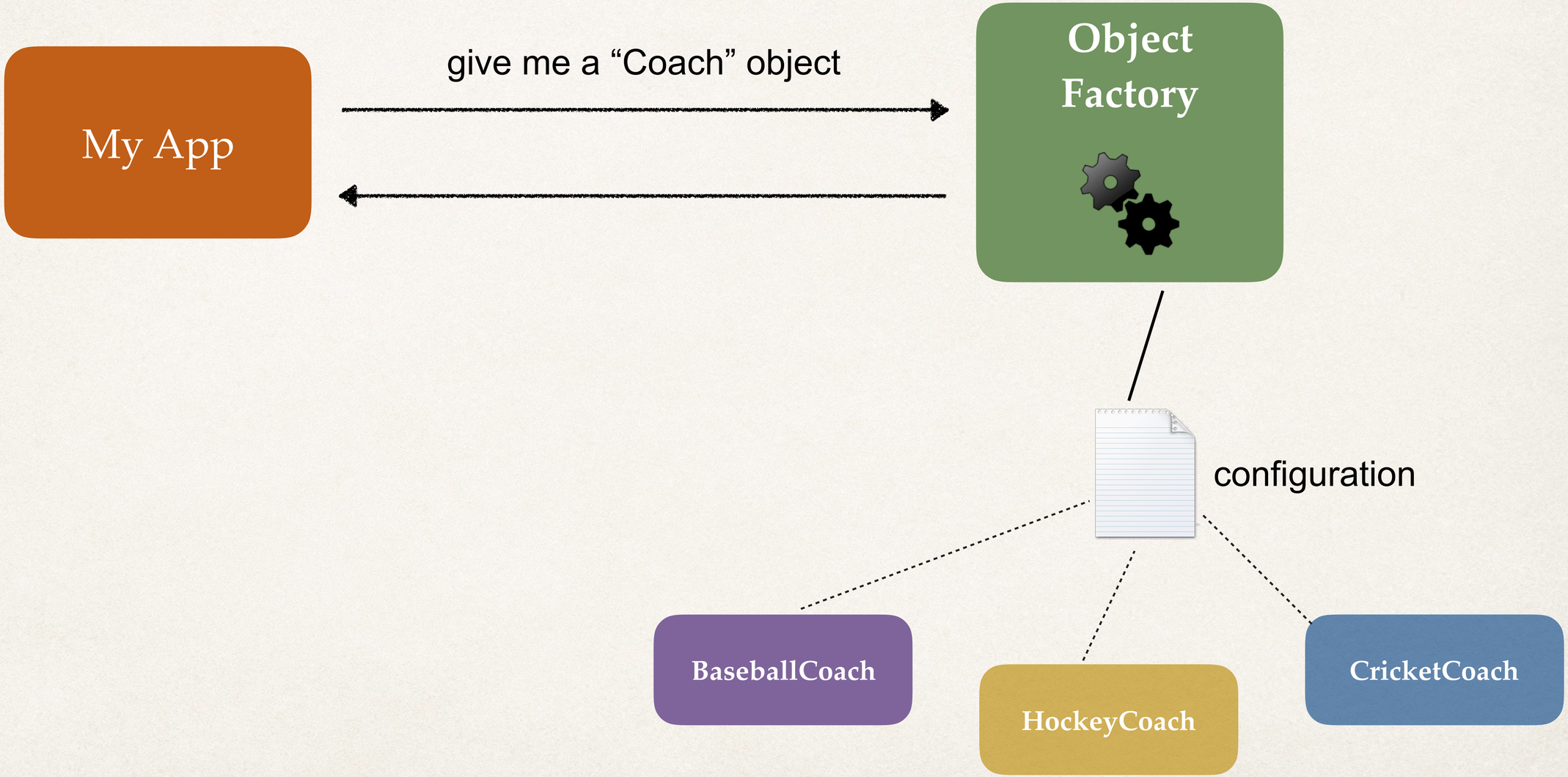
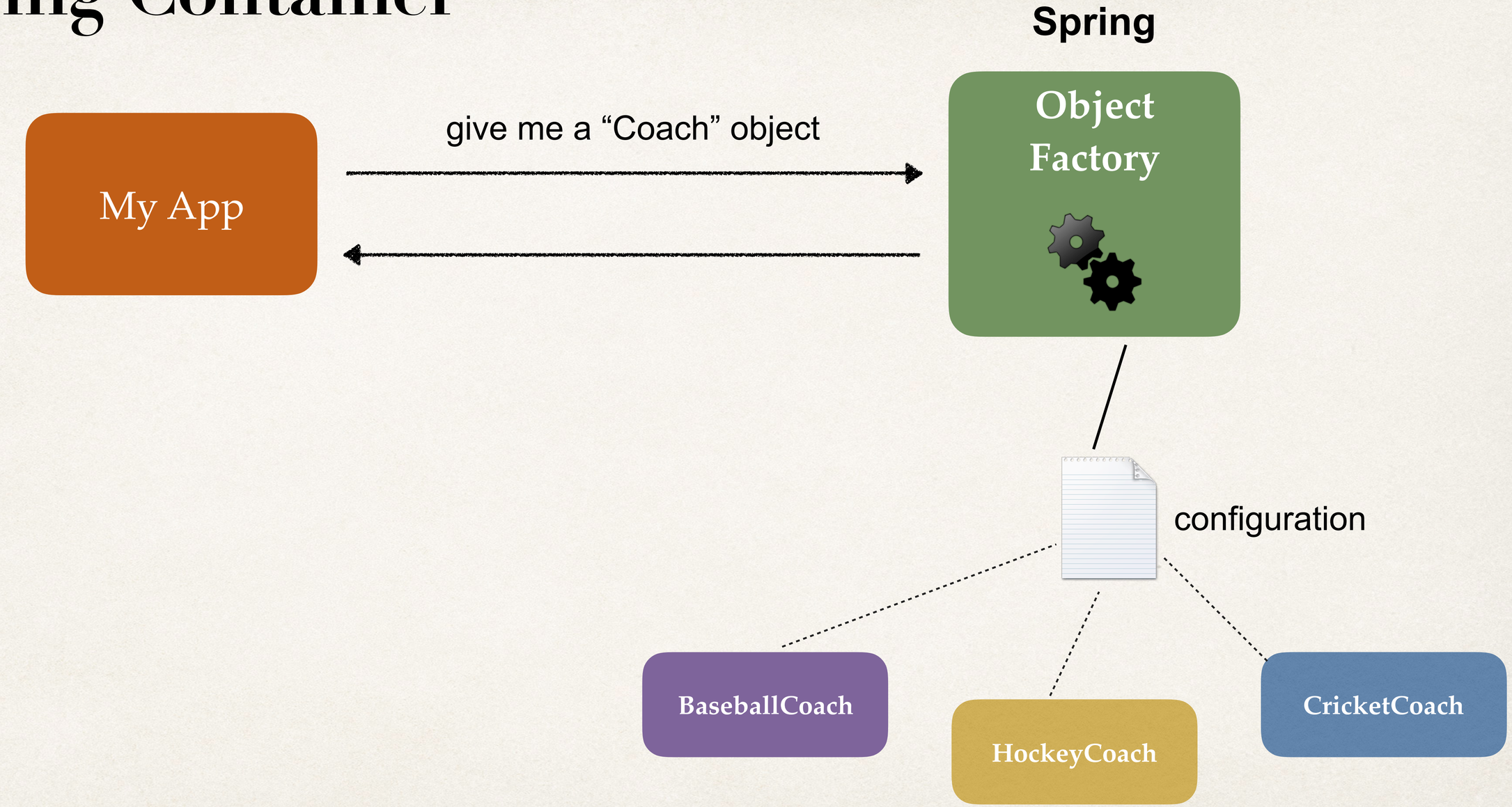
luv2code

Spring Inversion of Control

Ideal Solution



Spring Container



Spring Container

- Primary functions
  - Create and manage objects *(Inversion of Control)*
  - Inject object's dependencies *(Dependency Injection)*

Configuring Spring Container

- XML configuration file *(legacy, but most legacy apps still use this)*
- Java Annotations *(modern)*
- Java Source Code *(modern)*

Spring Development Process

Step-By-Step

1. Configure your Spring Beans

2. Create a Spring Container

3. Retrieve Beans from Spring Container

Step 1: Configure your Spring Beans

File: applicationContext.xml

```
<beans ... >

    <bean  id="myCoach"
        class="com.luv2code.springdemo.BaseballCoach">
    </bean>

</beans>
```

Step 2: Create a Spring Container

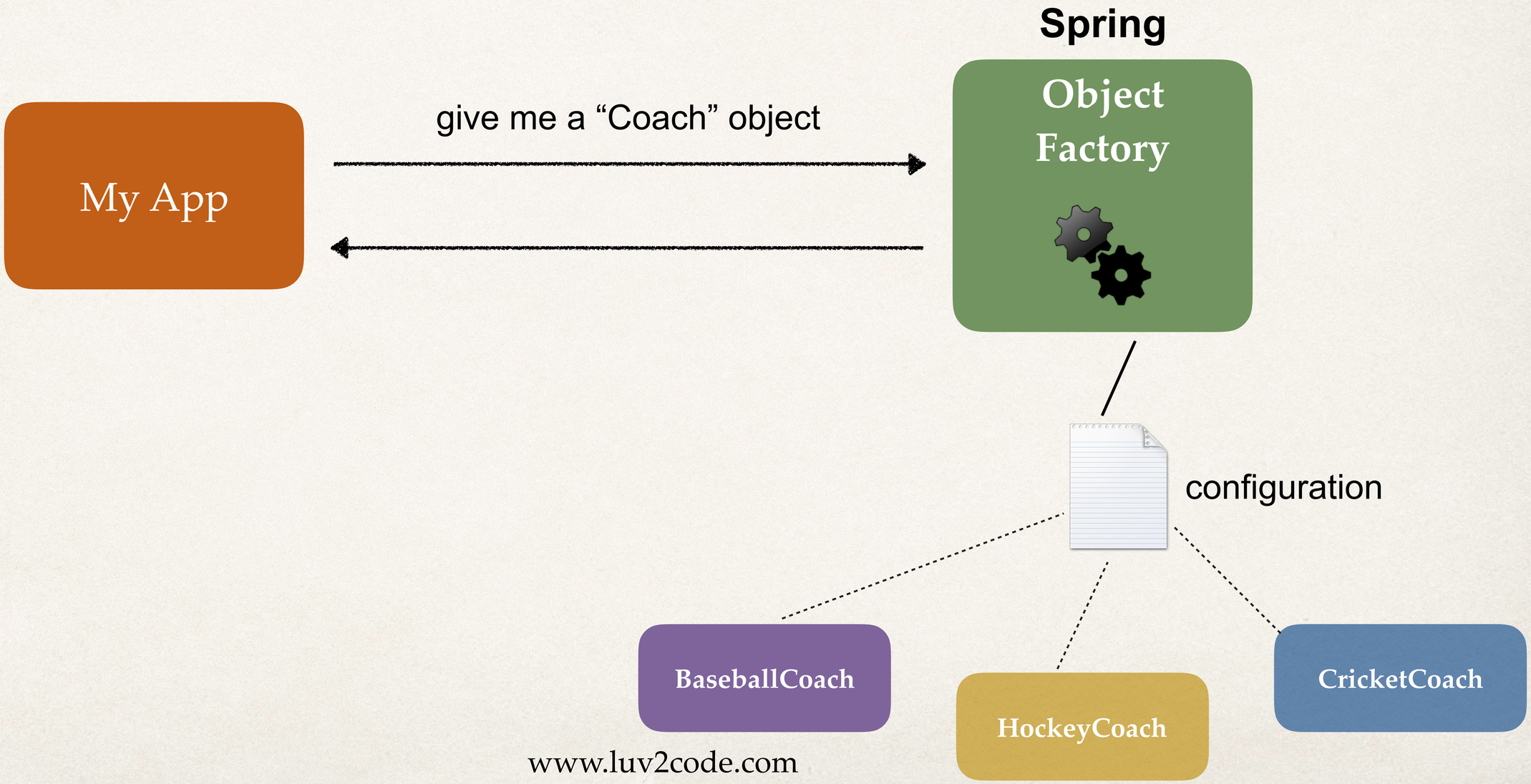
- Spring container is generically known as **ApplicationContext**

- Specialized implementations
  - ClassPathXmlApplicationContext
  - AnnotationConfigApplicationContext
  - GenericWebApplicationContext
  - others …

Step 2: Create a Spring Container

```
ClassPathXmlApplicationContext context =
    new ClassPathXmlApplicationContext("applicationContext.xml");
```

Step 3: Retrieve Beans from Container



Step 3: Retrieve Beans from Container

```
// create a spring container
ClassPathXmlApplicationContext context =
    new ClassPathXmlApplicationContext("applicationContext.xml");

// retrieve bean from spring container
Coach theCoach = context.getBean("myCoach", Coach.class);
```

File: applicationContext.xml

```
<bean  id="myCoach"
    class="com.luv2code.springdemo.BaseballCoach">
</bean>
```