Подробный курс

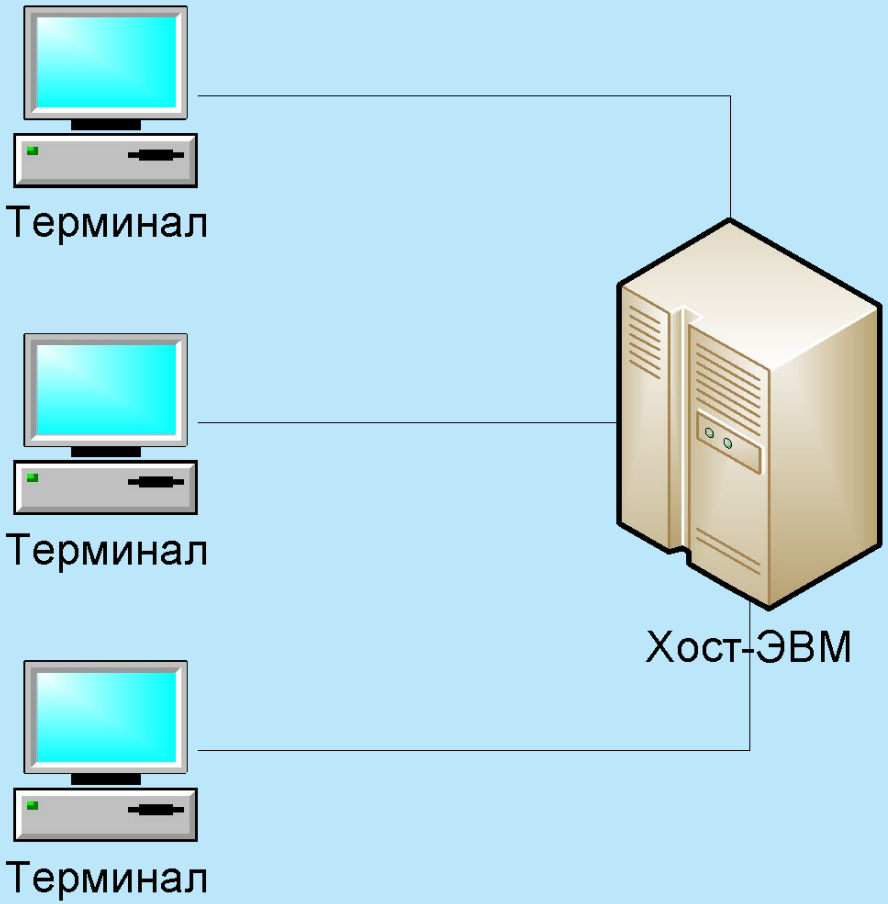
Современные веб-технологии

Лекция 4. Архитектурные особенности проектирования и разработки Веб-приложений

Понятие «Архитектура»

- **Архитектура –** это набор значимых решений по поводу организации системы программного обеспечения, набор структурных элементов и их интерфейсов, при помощи которых компонуется система, вместе с их поведением, определяемым во взаимодействии между этими элементами, компоновка элементов в постепенно укрупняющиеся подсистемы, а также стиль архитектуры, который направляет эту организацию – элементы и их интерфейсы, взаимодействия и компоновку

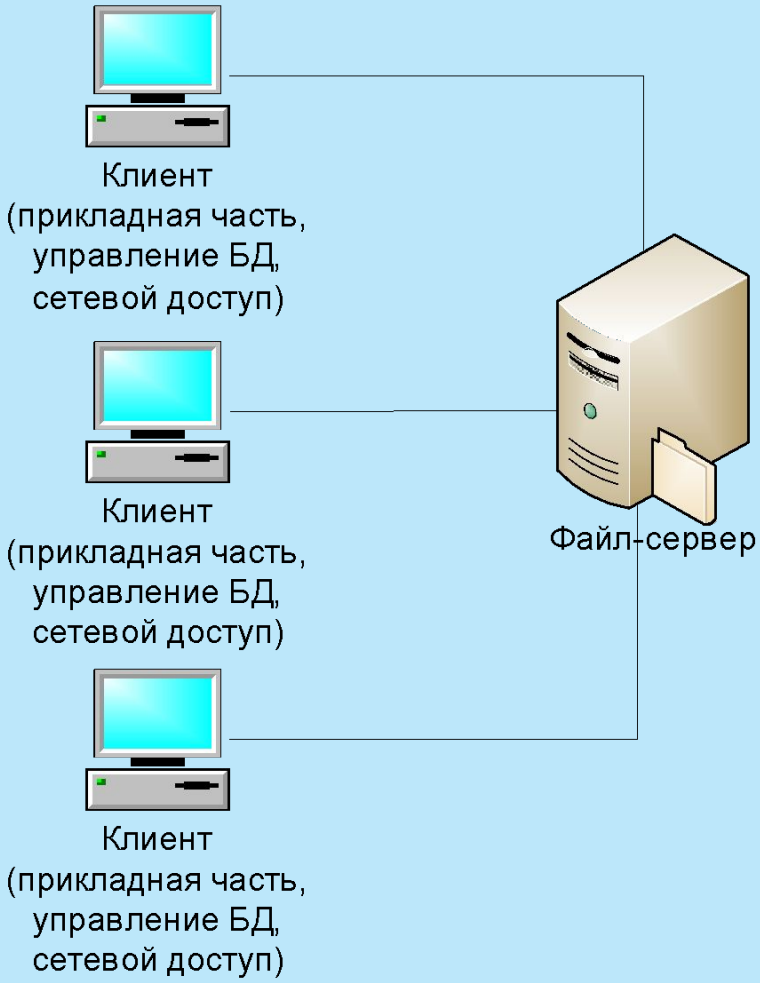
Централизованная архитектура



Централизованная архитектура

- **Достоинства:**
  - пользователи совместно используют дорогие ресурсы ЭВМ и дорогие периферийные устройства
  - централизация ресурсов и оборудования облегчает обслуживание и эксплуатацию вычислительной системы
  - отсутствует необходимость администрирования рабочих мест пользователей
- **Главный недостаток:**
  - пользователи полностью зависят от администратора хост-ЭВМ

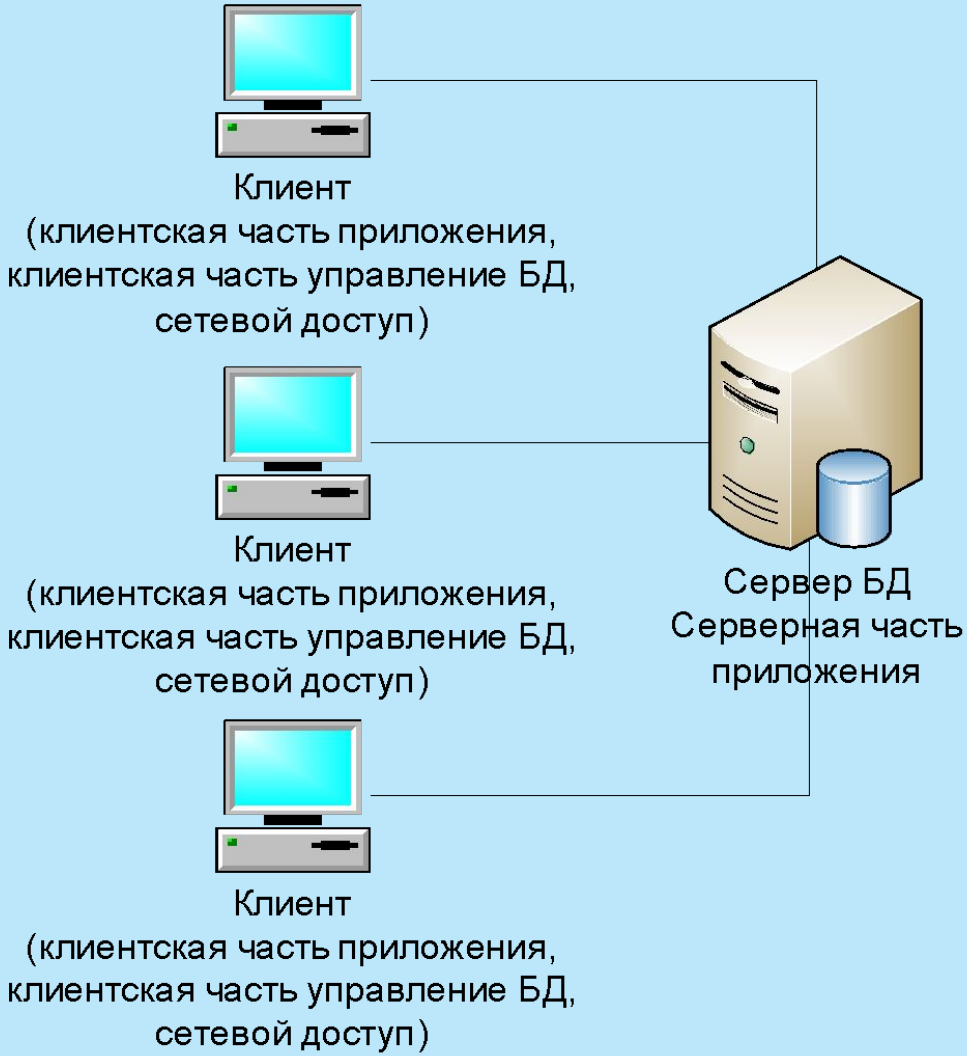
Архитектура «файл-сервер»



Архитектура «файл-сервер»

- **Достоинства:**
  - многопользовательский режим работы с данными
  - удобство централизованного управления доступом
  - низкая стоимость разработки
  - высокая скорость разработки
  - невысокая стоимость обновления и изменения ПО
- **Недостатки:**
  - проблемы многопользовательской работы с данными
  - низкая производительность
  - плохая возможность подключения новых клиентов
  - ненадежность системы

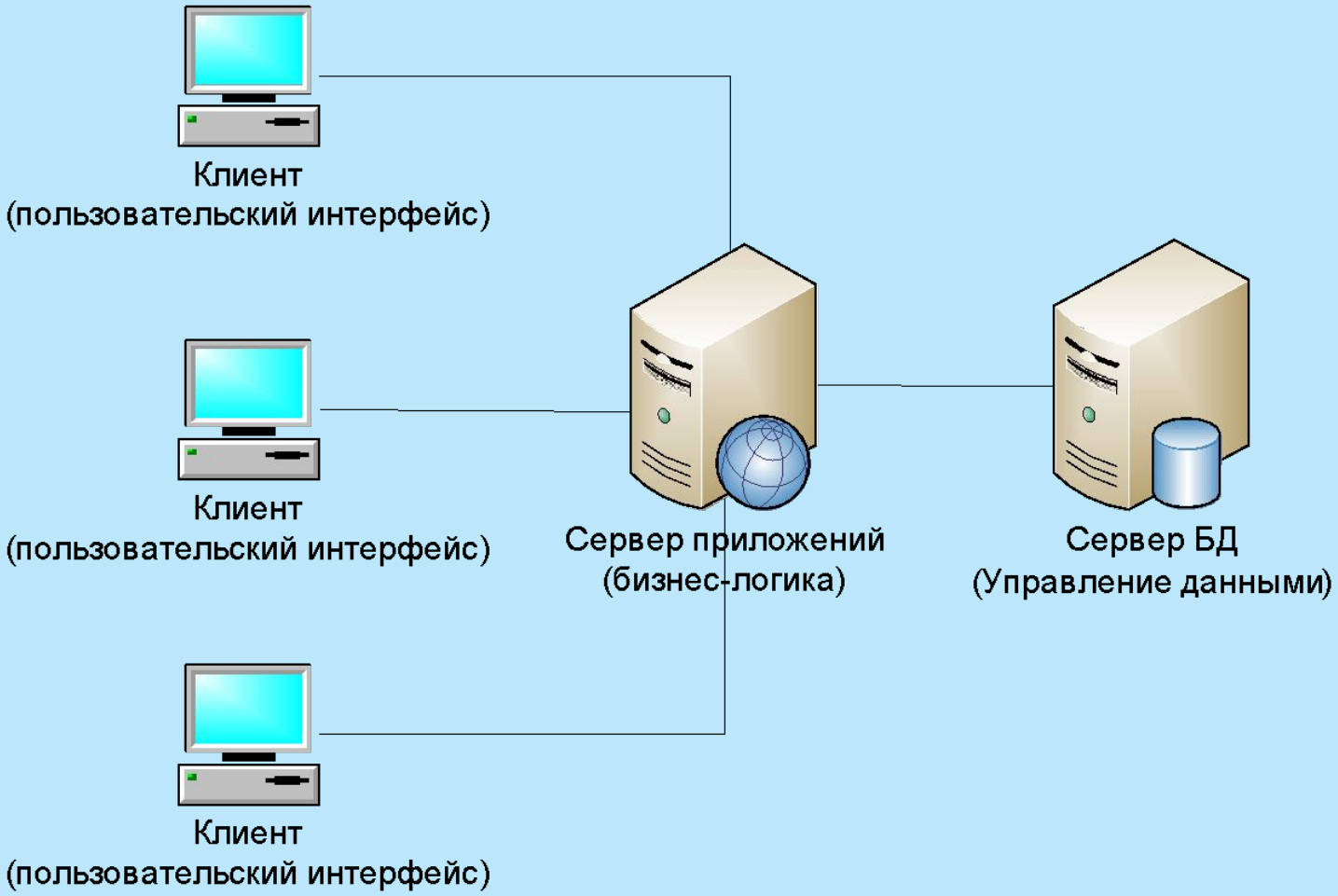
Двухуровневая архитектура «клиент-сервер»



Двухуровневая архитектура «клиент-сервер»

- **Достоинства:**
  - возможность распределить функции вычислительной системы между несколькими независимыми компьютерами
  - все данные хранятся на защищенном сервере
  - поддержка многопользовательской работы
  - гарантия целостности данных
- **Недостатки:**
  - неработоспособность сервера может сделать неработоспособной всю вычислительную сеть
  - сложное администрирование
  - высокая стоимость оборудования
  - бизнес логика приложений осталась в клиентском ПО

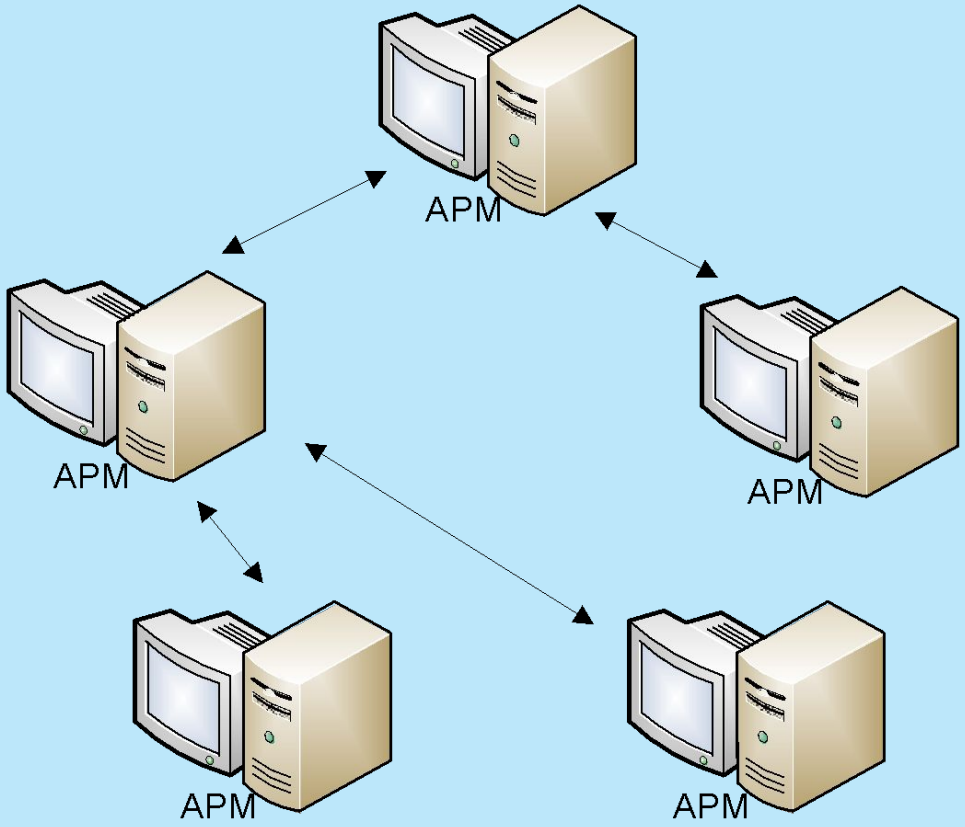
Многоуровневая архитектура «клиент-сервер»



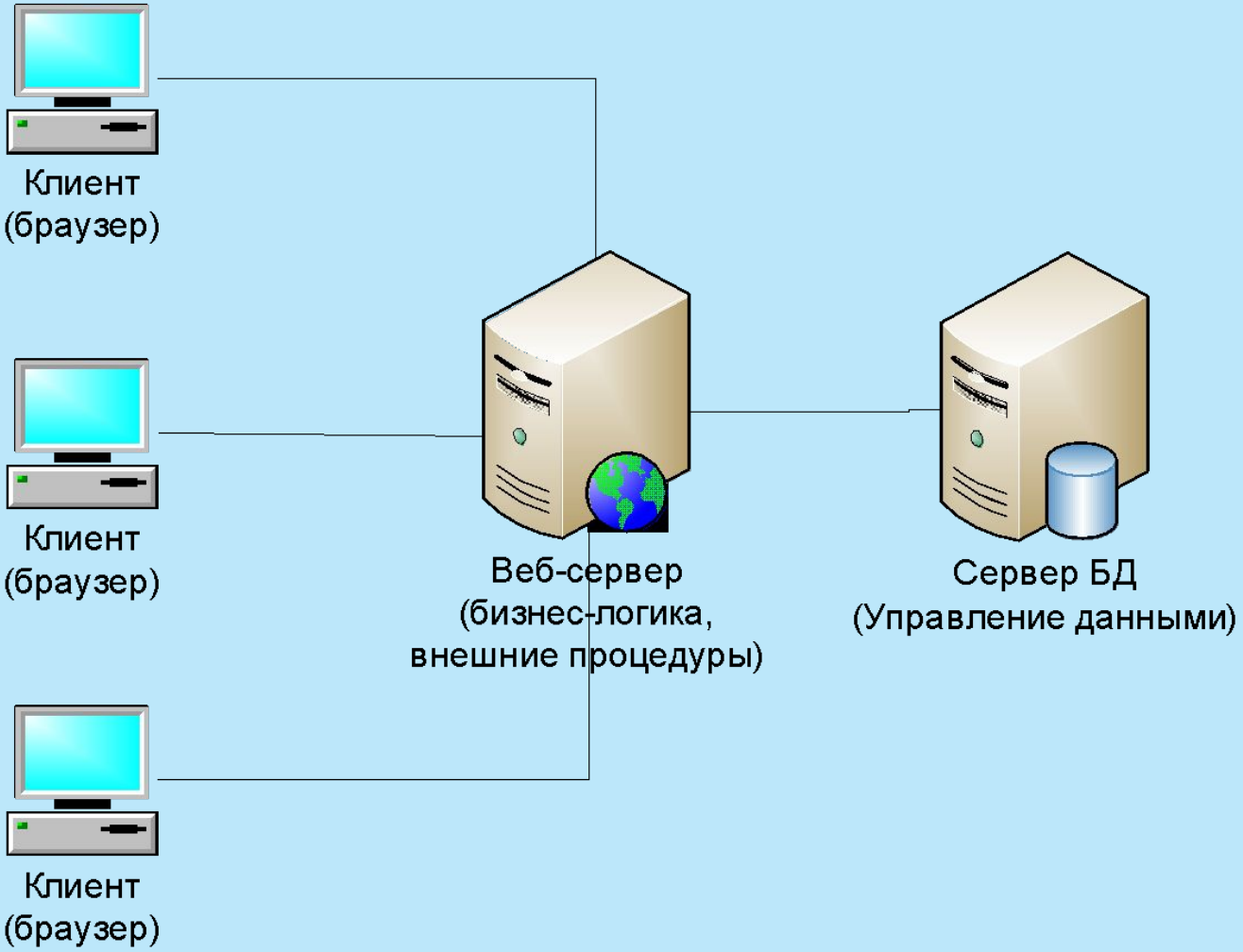
Многоуровневая архитектура «клиент-сервер»

- **Достоинства:**
  - клиентское ПО не нуждается в администрировании
  - масштабируемость
  - конфигурируемость
  - высокая безопасность и надежность
  - низкие требования к скорости канала между терминалами и сервером приложений
  - низкие требования к производительности и техническим характеристикам терминалов
- **Недостатки:**
  - сложность администрирования и обслуживания
  - более высокая сложность создания приложений
  - высокие требования к производительности серверов приложений и сервера базы данных
  - высокие требования к скорости канала (сети) между сервером базы данных и серверами приложений

Архитектура распределенных систем



Архитектура Веб-приложений



Архитектура Веб-приложений

- Отсутствие необходимости использовать дополнительное ПО на стороне клиента
- Возможность подключения практически неограниченного количества клиентов
- Централизованное место хранения данных
- Недоступность при отсутствии работоспособности сервера или каналов связи
- Достаточно низкая скорость Веб-сервера и каналов передачи данных

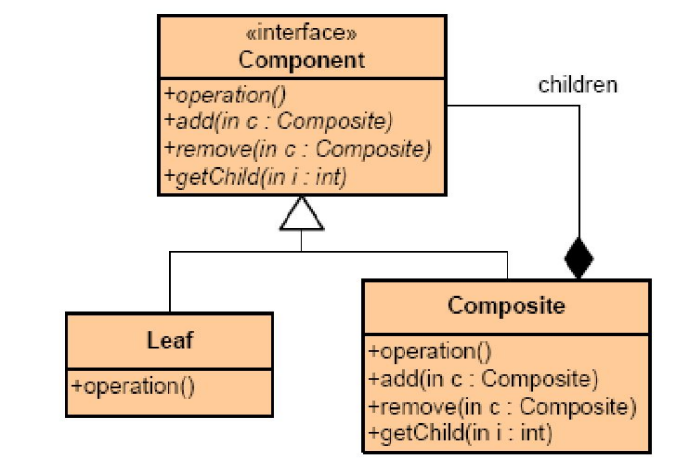
Сервис-ориентированная архитектура

- **Сервис-ориентированная архитектура (SOA)** – модульный подход к разработке программного обеспечения, основанный на использовании сервисов со стандартизированными интерфейсами
- **Принципы SOA:**
  - архитектура не привязана к какой-то определенной технологии
  - независимость организации системы от используемой вычислительной платформы
  - независимость организации системы от применяемых языков программирования
  - использование сервисов, независимых от конкретных приложений, с единообразными интерфейсами доступа к ним
  - организация сервисов как слабосвязанных компонентов для построения систем

Структурные паттерны классов/объектов

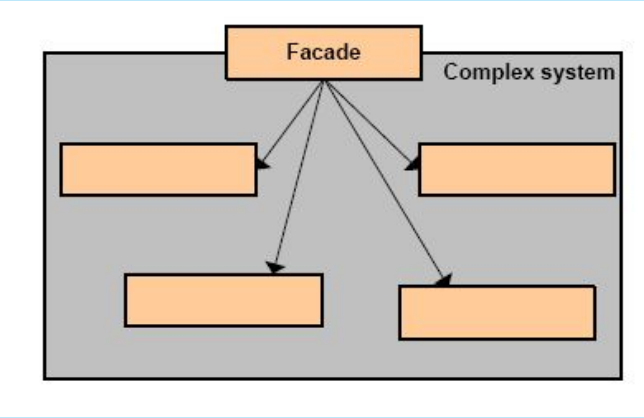
- Адаптер (Adapter) – GoF
- Декоратор (Decorator), Оболочка (Wrapper) – GoF
- Заместитель (Proxy) или Суррогат (Surrogate) – GoF
- Информационный эксперт (Information Expert) – GRASP
- Компоновщик (Composite) – GoF
- Мост (Bridge), Handle (описатель),Тело (Body) – GoF
- Низкая связанность (Low Coupling) – GRASP
- Приспособленец (Flyweight) – GoF
- Устойчивый к изменениям (Protected Variations) – GRASP
- Фасад (Facade) – GoF

Компоновщик (Composite)



- **Проблема**: Как обрабатывать группу или композицию структур объектов одновременно?
- **Решение**: Решение заключается в определении классов для композитных и атомарных объектов таким образом, чтобы они реализовывали один и тот же интерфейс, а также имели ссылки на своих детей, и, возможно, на родителя.

Фасад (Facade)

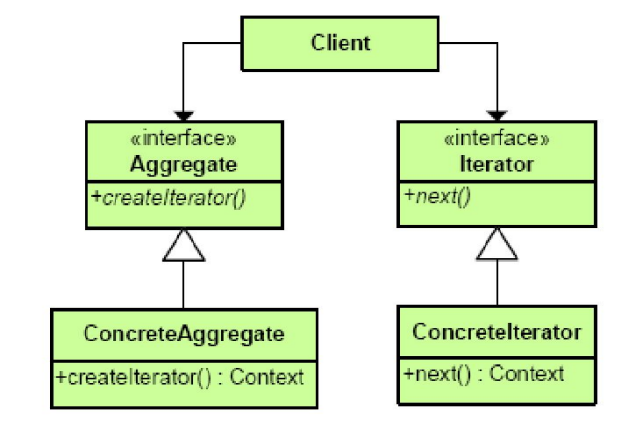


- **Проблема**: Как обеспечить унифицированный интерфейс с набором разрозненных реализаций или интерфейсов, например, с подсистемой, если нежелательно высокое связывание с этой подсистемой или реализация подсистемы может измениться?
- **Решение**: Определить одну точку взаимодействия с подсистемой – фасадный объект, обеспечивающий общий интерфейс с подсистемой и возложить на него обязанность по взаимодействию с ее компонентами. Фасад – это внешний объект, обеспечивающий единственную точку входа для служб подсистемы. Реализация других компонентов подсистемы закрыта и не видна внешним компонентам.

Поведенческие паттерны классов/объектов

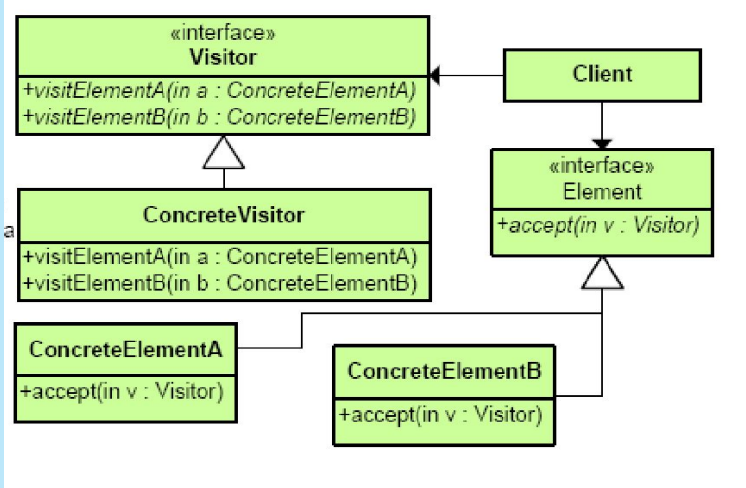
- Интерпретатор (Interpreter) – GoF
- Итератор (Iterator) или Курсор (Cursor) – GoF
- Команда (Command), Действие (Action) или Транзакция (Транзакция) – GoF
- Наблюдатель (Observer), Опубликовать – подписаться (Publish – Subscribe) или Delegation Event Model – GoF
- Не разговаривайте с неизвестными (Don't talk to strangers) – GRASP
- Посетитель (Visitor) – GoF
- Посредник (Mediator) – GoF
- Состояние (State) – GoF
- Стратегия (Strategy) – GoF
- Хранитель (Memento) – GoF
- Цепочка обязанностей (Chain of Responsibility) – GoF
- Шаблонный метод (Template Method) – GoF
- Высокое зацепление (High Cohesion) – GRASP
- Контроллер (Controller) – GRASP
- Полиморфизм (Polymorphism) – GRASP
- Искусственный (Pure Fabrication) – GRASP
- Перенаправление (Indirection) – GRASP

Итератор (Iterator)



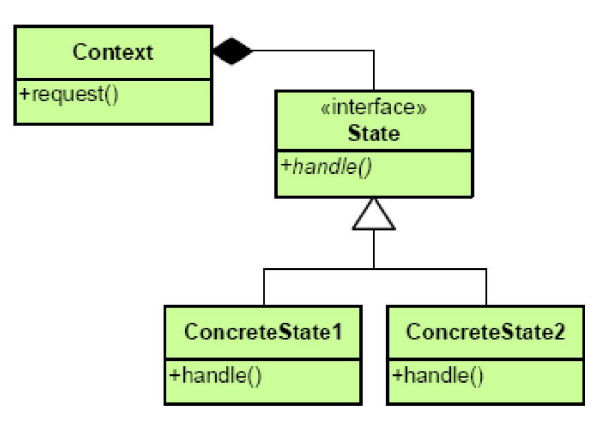
- **Проблема**: Составной объект, например, список, должен предоставлять доступ к своим элементам (объектам), не раскрывая их внутреннюю структуру, причем перебирать список требуется по-разному в зависимости от задачи.
- **Решение**: Создается класс «Итератор», который определяет интерфейс для доступа и перебора элементов, «КонкретныйИтератор» реализует интерфейс класса «Итератор» и следит за текущей позицией при обходе «Агрегата». «Агрегат» определяет интерфейс для создания объекта – итератора. «КонкретныйАгрегат» реализует интерфейс создания итератора и возвращает экземпляр класса «КонкретныйИтератор», «КонкретныйИтератор» отслеживает текущий объект в агрегате и может вычислить следующий объект при переборе.

Посетитель (Visitor)



- **Проблема:** Над каждым объектом некоторой структуры выполняется операция. Определить новую операцию, не изменяя классы объектов.
- **Решение**: Клиент, использующий данный паттерн, должен создать объект класса «КонкретныйПосетитель», а затем посетить каждый элемент структуры. «Посетитель» объявляет операцию «Посетить» для каждого класса «КонкретныйЭлемент» (имя и сигнатура данной операции идентифицируют класс, элемент которого посещает «Посетитель» – то есть, посетитель может обращаться к элементу напрямую). «КонкретныйПосетитель» реализует все операции, объявленные в классе «Посетитель». Каждая операция реализует фрагмент алгоритма, определенного для класса соответствующего объекта в структуре.

Состояние (State)

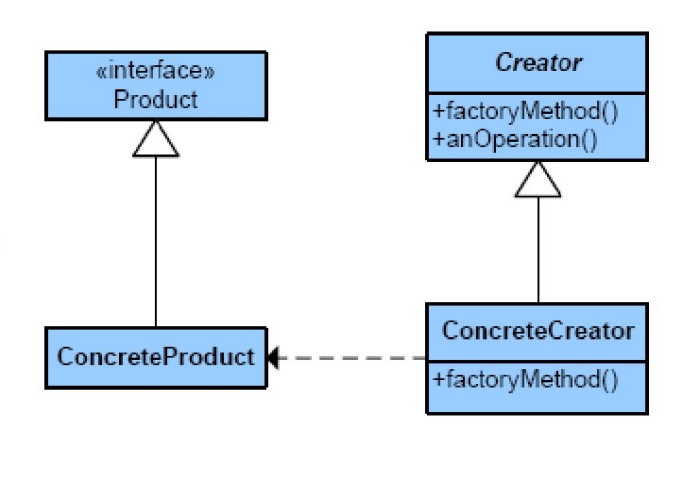


- **Проблема**: Варьировать поведение объекта в зависимости от его внутреннего состояния
- **Решение**: Класс «Контекст» делегирует зависящие от состояния запросы текущему объекту «КонкретноеСостояние» (хранит экземпляр подкласса «КонкретноеСостояние», которым определяется текущее состояние), и определяет интерфейс, представляющий интерес для клиентов. «КонкретноеСостояние» реализует поведение, ассоциированное с неким состоянием объекта «Контекст». «Состояние» определяет интерфейс для инкапсуляции поведения, ассоциированного с конкретным экземпляром «Контекста».

Порождающие паттерны классов/объектов

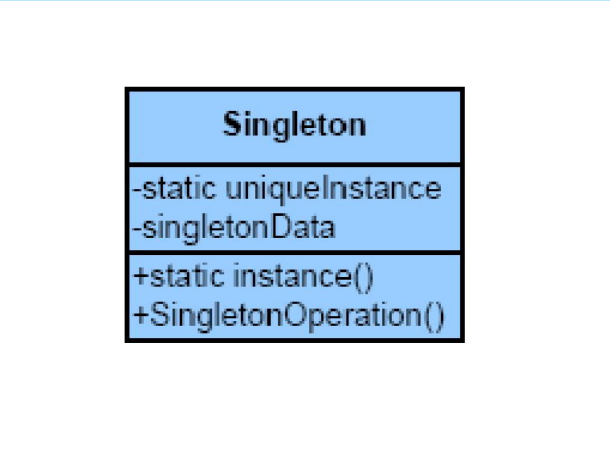
- Абстрактная фабрика (Abstract Factory, Factory) – GoF
- Одиночка (Singleton) – GoF
- Прототип (Prototype) – GoF
- Создатель экземпляров класса (Creator) – GRASP
- Строитель (Builder) – GoF
- Фабричный метод (Factory Method) или Виртуальный конструктор (Virtual Constructor) – GoF

Фабричный метод (Factory Method)



- **Проблема**: Определить интерфейс для создания объекта, но оставить подклассам решение о том, какой класс инстанциировать, то есть, делегировать инстанциирование подклассам.
- **Решение**: Абстрактный класс «Создатель» объявляет Фабричный Метод, возвращающий объект типа «Продукт» (абстрактный класс, определяющий интерфейс объектов, создаваемых фабричным методом). «Создатель» также может определить реализацию по умолчанию Фабричного Метода, который возвращает «КонкретныйПродукт». «КонкретныйСоздатель» замещает Фабричный Метод, возвращающий объект «КонкретныйПродукт». «Создатель» «полагается» на свои подклассы в определении Фабричного Метода, возвращающего объект «КонкретныйПродукт».

Одиночка (Singleton)

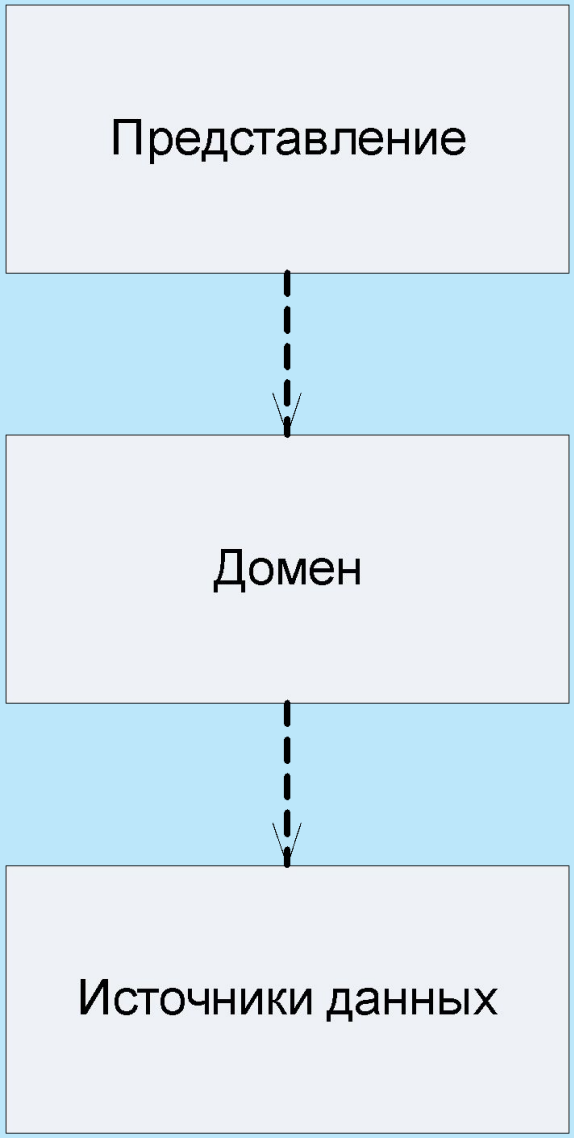


- **Проблема**: Какой специальный класс должен создавать «Абстрактную фабрику», и как получить к ней доступ? Необходим лишь один экземпляр специального класса, различные объекты должны обращаться к этому экземпляру через единственную точку доступа.
- **Решение**: Создать класс и определить статический метод класса, возвращающий этот единственный объект.

Структурные паттерны архитектуры

- Репозиторий
- Клиент/сервер
- Обьектно – ориентированный, Модель предметной области (Domain Model), модуль таблицы (Data Mapper)
- Многоуровневая система (Layers) или абстрактная машина
- Потоки данных (конвейер или фильтр)

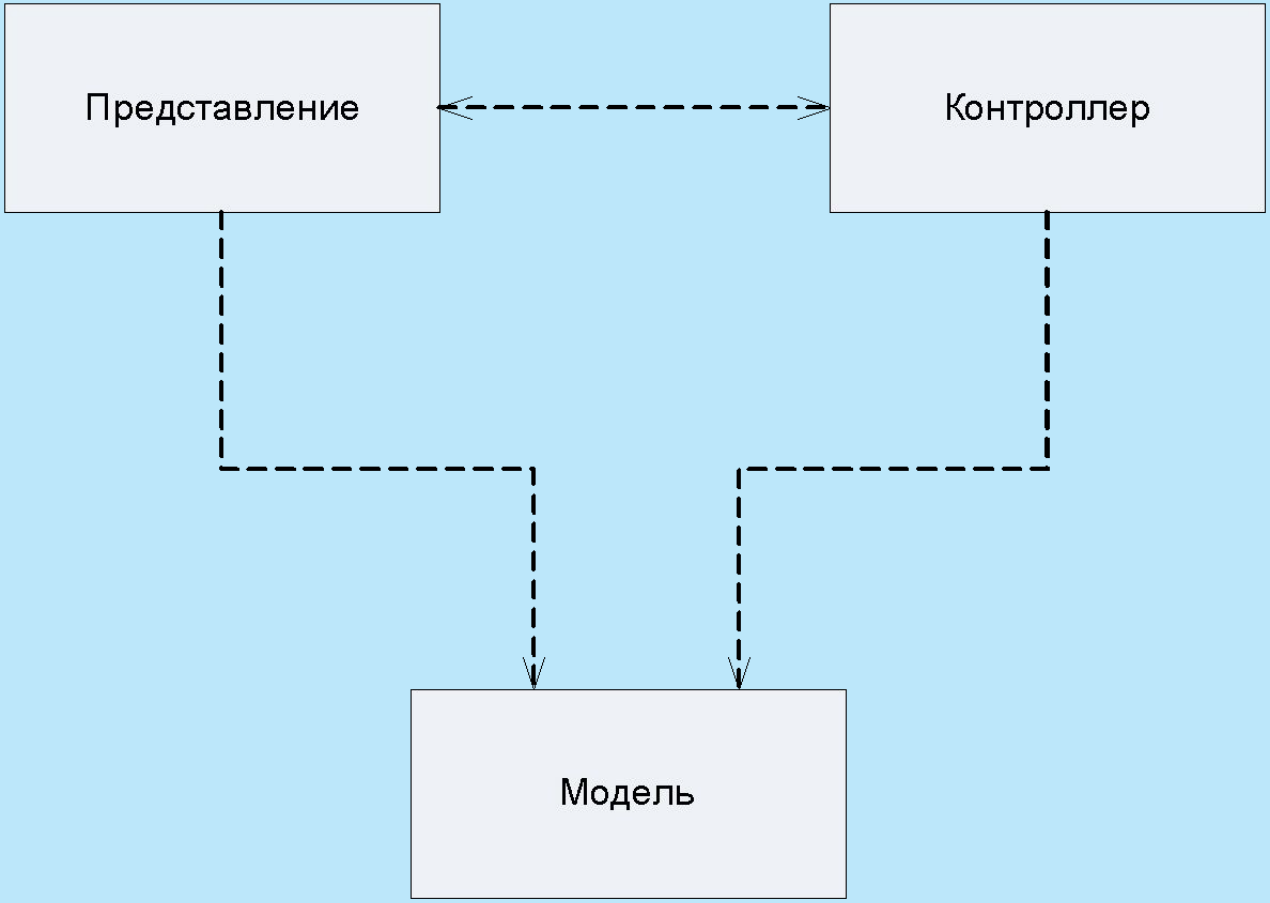
Многоуровневая система (Layers) или абстрактная машина



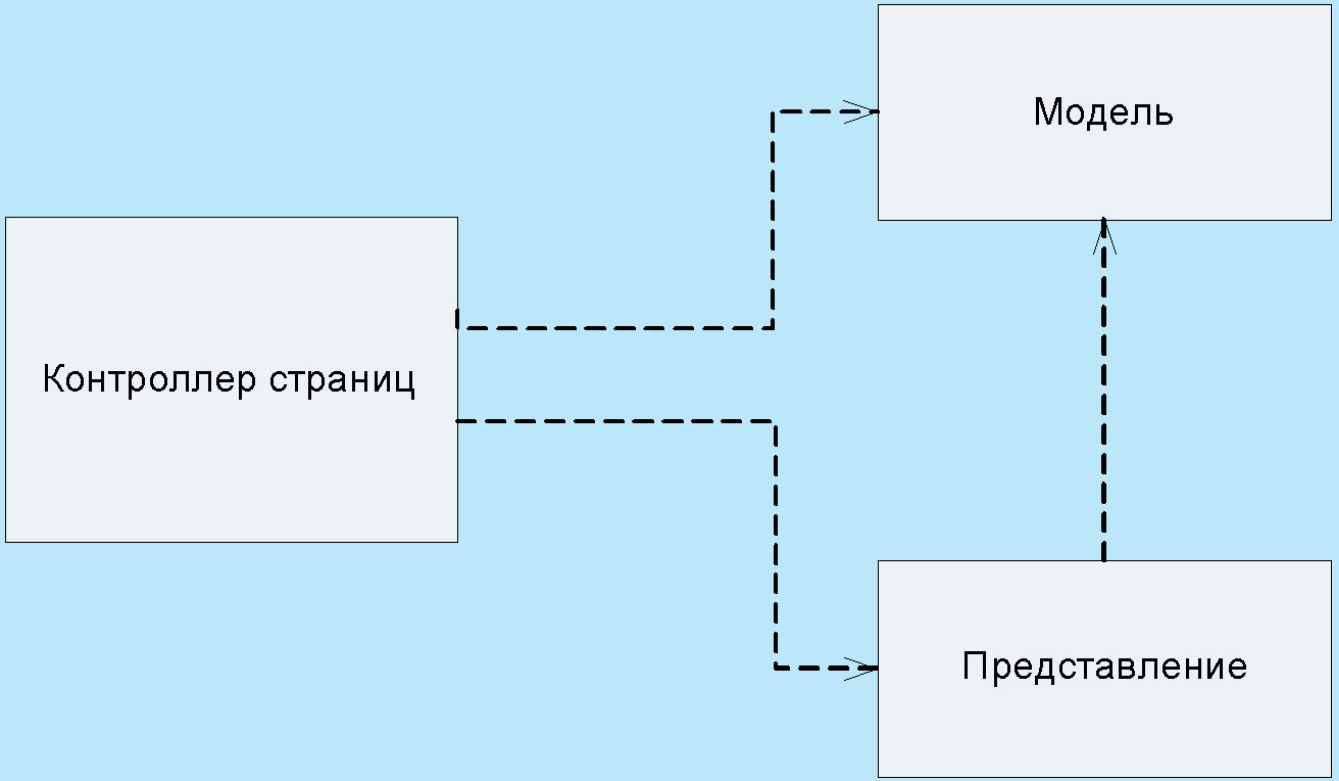
Паттерны, предназначенные для представления данных в Web

- Модель-представление-контроллер (Model View Controller)
- Контроллер страниц (Page Controller)
- Контроллер запросов (Front Controller)
- Представление по шаблону (Template View)
- Представление с преобразованием (Transform View)
- Двухэтапное представление (Two Step View)
- Контроллер приложения (Application Controller)

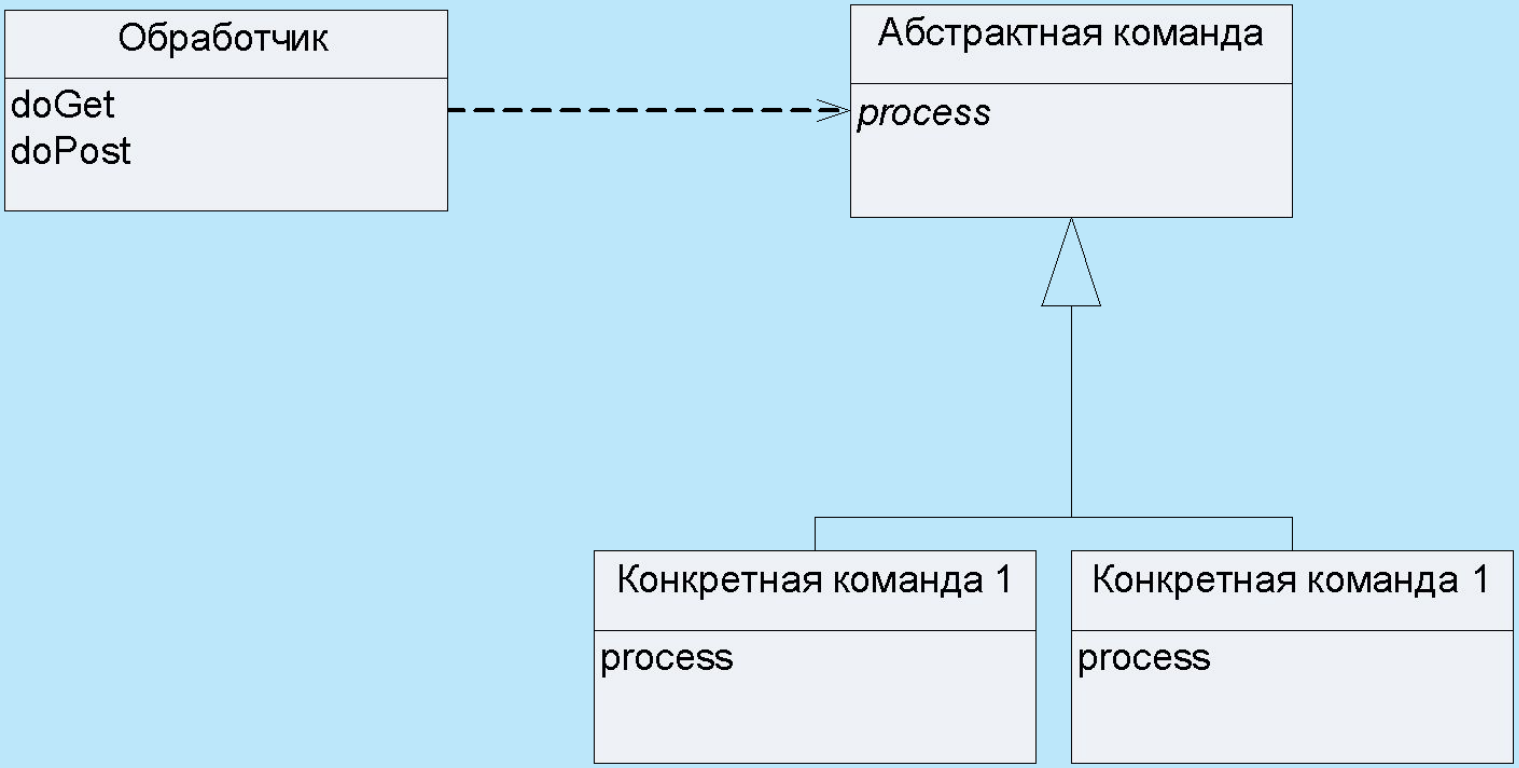
Модель-представление-контроллер (Model View Controller)



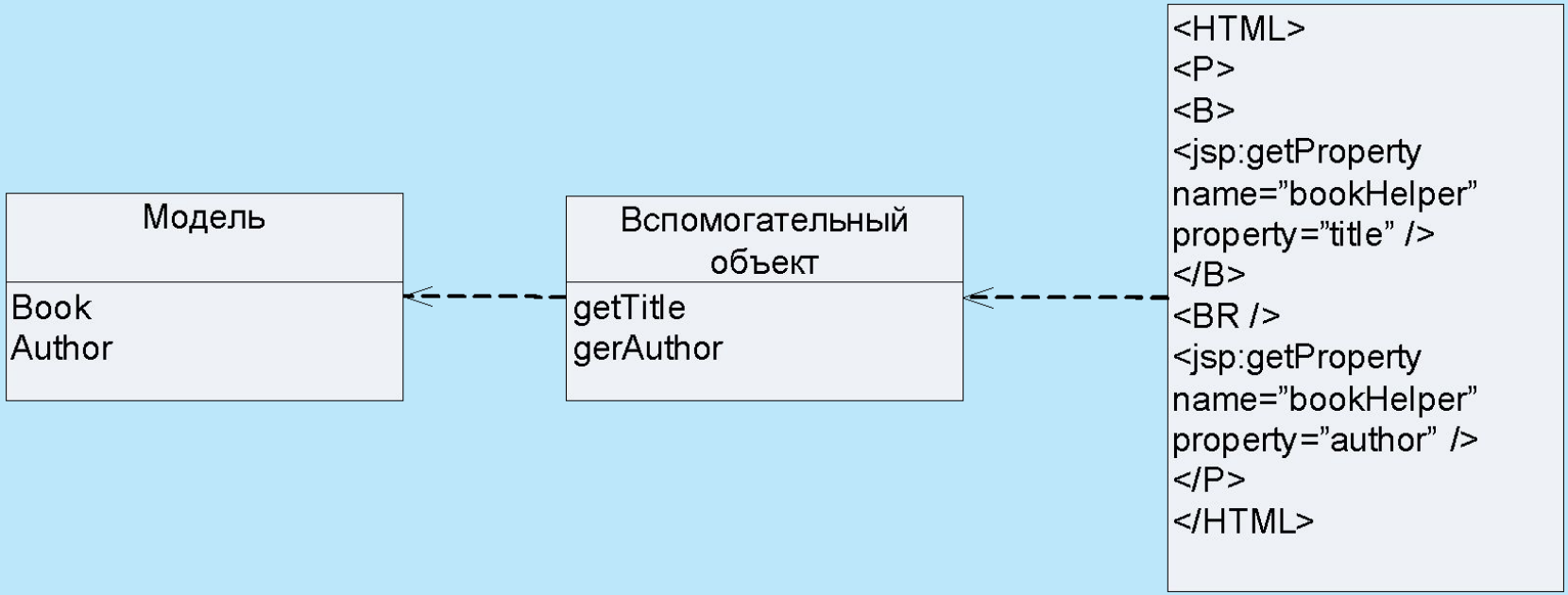
Контроллер страниц (Page Controller)



Контроллер запросов (Front Controller)



Представление по шаблону (Template View)



GET и POST

- **GET**
  - страницу всегда можно сохранить в закладках (CEO-дружелюбен)
  - быстрее POST, так как вся информация находится в заголовках
  - информация, посылаемая на сервер, всегда видима (в адресной строке)
- **POST**
  - можно отправить много информации на сервер, объем неограничен
  - отправляемая информация не показывается в адресной строке.
  - метод POST в отличие от метода GET позволяет передавать запросу файлы