luv2code

Spring Dependency Injection

Dependency Injection

The dependency inversion principle.

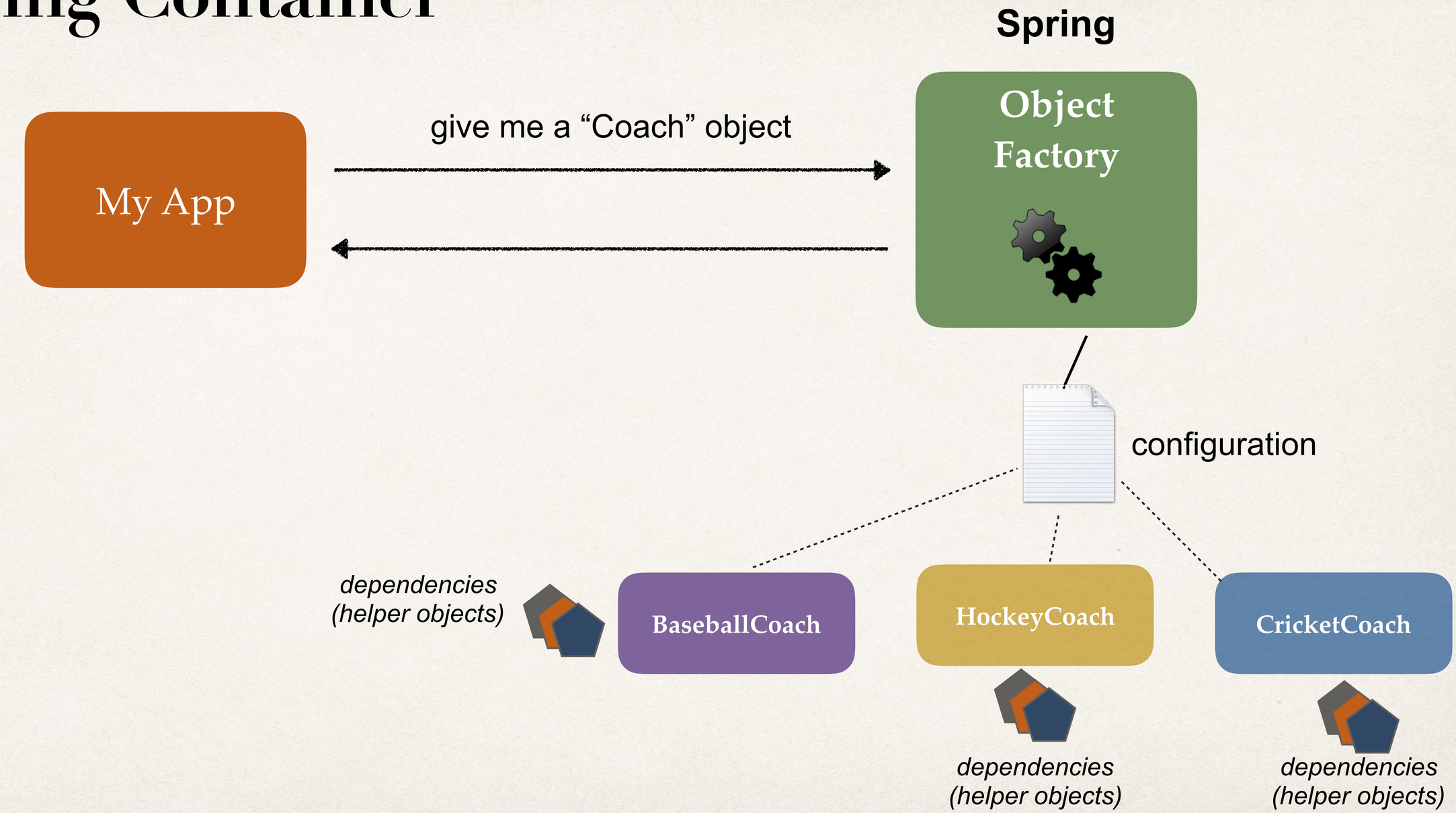
The client delegates to calls to another object the responsibility of providing its dependencies.

Car Factory

give me a “Car” object

Car Factory

Spring Container



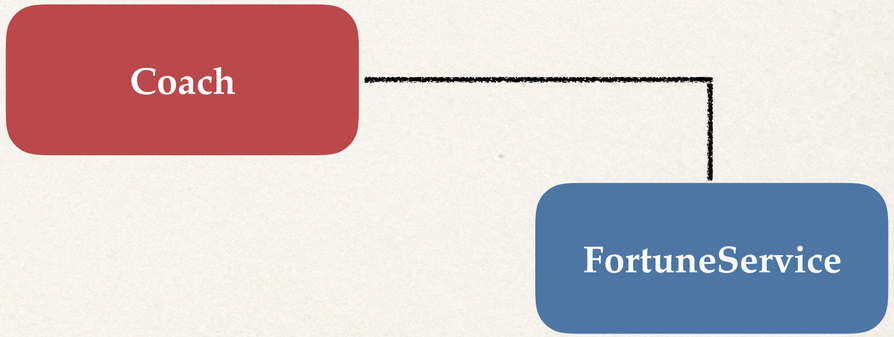
Spring Container

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

Demo Example

- Our **Coach** already provides daily workouts

- Now will also provide daily fortunes
  - New helper: **FortuneService**
  - This is a ***dependency***



Injection Types

- There are many types of injection with Spring

- We will cover the two most common
  - Constructor Injection
  - Setter Injection

- Will talk about “auto-wiring” in the Annotations section later

Development Process - Constructor Injection

1. Define the dependency interface and class

2. Create a constructor in your class for injections

3. Configure the dependency injection in Spring config file

Step 1: Define the dependency interface and class

File: FortuneService.java

```
public interface FortuneService {

   public String getFortune();

}
```

File: HappyFortuneService.java

```
public class HappyFortuneService implements FortuneService {

   public String getFortune() {
      return "Today is your lucky day!";
   }
}
```

Step 2: Create a constructor in your class for injections

File: BaseballCoach.java

```
public class BaseballCoach implements Coach {

    private FortuneService fortuneService;

    public BaseballCoach(FortuneService theFortuneService) {
        fortuneService = theFortuneService;
    }
    ...
}
```

Step 3: Configure the dependency injection in Spring config file

File: applicationContext.xml

```
<bean id="myFortuneService"
      class="com.luv2code.springdemo.HappyFortuneService">
 </bean>

<bean  id="myCoach"
    class="com.luv2code.springdemo.BaseballCoach">
       <constructor-arg ref="myFortuneService" />
</bean>
```

Spring Container