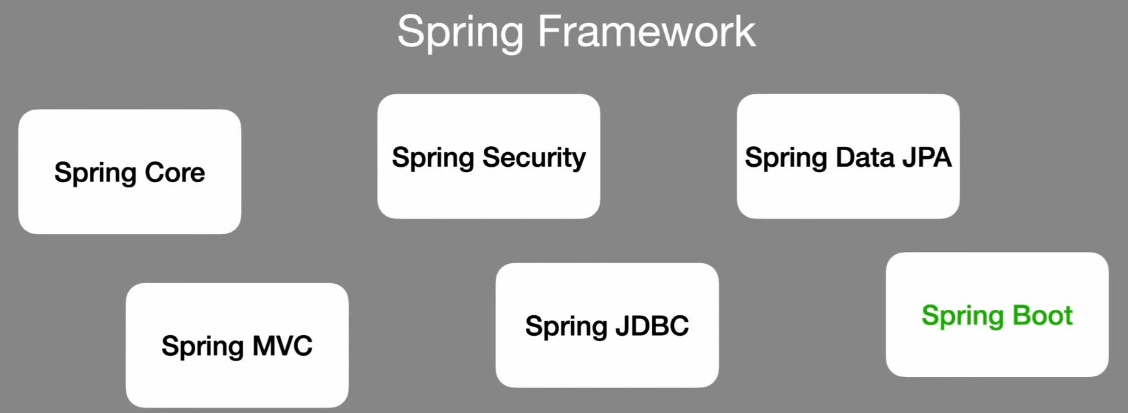
**Spring Boot Введение**

**Spring Boot** – это не отдельный вид **Spring Framework**, а лишь еще одна часть Spring Framework(такая же, как MVC, Core или Security). **Spring Boot** лишь еще один компонент Spring Frameworkкоторый упрощает конфигурацию и разработку. Все остальные компоненты Spring Framework, которые мы привыкли использовать, остаются.



**Проблемы Spring Framework которые решает** **Spring Boot**:

**1. Много конфигурации.**

Чем больше компонентов Spring Framework подключаем тем больше становится конфигурация (Hiberante, Security, Thymeleaf, JDBC …). **Spring Boot** предугадывает самую логичную конфигурацию, поэтому нам в большенстве случаев не надо ничего конфигурировать (конфигурация по умолчанию делает, то, что надо).

Например, мы подключили в pom.xml зависимость Thymeleaf. **Spring Boot** увидит эту зависимость и Thymeleaf автоматически заработает (конфигурация не понадобится).

**2. Зависимости и их совместимости.**

Если мы используем **Spring** без компонента **Boot**, то мы подключаем отдельно каждую зависимость.

В случае с **Spring Boot**, **Spring Boot** предоставляет так называемые «стартеры». Стартеры — это зависимости которые сгруппированы в одну большую зависимость. И теперь мы не подключаем каждую зависимость отдельно, а подключаем стартеры. Эти стартеры составлены так, что в них зависимости точно не приводят к конфликту. Поэтому подключая стареты мы избегает проблем с зависимостями, когда одна версия зависимости не сочетается с другой.

Пример одного из стартеров: он называется **spring-boot-starter-web** и этот стартер в себе содержит **spring-core**, **spring-web**, **spring-webmvc** и другие зависимости.



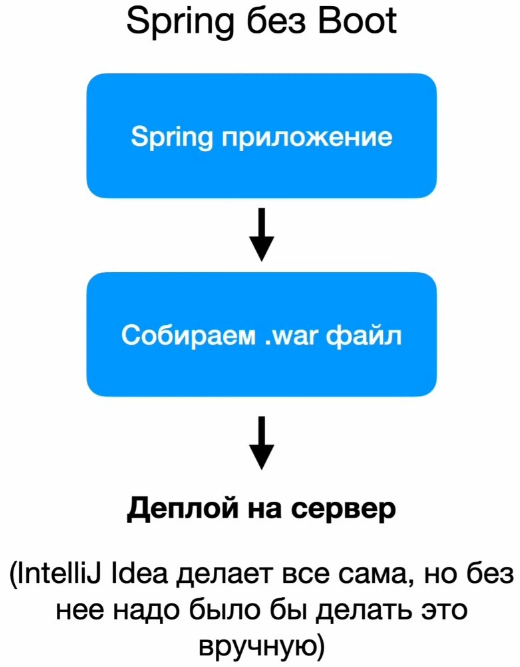
**2. Сборка приложений и деплой на сервер.**

Что происходит за кулисами сборки приложений БЕЗ **Spring Boot**?

Наше **Spring** приложение собирается в .war файл и этот .war файл деплоится на сервер (например TomCat). В случае если мы будет деплоить наше приложение на удаленный сервер, у нас не будет там среды разработки (Intellij Idea), нам придется всё делать вручную, через командную строку. И в этом случае нам придется самим собирать наше приложение в .war файл, запускать сервер и деплоить наш .war файл на сервер.

Что происходит за кулисами сборки приложений С **Spring Boot**?

**Spring Boot** это приложение которое в себе уже содержит встроенный веб-сервер. В **Spring** приложении есть один класс с методом ***main()*** (обычный java класс). Мы просто будем запускать этот метод в этом java классе и в итоге наше приложение будет запускаться. Под капотом будет запускаться встроенный веб-сервер и на этом сервере будет работать наше приложение. Тоесть не придется собирать .war файл, отдельно запускать веб-сервер и на него кидать наш .war файл. Конечно при желании можно собрать **Spring Boot** приложение в .war файл и задеплоить его как обычно на сервер вручную, но гораздо удобнее запускать приложение через метод main().



**Плюсы Spring Boot:**

**1.** Разработка становится легче.

**2.** Меньше конфигурации.

**3.** Фокусируемся на разработке самого приложения, а не на конфигурации.

**Минусы Spring Boot:**

**1.** Чем Больше магии под капотом, чем больше Framework берет на себяи чем меньше мы видим, что присходит по капотом, тем больше вероятность того, что могут появится сложно устранимые ошибки. В случае если появится ошибки будет сложно понять это мы допустили эту ошибку или это **Spring Boot** что то сконфигурировал не так. Поэтому помимо испоьзования **Spring Boot** важно понимать как конфигурировать приложения вручную.