1. Spring Context

Spring Boot لديها سياق خاص لعمل البرنامج و يعتمد السياق او (Spring Context) على مايسمى Beans.
على سبيل المثال لدينا دالة () message:

```
public void message(){
    System.out.println("hey");
}
```

لن يتم تشغيل الدالة الا بعد استدعائها في دالة main في السياق المعتاد، ولكن إذا أردنا اتباع سياق spring الخاص نقوم باستخدام مايسمى ب Spring Beans.

2. Spring Beans

Spring beans هي كائنات Java يتم إدارتها بواسطة Spring container.

```
@Bean
public void message(){
    System.out.println("hey");
}
```

عندما نقوم بإضافة @Bean اعلى الداله، سيقوم Spring بتشغيل واستدعاء العديد من الدوال والاسطر في سياقه الخاص وسيقوم باستدعاء الداله () message تلقائي.

لا يمكننا التنبؤ بالسياق الخاص ب Spring في حال كانت لدينا دالتان او أكثر.

```
@Bean
public void message(){
    System.out.println("hey from 1");
}

@Bean
public void message2(){
    System.out.println("hey from 2");
}
```

3. Spring Container

حاوية Spring هي المسؤولة عن إنشاء وتكوين وتجميع Spring Beans.

على سبيل المثال لو قمنا بانشاء دالة تقوم بارجاع قيمة نصية

```
@Bean
public String getName(){
    return "Maha";
}
```

سيقوم Spring باستدعاء الدالة وتنفيذها ومن ثم تخزين القيمة النصية في Spring Container.

لو قمنا بانشاء دالة تقوم باستقبال قيمة نصية وقمنا بإضافة @Bean سيقوم Spring بأخذ القيمة النصية المخزنة في Container وتمريرها للدالة.

```
@Bean
public String printName(String name){
    System.out.println(name);
    return name;
}

// Output:
// Maha
```

لدينا طريقتين لتحديد أي قيمة نصية يتم تمريرها في حال كانت لدينا أكثر من قيمة نصية مخزنة في Container:

1- باستخدام @Primary

```
@Bean
@Primary
public String getName(){
    return "Maha";
}
```

بوضع @Primary فوق الدالة تعني ان Spring سيقوم دائماً باستعمال القيمة النصية التي تم ارجاعها من هذه ال Bean.

2- باستخدام @Qualifier مع قيمة نصية فريدة

```
@Bean
@Qualifier("1")
public String getName(){
    return "Maha";
}

@Bean
@Qualifier("2")
public String getName2(){
    return "majd";
}

@Bean
public String printName(@Qualifier("1")String name){
    System.out.println(name);
    return name;
}

//Output:
//Maha
```

بوضع @Qualifier ("1") في Parameter الدالة تعني ان Spring سيقوم بأخذ القيمة النصية المرجعة من ال Bean المعرف ب @Qualifier ("1").

4. التعليقات التوضيحية Annotation

التعليقات التوضيحية لـ Spring Boot هي شكل من أشكال البيانات الوصفية التي توفر بيانات حول البرنامج. بمعنى آخر، يتم استخدام التعليقات التوضيحية لتوفير معلومات تكميلية حول البرنامج.
جميع الـ annotations مرفقة في المرجع التالي:

https://www.javatpoint.com/spring-boot-tutorial

5. ملفات Json (Java Object Notaion)

سميَ بهذا الاسم لمشابهته طريقة إنشاء الكائنات بلغة JavaScript.

5.1 ما هو JSON: هو ملف نصي يستخدم لتمثيل البيانات وتبادلها بين التطبيقات المختلفة، ومن ذلك نستنتج انه فقط طريقة لتمثيل البيانات وتناقلها وليس لغة برمجة بحد ذاته. كما يكون امتداد الملف json .

5.2 مميزاتها: استخدمت ملفات JSON على نطاق واسع لسهولة كتابتها واستخدامها. ومن استخداماتها الاساسية أن تكون حلقة ربط بين الخادم ولغة البرمجة بحيث يتم تناقل البيانات بينهم بهذه الصيغة، ويتم كتابة الملف باستخدام الأقواس المتعرجة { }.

5.3 أنواع بيانات Json: String, numbers, null, Boolean, arrays and objects

نصية (String) : سلسلة من الأحرف تمثل قيمة ما.

مثال : {"title": "Hello World"}

رقم (Number) : هو رقم صحيح أو حقيقي او عشري وقد يحتوي على اشارة.

مثال : {"age": 20}

قيمة منطقية (Boolean) : قيمة قد تكون اما صحيحة (true) او خاطئة (false) وهي مستخدمة بكثرة لاعتماد الحاسب عليها.

مثال : {"isAnimal": true}

قيمة فارغة (Null) : عدم وجود قيمة فعلية للمتغير.

مثال : {"job" : null}

كعدم وجود وظيفة للشخص.

مصفوفة (Array): هي سلسلة من القيم المترابطة مخزنة في متغير.

مثال : {"oddNumbers" : [1, 3, 5, 7, 9]}

الكائن (object) مجموعة من اسم للمتغيرات والقيم الخاصة بها

مثال: {"employee”: { "id" : 100, "name" : "Sami" }}

5.4 المفتاح والقيمة

المفتاح: يمثل اسم خاص لقيمة البيانات ويتم وضعه بين علاماتي التنصيص " " ويجب أن يكون متبوعا بـ نقطتان رأسيتان: ليفصل بين الاسم وقيمة البيانات. كما تكون المفاتيح عباره عن نصوص (string) مثل :
Company, Pen.

القيمة: تمثلّ البيانات ويمكن أن تكون اكثر من نوع: اسم، رقم، مصفوفه... *يجب تكون القيم أنواع متاح استخدامها مثل :
(String, number, object, array, Boolean or null)

مثال : { "Name": "Ahmed" }

وبين أزواج المفاتيح والقيم المتعددة يتم الفصل بينهم بوضع علامة فاصله (,)

مثال : {" Name": "Ali" , "Age": 23 , "id": 4433}

5.5 كائنات JSON المتداخلة (Nested JSON Objects) :

أي يمكن ان يحتوي الكائن على كائن آخر بداخلة كما في المثال التالي:

مثال :
```
{ "Book" : { " BookName": " Java programming", " price": 200, "
AuthorName": { " firstName": " xxxxxx", " secondName": " xxxxxx",
"AuthorMajor": ["CS", "IT", "IS"] } }
```

مثال شامل: لنأخذ الطالب كمثال نعرض به ما تعلمناه بطريقة ملفات JSON.

```
{"name":{ "FristName":"Ghaida" ,"LastName":"Albohairy" }, "id":437005 ,
"major":"Information Technology", "courses": [2,1] , "isSenior":False , "GPA":
null }
```