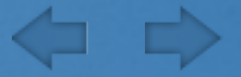
  public static class G {
    @Configure
    public int attr;
  }

  @SetIf(condition=SetIfCondition.matches,
    value="inner")
  public void after(String name, G value) {
    System.out.println(value.attr);
  }
}
```



Output

```
42
```



# Raw composite values

f.json

```
{
  @inner: {
    attr: 42
  }
}
```



F class

```
@ConfigureMe(name="f")
public class F {
  public G inner;

  public static class G {
    @Configure
    public int attr;
  }

  @SetIf(condition=SetIfCondition.matches,
    value="inner")
  public void after(String name, Object value) {
    System.out.println(value.getClass());
    System.out.println(value);
  }
}
```



Output

```
class java.util.HashMap
{attr=42}
```



# Raw array values

f.json

```
{
  @inner: [
    {attr: 1},
    {attr: 2}
  ]
}
```



F class

```
@ConfigureMe(name="f")
public class F {
    public G[] inner;

    public static class G {
        @Configure
        public int attr;
    }

    @SetIf(condition=SetIfCondition.matches,
        value="inner")
    public void after(String name, Object value) {
        System.out.println(value.getClass());
        System.out.println(value);
    }
}
```



Output

```
class java.util.ArrayList
[{"attr":1}, {"attr":2}]
```