ПРИМЕНЕНИЕ ФРЕЙМВОРКА ВНЕДРЕНИЯ ЗАВИСИМОСТЕЙ DAGGER 2 НА ПРИМЕРЕ ПРОГРАММНОГО СРЕДСТВА ПРЕДУПРЕЖДЕНИЯ ВОДИТЕЛЯ АВТО О СНИЖЕНИИ СКОРОСТИ НА ПЛАТФОРМЕ ANDROID

Лукин И.С., Красновский О.С., Труш Е.Г.

Институт информационных технологий БГУИР,
г. Минск, Республика Беларусь

Образцова О. Н. – и.о. зав. кафедрой ИСиТ, к.т.н., доцент

В работе изложены базовые понятия и информация по применению метода внедрения зависимостей на примере программного средства предупреждения водителя авто о снижении скорости на платформе Android.

Прежде чем начать говорить о внедрении зависимостей, в первую очередь, стоит понять, что из себя представляют зависимости и связи. Например, в программном средстве есть класс LocationService, отвечающий за отслеживание местоположения пользователя, расчета скорости передвижения и инициации отправки предупреждений. Данный класс использует в себе другой класс, который называется LocationServiceController. Из этого следует, что класс LocationService зависит от класса LocationServiceController, и не может работать без него. Так же это означает, что везде, где будет применяться класс LocationService будет применяться и класс LocationServiceController, то есть нельзя повторно использовать первый класс без повторного использования второго. В данном случае LocationService – зависимый класс, а LocationServiceController – зависимость. Зависимый зависит от своих зависимостей.

Два зависимых класса называются связанными. Связь между ними может быть двух типов – сильная либо слабая. Так же, стоит отметить, что зависимости всегда направлены, это значит, что класс LocationService зависит от LocationServiceController, в то же время класс LocationServiceController может быть полностью независим.

Стоит всегда иметь ввиду, что большое количество неконтролируемых зависимостей в классе может привести к так называемым проблемам сильных связей, что в свою очередь приводит к следующим факторам:

1. Усложнение тестирования программного средства – очень сложно, а порой и невозможно протестировать работу класса-зависимости отдельно, в случае если его экземпляр создается только в зависимом классе;
2. Невозможность повторного применения кода, что является нарушением одной из базовых идей объектно-ориентированного программирования;
3. Ухудшение процесса поддержки кода в случае роста проекта – данный фактор является следствием двух предыдущих.

Одним из решений проблем сильных связей и является метод внедрения зависимостей.

Внедрение зависимостей — это метод, при котором один объект предоставляет зависимости другого объекта. Зависимость — это объект, который мы можем использовать (LocationServiceController). Внедрение — это передача зависимости зависимому объекту (LocationService), который будет данной зависимостью пользоваться. LocationServiceController — это часть состояния LocationService. Передать зависимости классу, вместо того чтобы позволить LocationService создать эту зависимость — базовое требование метода проектирования «Внедрение зависимости». Данный метод основывается на концепции инверсии контроля, которая простыми словами говорит о том, что ни один класс не должен создавать экземпляр другого класса, а должен получать все экземпляры из класса конфигуратора.

При разработке программного средства предупреждения водителя авто о снижении скорости на платформе Android для обеспечения реализации метода реализации зависимостей был применен фреймворк «Dagger 2».

Dagger 2 — это один из фреймворков с открытым исходным кодом для внедрения зависимостей который генерирует большое количество шаблонного кода за разработчика. Единственной причиной применения именно этого фреймворка это тот факт, что сейчас это единственный фреймворк внедрения зависимостей, который генерирует полностью отслеживаемый Java код, имитирующий тот код, который разработчик мог написать вручную. Это означает, что в построении графа зависимостей нет ничего заурядного. Dagger 2 менее динамичен, чем другие (в нем не используется рефлексия), но простота и производительность сгенерированного кода находятся на том же уровне, что и у написанного вручную. Коротко, Dagger 2 генерирует весь шаблонный код для внедрения зависимостей за разработчика.

На рисунке 1 представлен пример схемы предоставления зависимостей с точки зрения фреймворка Dagger 2.

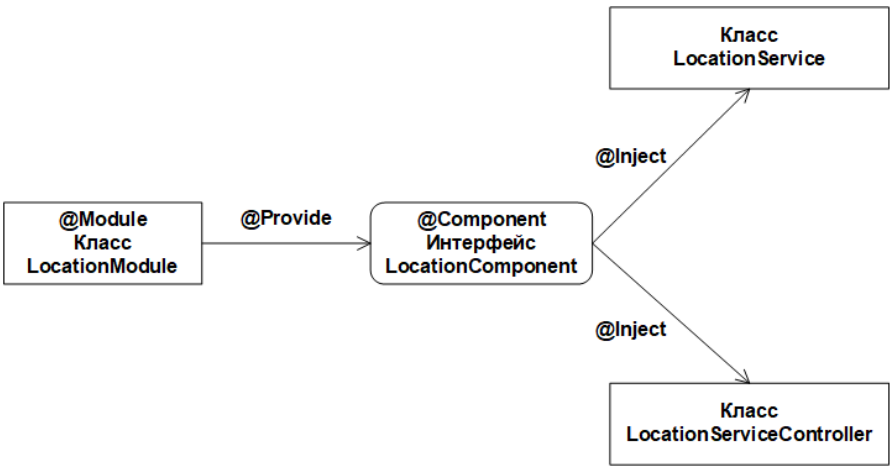

Рисунок 1 – Пример схемы предоставления зависимостей Dagger 2

Dagger 2 использует аннотации и обработчики аннотаций соответственно. Таким образом, можно отследить весь сгенерированный код во время компиляции. Исходя из этого, нет ухудшения производительности, а ошибки легко отслеживаются.

Аннотации — вид метаданных, который может быть связан с классами, методами, полями и даже другими аннотациями. Аннотации используются в Java для предоставления дополнительной информации, как альтернатива XML или маркерными интерфейсам. К аннотациям также можно получить доступ и в процессе выполнения программы через механизм рефлексии.

Обработчик аннотаций — это генератор кода, которые скрывают шаблонный код и создает его во время компиляции. Стоить отметить, что пока эти действия выполняются во время компиляции, никакого отрицательного влияния на производительность нет.

В качестве примера аннотаций фреймворка Dagger 2 рассмотрим наиболее важные из них – Inject, Component, Module и Provides.

Inject – метка зависимостей, которые должны быть предоставлены фреймворком для последующего их внедрения. Иначе выражаясь, аннотация Inject сообщает фреймворку какие зависимости должны быть предоставлены зависимому объекту. В качестве примера – для предоставление классу LocationService экземпляра LocationServiceController, класс зависимость должен быть помечен аннотацией Inject.

Аннотация Component – это аннотация интерфейса объединяющего части процесса внедрения зависимостей. При использовании данной аннотации определяется из каких модулей или других компонентов будут браться зависимости. Также здесь можно определить какие зависимости будут видны открыто и где компонент способен внедрять объекты.

Module – если вкратце, то эта аннотация отмечает модули и классы программного средства.

Provides – этой аннотацией помечаются методы предоставления зависимостей внутри модулей.

Как результат, применение фреймворка Dagger 2 для реализации метода внедрения зависимостей в программном средстве предупреждения водителя авто о снижении скорости на платформе Android позволило создать легко тестируемое, стабильное и надежное приложение, с возможностью последующего расширения функционала и поддержки имеющегося.

Список использованных источников:

1. Jenkov [Электронный ресурс]. – Режим доступа: http://tutorials.jenkov.com/dependency-injection/index.html. – Дата доступа: 01.02.2019

2. Medium [Электронный ресурс]. – Режим доступа: https://medium.com/@harivigneshjayapalan/dagger-2-for-android-beginners-introduction-be6580cb3edb. – Дата доступа: 02.02.2019