Understanding Dependency Injection

Bogdan Sucaciu
Tech Lead

@bsucaciu bsucaciu.com

Inversion of Control

Oven.java

```
class Oven {

  public void turnOn() { … }

}
```

Inversion of Control

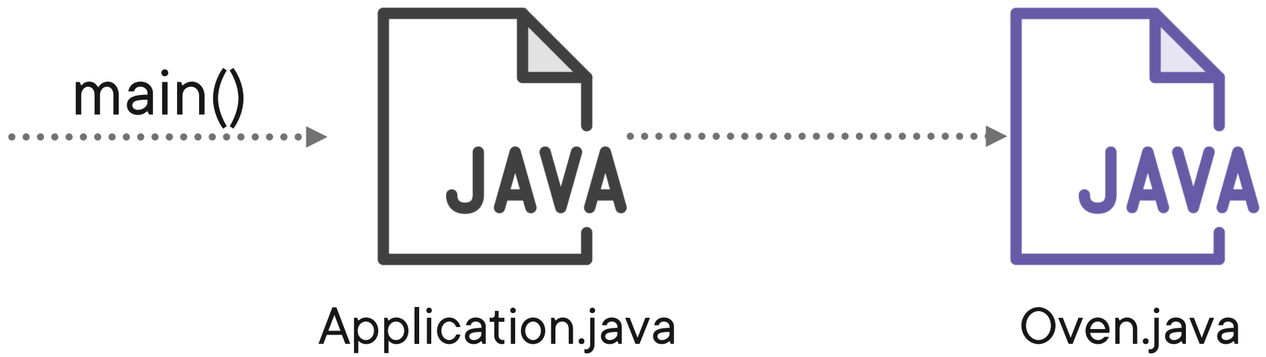
Oven.java

```
class Oven {

  public void turnOn() { … }

}
```

Application.java

```
class Application {

  public static void main(String[] args){
    Oven oven = new Oven();
    oven.turnOn();
  }
}
```

Inversion of Control



Inversion of Control

Oven.java

```
class Lights { … }
class Grill { … }

class Oven {

  private Lights lights;
  private Grill grill;

  public Oven(Lights lights, Grill grill) {
    this.lights = lights;
    this.grill = grill;
  }

  public void turnOn() {
    lights.turnOn();
    grill.turnOn();
  }
}
```

Inversion of Control

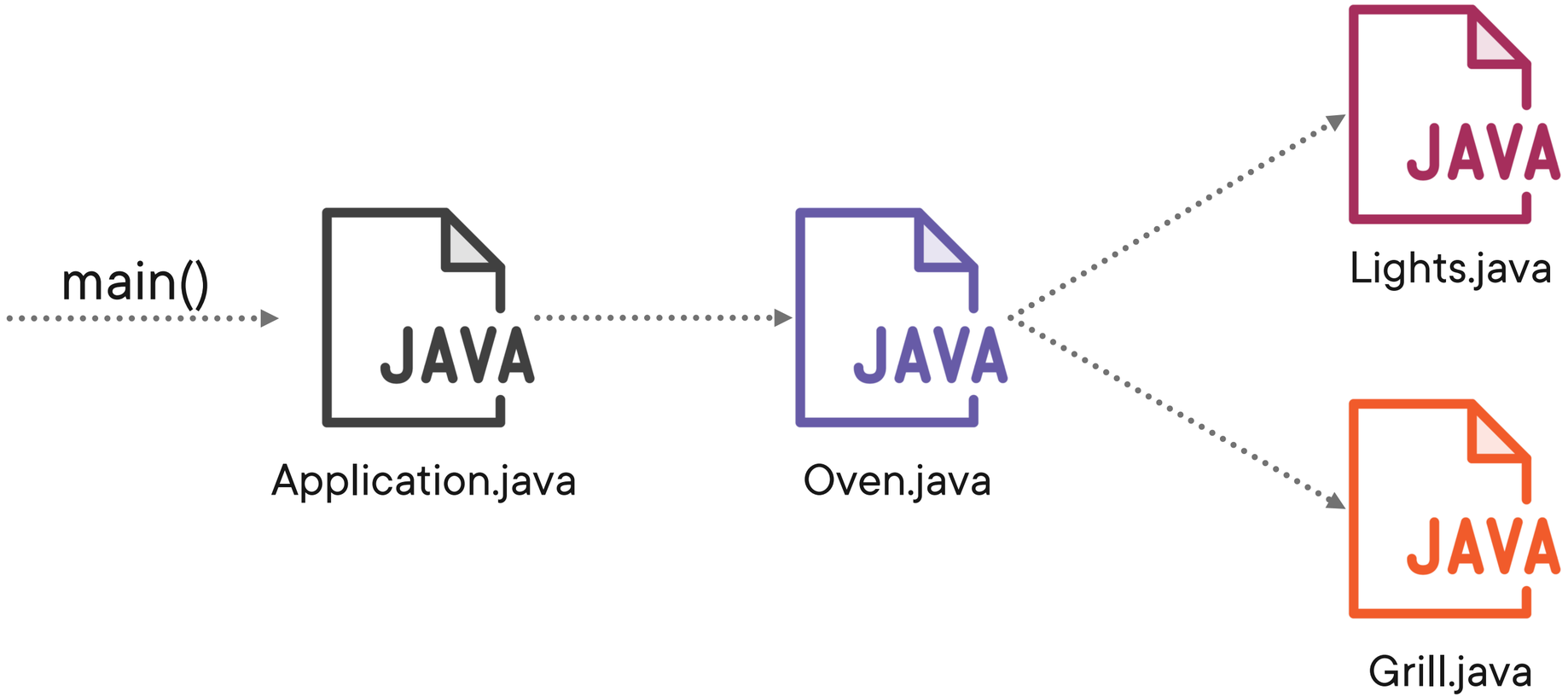
Oven.java

```
class Lights { … }
class Grill { … }

class Oven {

  private Lights lights;
  private Grill grill;

  public Oven(Lights lights, Grill grill) {
    this.lights = lights;
    this.grill = grill;
  }

  public void turnOn() {
    lights.turnOn();
    grill.turnOn();
  }
}
```

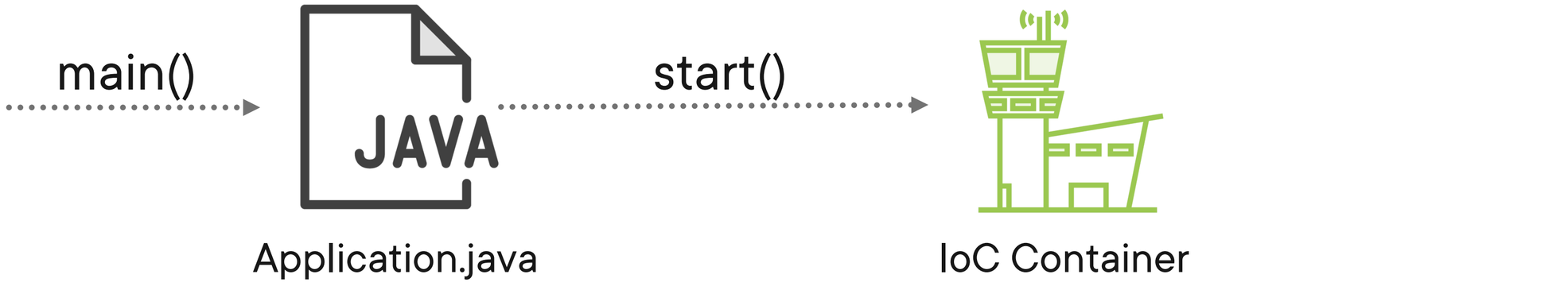
Application.java

```
class Application {

  public static void main(String[] args){
    Lights lights = new Lights();
    Grill grill = new Grill();

    Oven oven = new Oven(lights, oven);
    oven.turnOn();
  }
}
```

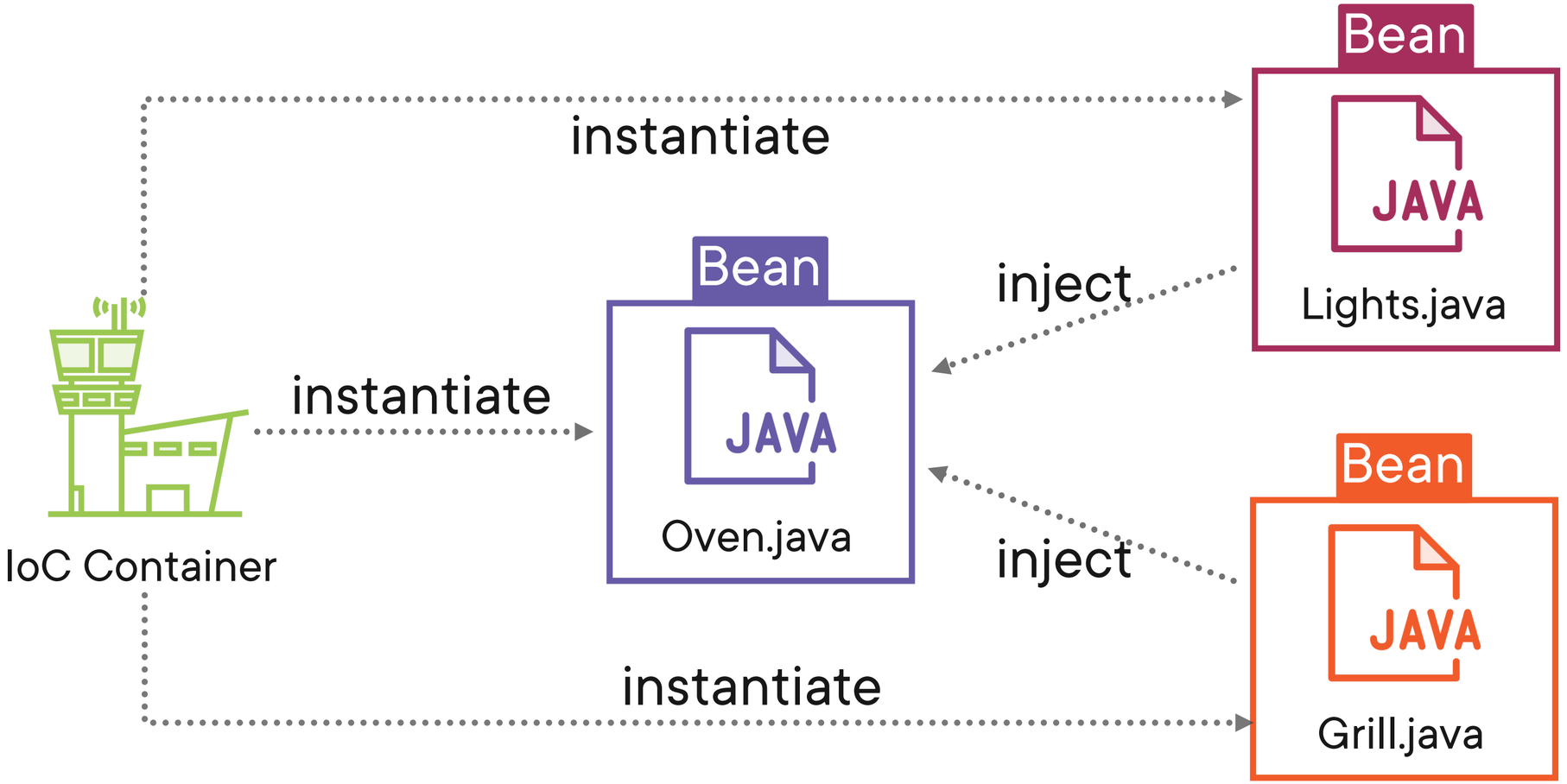
Inversion of Control



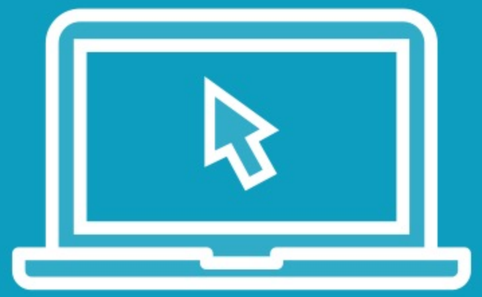
Inversion of Control



Dependency Injection



Demo



Dependency Injection

- Constructor injection
- Setter injection
- Field injection

Bean Qualifiers

Bean Qualifiers

Modes.java

```
interface CookingMode { ... }

class GrillCookingMode implements CookingMode { ... }
class FanCookingMode implements CookingMode { ... }
class DefrostingCookingMode implements CookingMode { ... }

class Oven {

  private CookingMode cookingMode;

  @Inject
  public Oven(CookingMode cookingMode) {
    this.cookingMode = cookingMode;
  }
}
```

Qualify by Type

Modes.java

```
interface CookingMode { ... }

class GrillCookingMode implements CookingMode { ... }
class FanCookingMode implements CookingMode { ... }
class DefrostingCookingMode implements CookingMode { ... }

class Oven {

  private CookingMode cookingMode;

  @Inject
  public Oven(GrillCookingMode cookingMode) {
    this.cookingMode = cookingMode;
  }
}
```

Qualify by Name

Modes.java

```
interface CookingMode { ... }

class GrillCookingMode implements CookingMode { ... }
class FanCookingMode implements CookingMode { ... }
class DefrostingCookingMode implements CookingMode { ... }

class Oven {

  private CookingMode cookingMode;

  @Inject
  public Oven(@Named(“Fan”) CookingMode cookingMode) {
    this.cookingMode = cookingMode;
  }
}
```

Qualify by Name

Modes.java

```
interface CookingMode { ... }

class GrillCookingMode implements CookingMode { ... }
class FanCookingMode implements CookingMode { ... }
class DefrostingCookingMode implements CookingMode { ... }

class Oven {

  private CookingMode cookingMode;

  @Inject
  public Oven(@Named(“Fan”) CookingMode cookingMode) {
    this.cookingMode = cookingMode;
  }
}
```

Qualify by Name

Modes.java

```
interface CookingMode { ... }

class GrillCookingMode implements CookingMode { ... }
class FanCookingMode implements CookingMode { ... }
class DefrostingCookingMode implements CookingMode { ... }

class Oven {

  private CookingMode cookingMode;

  @Inject
  public Oven(@Named("Fan") CookingMode cookingMode) {
    this.cookingMode = cookingMode;
  }
}
```

Qualify by Name

Modes.java

```
interface CookingMode { ... }

class GrillCookingMode implements CookingMode { ... }
class FanCookingMode implements CookingMode { ... }
class DefrostingCookingMode implements CookingMode { ... }

class Oven {

  private CookingMode cookingMode;

  @Inject
  public Oven(@Named("Fan") CookingMode cookingMode) {
    this.cookingMode = cookingMode;
  }
}
```

Qualifying by Annotation

Oven.java

```
interface CookingMode { … }

class GrillCookingMode implements CookingMode { … }
class FanCookingMode implements CookingMode { … }
class DefrostingCookingMode implements CookingMode { … }

class Oven {

  private CookingMode cookingMode;

  @Inject
  public Oven(@Defrosting CookingMode cookingMode) {
    this.cookingMode = cookingMode;
  }
}
```

Defrosting.java

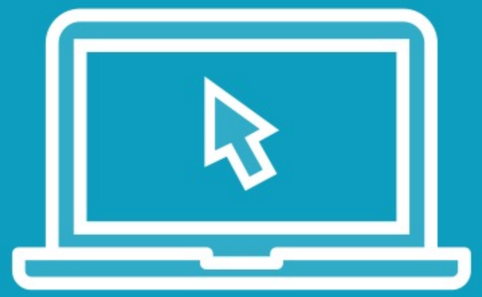
```
@Qualifier
@Retention(RUNTIME)
public @interface Defrosting {
}
```

Qualify by Priority

@Primary

@Secondary

Demo



Bean Qualifiers

- by Name
- by Annotation
- by Priority

Bean Conditionals and Replacements

Bean Conditionals

Oven.java

```
@Singleton
@Requires(property="safety.switch.enabled", value="true")
class Oven {

  private CookingMode cookingMode;

  @Inject
  public Oven(CookingMode cookingMode) {
    this.cookingMode = cookingMode;
  }
}
```

Bean Conditionals

Requirement	Present	Missing
Classes	classes=	missingClasses=
Beans	beans=	missingBeans=
Environment	env=	notEnv=
Configuration Package	configuration=	missingConfiguration=
Property	property=	missingProperty=
OS	os=	notOs=
Resource	resources=	
SDK	sdk=	
Entities	entities=	

Bean Replacement

FanCookingMode.java

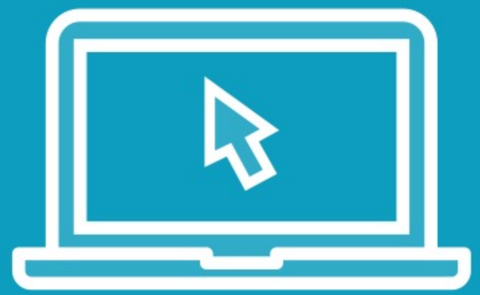
```
@Singleton
class FanCookingMode implements CookingMode { ... }
```

Bean Replacement

FanCookingMode.java

```
@Singleton
class FanCookingMode implements CookingMode { ... }

@Singleton
@Replaces(FanCookingMode.class)
class UpdatedFanCookingMode implements CookingMode { ... }
```

Demo

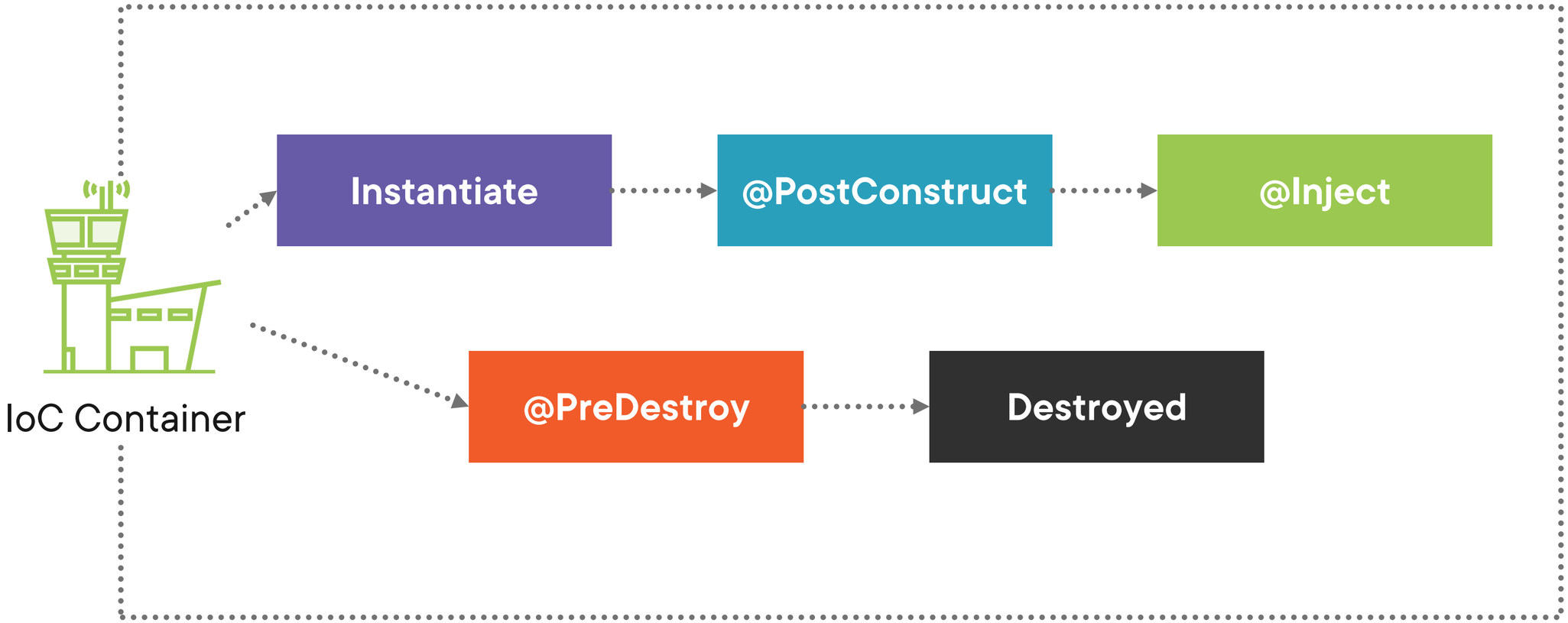


Bean Conditionals

Replace existing beans

Bean Lifecycle and Scopes

Bean Lifecycle



Bean Scopes

@Singleton

@Context

@Prototype

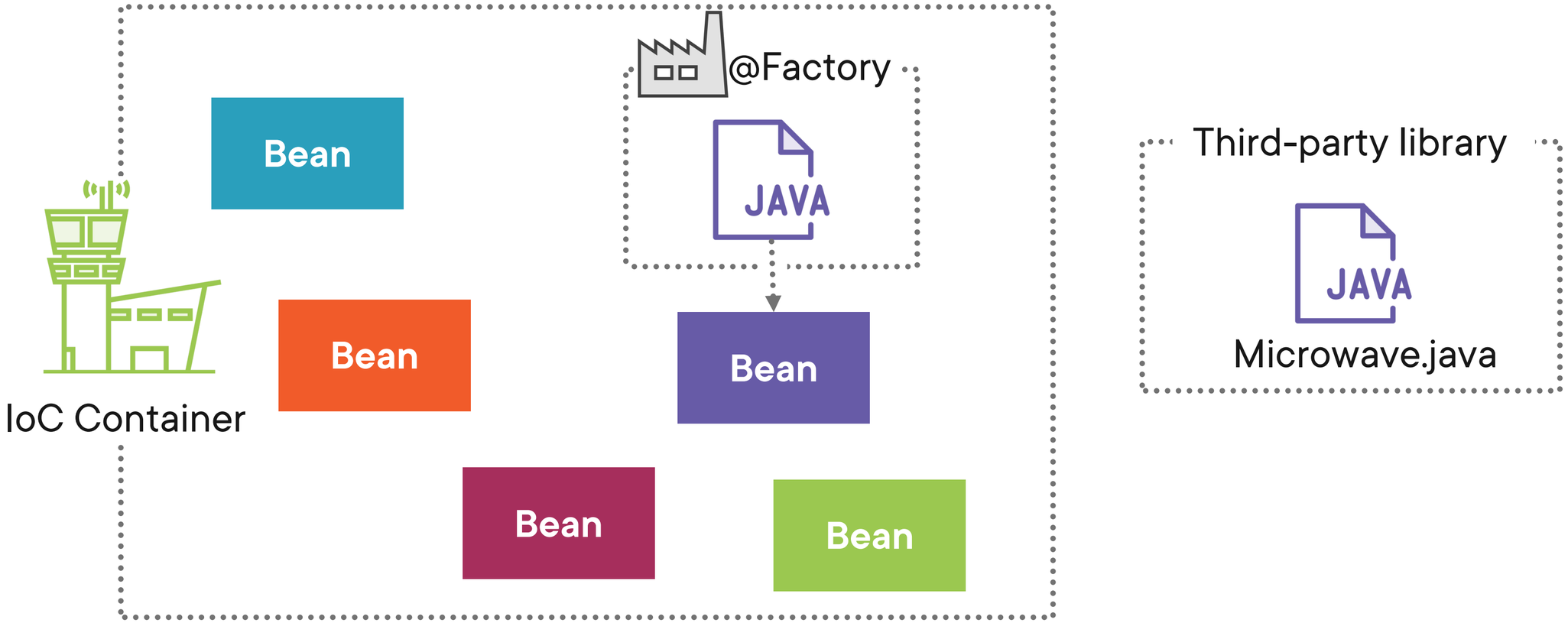
@ThreadLocal

@Infrastructure

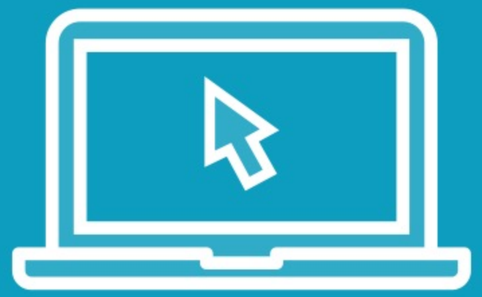
@RequestScope

@Refreshable

Bean Factory



Demo



Bean Scopes

Bean Factories

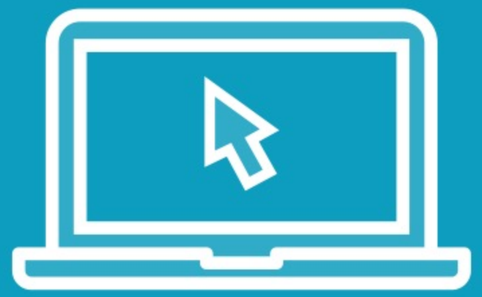
Bean Introspection

Bean Introspection

Meal.java

```
@Introspected
class Meal {

    private String name;
    private Integer secondsToCook;
    private Integer power;

     //Constructors
     //Getters
     //Setters
}
```

Demo



Bean Introspection

Up Next:
Configuring Micronaut Applications