1-Using Multiple @Configuration Classes

To separate the concerns or to achieve modularization, we can define multiple @Configuration classes. During container initialization, we can use one of the following constructors of AnnotationConfigApplicationContext

public AnnotationConfigApplicationContext(Class<?>... annotatedConfigurationClasses)

public AnnotationConfigApplicationContext(String... basePackages)

Example

Beans

```
class DataSourceBean {
  public String getData() {
      return "some app data";
  }
}

public class Client {
  @Autowired
  private DataSourceBean dataSourceBean;
  public void showData() {
      System.out.println(dataSourceBean.getData());
  }
}
```

Configuration classes

```
@Configuration
public class DataSourceConfig {
  @Bean
  DataSourceBean dataSourceBean() {
      return new DataSourceBean();
  }
}

@Configuration
public class AppConfig {
  @Bean
  Client clientBean() {
      return new Client();
  }
  public static void main(String[] args) {
      AnnotationConfigApplicationContext context =
            new AnnotationConfigApplicationContext(AppConfig.class,
DataSourceConfig.class);
      context.getBean(Client.class).showData();
  }
}
```

Output: some app data

2- Using @Import

In above section, we saw how to use multiple classes by providing all @Configuration classes to the constructor of AnnotationConfigApplicationContext. That approach may not always be ideal. Often it is preferable to use an aggregation approach, where one @Configuration class logically imports the bean definitions defined by another.

The @Import annotation provides this kind of support. It is the equivalent to the <import/> element in a bean XML file.

Example

Bean classes

```
class DataSourceBean {
  public String getData() {
      return "some app data";
  }
}

public class Client {
  @Autowired
  private DataSourceBean dataSourceBean;
  public void showData() {
      System.out.println(dataSourceBean.getData());
  }
}
```

Configuration classes

```
@Configuration
public class DataSourceConfig {
  @Bean
  DataSourceBean dataSourceBean() {
      return new DataSourceBean();
  }
}
```

Following @Configuration class imports the above one.

```
@Configuration
@Import(DataSourceConfig.class)
public class AppConfig {
  @Bean
  Client clientBean() {
      return new Client();
  }
  public static void main(String[] args) {
      AnnotationConfigApplicationContext context =
              new AnnotationConfigApplicationContext(AppConfig.class);
      context.getBean(Client.class).showData();
  }
}
```

Output: some app data