

Reducing Code with Custom Auto Configurations



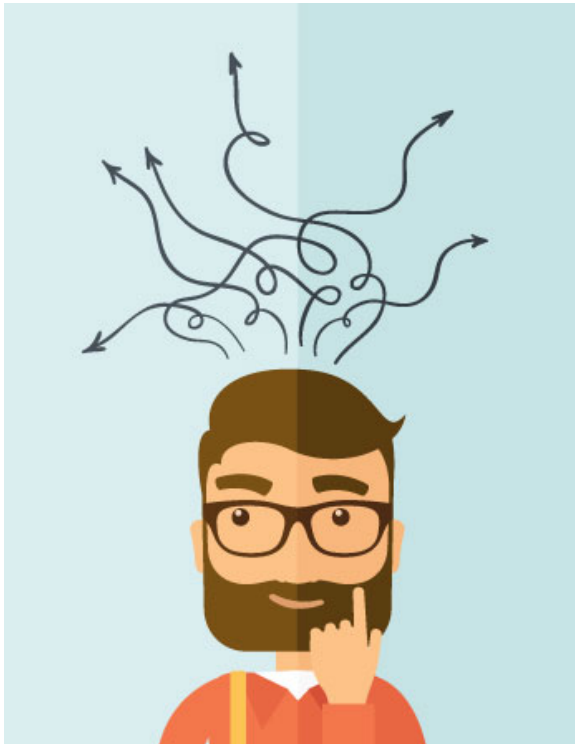
Dustin Schultz

SOFTWARE ENGINEER

@schultzdustin <http://dustin.schultz.io/> dustin@schultz.io

Smarter

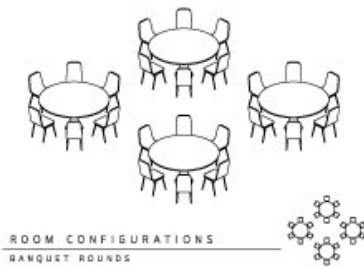
- **@EnableAutoConfiguration**
 - Review
 - Tuning
 - Demystifying
 - Writing our own



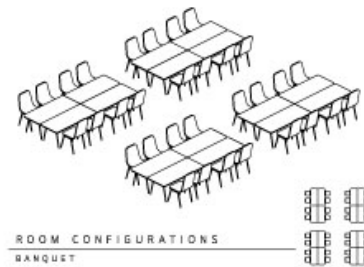
Spring Framework is highly configurable

- **Web?**
- **Front end?**
- **Data access?**
- **Security?**

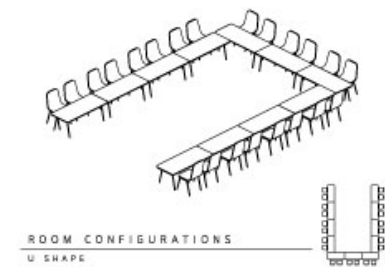
Decisions. So many decisions.



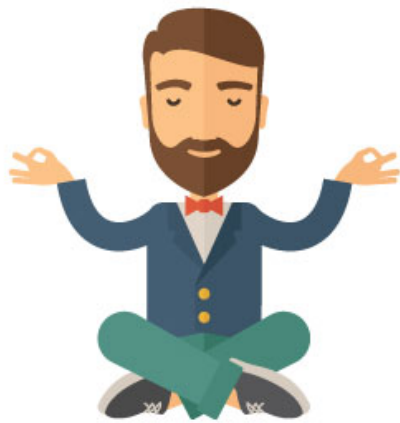
**Banquet Style
Round**



**Banquet Style
Square**



**U Shape
Configuration**



**@EnableAutoConfiguration is like having
your own opinionated event planner.**



Phil Webb
@phillip_webb

Follow

For those on reddit.com/r/java/ saying it's Spring Boot is "the framework for a framework" here's a diagram:



RETWEETS
112

LIKES
66



7:54 PM - 8 Sep 2015



Spring Boot is **not** Spring
Boot without Auto
Configurations!

```
package schultz.dustin.io
```

```
@EnableAutoConfiguration
```

```
public class Foo {
```

```
    ...
```

```
}
```

◀ Package has significance

◀ Intelligent and seemingly
“magical” annotation that
enables features and configures
functionality


```
package schultz.dustin.io
```

```
@Configuration
```

```
@ComponentScan
```

```
@EnableAutoConfiguration
```

```
public class Foo {
```

```
    ...
```

```
}
```



◀ 3 very common annotations in
Spring Boot apps

```
package schultz.dustin.io
```

```
@SpringBootApplication
```

```
public class Foo {
```

```
    ...
```

```
}
```

◀ Introduced in Spring Boot 1.2.
Really three annotations in one.

Tuning your Auto Configuration

Intelligent Decision Making Based on Conditions



Presence/
Absence of Jar



Presence/
Absence of Bean



Presence/Absence
of Property



Many More!

=====

AUTO-CONFIGURATION REPORT

=====



Positive matches:

`EmbeddedServletContainerAutoConfiguration`
matched

- found web application

`StandardServletEnvironment`
(`OnWebApplicationCondition`)

...



Negative matches:

```
CassandraAutoConfiguration did not match
    - required @ConditionalOnClass classes
not found: com.datastax.driver.core.Cluster
(OnClassCondition)
```

...



Exclusions:

```
org.springframework.boot.autoconfigure.  
jdbc.DataSourceAutoConfiguration
```

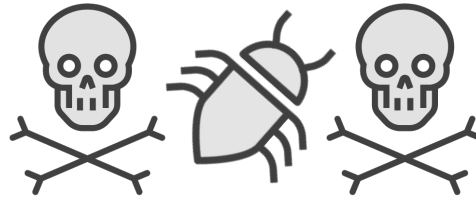



Unconditional classes:

```
org.springframework.boot.autoconfigure.  
PropertyPlaceholderAutoConfiguration
```

...

Enabling the Auto Configuration Report



--debug

cmd line arg

-Ddebug

VM arg

export DEBUG=true

Environment var

debug=true

application.properties

logging.level.=debug

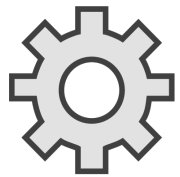
application.properties

Many more!

<http://github.com/dustinschultz/just-gif-it>

You've identified an auto configuration that needs tuning, now what?

```
1 @EnableAutoConfiguration(exclude = { SomeAutoConfig.class })
2 public class JustGifItApplication {
3     ...
4 }
```

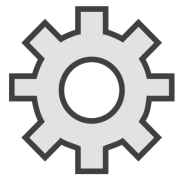


Excluding Unnecessary or Misconfigured Auto Configurations via Annotations

Via the `exclude` parameter of annotation as a comma separated list of classes

- <http://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#auto-configuration-classes>

```
1 @EnableAutoConfiguration(excludeName = { "a.b.SomeAutoConfig" })
2 public class JustGifItApplication {
3     ...
4 }
```

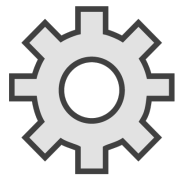


Excluding Unnecessary or Misconfigured Auto Configurations via Annotations

Via the `excludeName` **parameter of annotation as a comma separated list of class names**

- <http://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#auto-configuration-classes>

```
1 @SpringBootApplication(exclude = { SomeAutoConfig.class })
2 public class JustGifItApplication {
3     ...
4 }
```

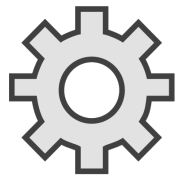


Excluding Unnecessary or Misconfigured Auto Configurations via Annotations

Via the `exclude` parameter of annotation as a comma separated list of classes

- <http://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#auto-configuration-classes>

```
1 @SpringBootApplication(excludeName = { "a.b.SomeAutoConfig" })
2 public class JustGifItApplication {
3     ...
4 }
```



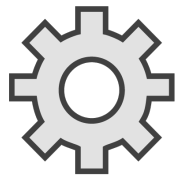
Excluding Unnecessary or Misconfigured Auto Configurations via Annotations

Via the `excludeName` **parameter of annotation as a comma separated list of classes**

- <http://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#auto-configuration-classes>

application.properties

```
1 ...  
2 spring.autoconfigure.exclude = my.company.SomeAutoConfig  
3 ...
```



Excluding Unnecessary or Misconfigured Auto Configurations via Properties

Via the application.properties **as a comma separated list of classes**

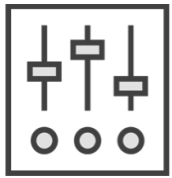
* Can be used in addition to annotations

When to Exclude Auto Configurations

- **Exclusions are all or nothing**
- **Use when you don't need configuration at all**

application.properties

```
1 # DAO (PersistenceExceptionTranslationAutoConfiguration)
2 spring.dao.exceptiontranslation.enabled = false
3
4 # Spring MVC
5 spring.mvc.favicon.enabled = false
```



Reconfiguring Auto Configurations via Properties (@ConditionalOnProperty)

Tune auto configurations via properties in the application.properties

* <http://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#common-application-properties>

```
PropertySourcesPlaceholderConfigurer#propertySourcesPlaceholderConfigurer  
    @ConditionalOnMissingBean (types: o.s.c.s.PropertySourcesPlaceholderConfigurer)
```

```
1 @Bean  
2 // Define our own  
3 public static PropertySourcesPlaceholderConfigurer  
4 propertySourcesPlaceholderConfigurer() {  
5     PropertySourcesPlaceholderConfigurer placeholderConfigurer  
6     = new PropertySourcesPlaceholderConfigurer();  
7     placeholderConfigurer.setNullValue("");  
8     return placeholderConfigurer;  
9 }
```



Overriding @ConditionalOnMissingBean Conditions

By defining a new @Bean before Auto Configuration happens

```
1 @Bean
2 // Disable HiddenHttpMethodFilter
3 public FilterRegistrationBean
4     register(HiddenHttpMethodFilter hiddenHttpMethodFilter) {
5     FilterRegistrationBean bean
6         = new FilterRegistrationBean(hiddenHttpMethodFilter);
7     bean.setEnabled(false);
8     return bean;
9 }
```



Excluding Auto Configured Servlets or Filters

By defining a `FilterRegistrationBean` **or a** `ServletRegistrationBean` **and calling** `setEnabled(false)`

```
1 @Configuration
2 @Import({
3     DispatcherServletAutoConfiguration.class,
4     EmbeddedServletContainerAutoConfiguration.class,
5     ...
6 })
7 public class JustGifItApplication {
8     ...
9 }
```



Completely Custom Auto Configuration

Remove `@EnableAutoConfiguration` **and manually configure an array of auto configuration classes with the** `@Configuration` **and** `@Import` **annotations**

When in doubt, always
check the Spring
Documentation first



Using the @ConditionalOn... Annotations

Spring 4 Conditional Configuration



A single condition that must be matched in order for a component to be registered.

Presence or Absence of Class on Classpath

@ConditionalOnClass

Attributes		
	[]	name = { "a.b.c.Foo" }
	[]	value = { Foo.class }

@ConditionalOnMissingClass

Attributes		
	[]	name = { "a.b.c.Foo" }
	[]	value = { Foo.class }

Presence or Absence of Defined Bean

@ConditionalOnBean







Attributes		
✓	[]	name = { "dataSource" }
✓	[]	value = { Foo.class }
✓	[]	type = { "a.b.c.Foo" }
✓	[]	annotation = { Foo.class }
✓	enum	search = { ALL }
✗	[]	ignored = { "a.b.c.Foo" }
✗	[]	ignoredType = { Foo.class }

@ConditionalOnMissingBean

Attributes		
✓	[]	name = { "dataSource" }
✓	[]	value = { Foo.class }
✓	enum	type = { "a.b.c.Foo" }
✓	[]	annotation = { Foo.class }
✓	[]	search = { ALL }
✓	[]	ignored = { "a.b.c.Foo" }
✓	[]	ignoredType = { Foo.class }

Presence or Absence of a Property Having Value

@ConditionalOnProperty

Attributes		
	<code>[]</code>	<code>name = { "my-property" }</code>
	<code>[]</code>	<code>value = { "my-property" }</code>
	<code>String</code>	<code>havingValue = "foo"</code>
	<code>[]</code>	<code>prefix = { "some.prefix" }</code>
	<code>boolean</code>	<code>matchIfMissing = false</code>
	<code>boolean</code>	<code>relaxedNames = true</code>

Additional Conditions

- `@ConditionalOnJava`
- `@ConditionalOnJndi`
- `@ConditionalOnResource`
- `@ConditionalOnExpression`
- `@ConditionalOnWebApplication`
- `@ConditionalOnNotWebApplication`
- Don't be afraid to mix and match!

Writing Our First Auto Configuration

<http://github.com/dustinschultz/just-gif-it-autoconfigure>

Demystifying Auto Configuration

@EnableAutoConfiguration Unveiled

@EnableAutoConfiguration



@Import(EnableAutoConfigurationImportSelector.class)



SpringFactoriesLoader.loadFactoryNames(...)



/META-INF/spring.factories



o.s.b.a.EnableAutoConfiguration = o.s.b.a.SomeAutoConfig, ...

Module in Review

- **Auto configuration report**
- **Tuning auto configuration**
- **@ConditionalOn... annotations**
- **Writing our own custom auto configurations**