# Паттерн «Адаптер»

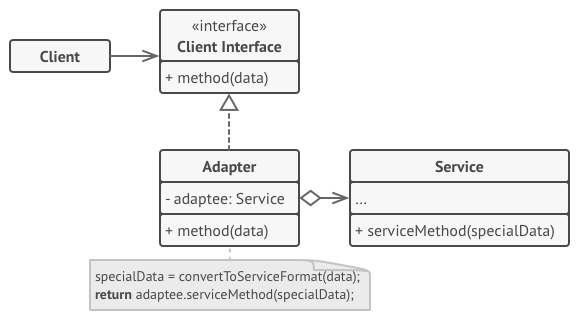
## Назначение и цель применения паттерна

Обеспечение совместной работы компонентов с несовместимыми интерфейсами.

Паттерн применяется тогда, когда необходимо осуществить взаимодействие объектов с несовместимыми интерфейсами.

## UML-диаграмма паттерна

### Адаптер объектов



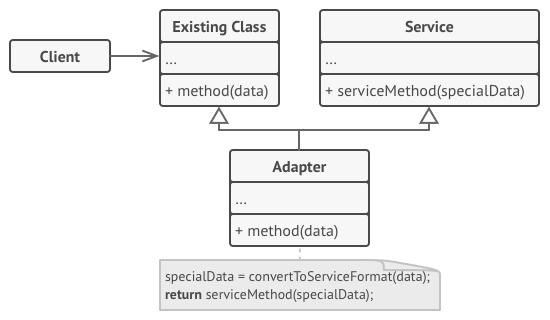
Client – класс, содержащий существующую логику программы.

Client Interface – описание протокола, с помощью которого клиентский класс может работать с другими классами.

Service – сторонний класс, который не может использоваться клиентским классом напрямую из-за несовместимости интерфейсов.

Adapter - класс, который может одновременно работать и с клиентом, и с сервисом. Он реализует клиентский интерфейс и содержит ссылку на объект сервиса. Адаптер получает вызовы от клиента через методы клиентского интерфейса.

### Адаптер классов

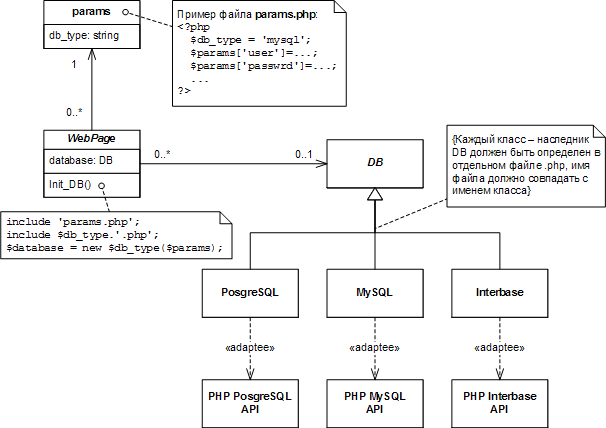


В данной реализации адаптер не нуждается в ссылке на объект сервиса, т.к. он наследуется от класса сервиса.

## Области применения

Проект, в котором используются сторонние библиотеки, работающие с данными иного формата, нежели классы основного проекта. Например, программа работает с данными в формате XML, а сторонняя библиотека работает только с данными в формате JSON.

В устаревших версиях языка PHP доступ к СУБД реализован в виде набора функций, для каждой СУБД они имеют различные наименования и, иногда, различный набор используемых параметров. Типичным примером использования шаблона Адаптер можно назвать создание классов, приводящих к единому интерфейсу функции языка PHP обеспечивающие доступ к различным СУБД.



## Особенности паттерна

В одной программе может существовать множество адаптеров.

Существует две реализации адаптера: адаптер объектов и адаптер классов. Адаптер объектов содержит ссылку на объект адаптируемого сервиса. Адаптер объектов может быть реализован во всех языках, поддерживающих парадигму ООП. Адаптер классов наследует как клиентский класс, так и класс сервера. Такой подход возможен только в языках, поддерживающих множественное наследование.

## Источники

1. Адаптер. Refactoring Guru. <https://refactoring.guru/ru/design-patterns/adapter>
2. Адаптер (шаблон проектирования). Википедия. <https://ru.wikipedia.org/wiki/Адаптер_(Шаблон_проектирования)>