УДК 004.42

ПРИМЕНЕНИЕ ШАБЛОНА ПРОЕКТИРОВАНИЯ «СТРОИТЕЛЬ» И ФРЕЙМВОРКА «SPRING» В МНОГОМОДУЛЬНЫХ ПРИЛОЖЕНИЯХ МАШТАБА ПРЕДПРИЯТИЯ

О.В. Алитойть, магистрант, инженер-программист,
С.И. Сиротко, кандидат физ.-мат. наук, доцент,
Белорусский государственный университет информатики и радиоэлектроники, oleg_al@tut.by

При проектировании и написании приложений часто приходится решать задачу по инициализации какого-либо объекта-сущности либо коллекции значениями, несущими определенную функциональную нагрузку. Особенно характерна это для многомодульных приложений, где, согласно принципу модульности, должна присутствовать возможность расширять либо подменять определенный функционал для достижения заданной цели бизнес-логики при подключении модуля, а целостность приложения должна сохраняться даже если отдельные модули отсутствуют. Достаточно гибким решением в данной ситуации является использование паттернов (шаблонов) проектирования: “Строитель” (“Builder”), “Одиночка” (“Singleton”) и “Фабрика” (“Factory”) при использовании базы для разработки (framework) “Spring”.

Паттерн “Строитель” (“Builder”) — относится к порождающим паттернам. Он решает задачу по отделению конструирования сложного объекта от его результирующего представления, чтобы в результате одного и того же процесса конструирования могли получаться разные представления. Тем самым обособляется код, реализующий конструирование, что позволяет изменять внутреннее представление продукта, и дает более тонкий контроль над процессом конструирования. Применение паттерна “Строитель” эффективно, если алгоритм создания сложного объекта не должен зависеть от его структуры, либо необходимо обеспечить различные представления объекта.

Паттерн “Одиночка” (“Singleton”) призван гарантировать единственность экземпляра класса и предоставить к нему глобальную точку доступа. Помимо контролируемого доступа к единственному экземпляру, достигается сокращение числа имён и большая по сравнению с операциями класса гибкость. В то же время глобальные объекты могут быть вредны для объектного программирования, например с точки зрения масштабируемости проекта. Применяется данный паттерн там, где необходим ровно один экземпляр некоторого класса, легко доступный всем клиентам.

Паттерн “Фабрика” (“Factory”) решает задачу предоставления интерфейса для создания семейств взаимосвязанных или взаимозависимых объектов, не специфицируя их конкретных классов. Данный паттерн применяется в случаях, когда система не должна зависеть от того, как создаются, компонуются и представляются входящие в нее объекты, либо входящие в семейство взаимосвязанные объекты должны использоваться совместно, либо система должна конфигурироваться одним из семейств составляющих ее объектов.

Рассмотрим совместное использование данных паттернов для получения автоматически заполняемой фабрики объектов с использованием java-фреймворка “Spring”.

Фреймворк “Spring” позволяет конструировать объекты по xml-описанию до их реального использования другими объектами приложения. Все созданные и проинициализированные объекты хранятся в глобальной фабрике, называемой бин-контейнером. Далее в приложении мы можем получить доступ к любому сконструированному объекту из бин-контейнера по его текстовому идентификатору.

Пусть в бин-контейнере сконфигурирована фабрика под названием “MainFactory” на основе класса java.util.HashMap, которая является “одиночкой”. Данная фабрика должна заполняться независимо из разных модулей, ее невозможно пересоздать и переинициализировать отдельно в каждом модуле, так как ее содержимое будет постоянно переписываться, и объект будет заполнен только частью необходимых значений. Для получения корректного объекта-фабрики существует следующее решение: в каждом модуле создается отдельный объект-фабрика “SomeModuleFactory” с необходимым набором значений либо объектов, которые должны попасть в основной объект-фабрику “MainFactory”, а также объект-строитель, который перенесет значения из “SomeModuleFactory” в “MainFactory”, соблюдая общему для всех модулей правила. Объект-строитель должен реализовывать интерфейс org.springframework.beans.factory.InitializingBean, который предоставляет фреймворк “Spring”. После инициализации объекта-строителя будет автоматически вызван метод afterPropertiesSet(), который обратится к методу build() данного объекта, чтобы заполнить заданный объект-фабрику “MainFactory” необходимыми значениями из заданного объекта-фабрики “SomeModuleFactory”. Сконфигурировав эти объекты для каждого модуля, получим в результате правильно заполненный глобальный объект-фабрику.

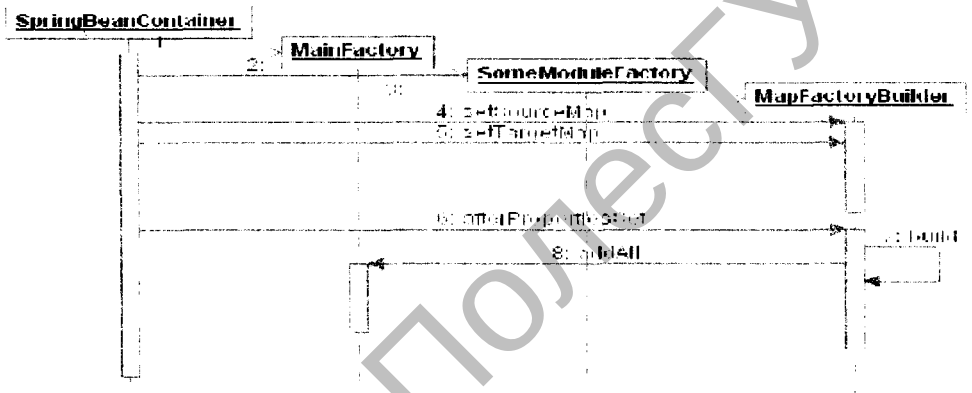

Рис.1. Диаграмма последовательности (упрощенная).

Приведенное решение поставленной задачи по инициализации глобальной фабрики в многомодульном приложении позволяет обеспечить целостность данных и необходимую общую последовательность инициализации приложения, при этом сохраняя прозрачность и контролируемость его структуры в рамках общих концепций проектирования сложных программ.