MVC расшифровывается как Model—View—Controller. Это принцип построения архитектуры большого приложения, при котором оно разбивается на три части.

Первая часть содержит всю бизнес-логику приложения. Такая часть называется Модель (Model). В ней содержится код, который делает все то, для чего приложение создавалось. Эта часть наиболее независимая от остальных.

Вторая часть содержит все, что касается отображения данных пользователю. Такая часть называется Вид (View). Именно в ней содержится код, который управляет показом окон, страниц, сообщений и т.д.

Третья часть содержит код, который занимается обработкой действий пользователя. Любые действия пользователя, направленные на изменения модели, должны обрабатываться тут. Такая часть называется Controller.

Такой подход позволяет независимо делать три вещи: логику программы (Model), механизм показа всех данных программы пользователю (View), обрабатывать ввод/действия пользователя (Controller).

В Controller’е сосредоточен весь код, который принимает решение, что менять в модели в ответ на действия пользователя. Например, пользователь принял решение закрыть программу, тогда надо сохранить данные модели в файл на диск. Или пользователь ввел новые данные. Тогда надо добавить их в модель, а модель потом уведомит все View об изменении данных, чтобы они отображали только актуальное их состояние.

Модель – это самая независимая часть системы. Она не зависит от View & Controller. Модель не может использовать классы из разделов View & Controller(!).

Основное ограничение вида (View) – вид не может менять модель. Классы вида могут обращаться к модели за данными или подписываться на события, но менять модель классы View не могут.

Основное ограничение контроллера – он не занимается отображением данных. Контроллер обрабатывает действия пользователя и меняет в соответствии с ними модель.

**Builder (Строитель)**

Описание:

Используется для создания сложного объекта с использованием простых объектов. Постепенно он создает больший объект от малого и простого объекта. Позволяет изменять внутреннее представление конечного продукта.

1. **Для чего используется репозиторий, какие функции он выполняет?**

Для простого объяснения назначения шаблона проектирования "Репозиторий" приведу пример из жизни. Допустим есть блог. Для работы с материалами блога нужен функционал CRUD. Допустим для хранения данных используется MySql. Где-то в контроллерах (сервисах) созданы методы для создания, обновления и удаления материалов с sql запросами к базе. А что, если по какой-то причине понадобилось сменить MySql на MongoDB. Из этого следует, что код придется переписывать под MongoDB. Для того, чтобы код был более гибким, можно использовать шаблон Repository, который будет посредником между хранилищем и нашим контроллером (или сервисом). Тогда можно при изменении типа хранилища просто использовать репозиторий и ничего переписывать не придется.

1. **Каково назначение слоя сервисов?**

Сервисы обрабатывают объект запроса

1. **Расскажите что такое паттерн FrontController.**

Front Controller определяется как «контроллер, который обрабатывает все запросы на веб-сайт». Он стоит перед веб-приложением и делегирует запросы последующим ресурсам. Он также обеспечивает интерфейс для общего поведения, такого как безопасность, интернационализация и представление определенных представлений определенным пользователям.

Это позволяет приложению изменять свое поведение во время выполнения.

Кроме того, это помогает читать и поддерживать приложение, предотвращая дублирование кода.

1. **Как используется абстрактная фабрика в вашем проекте?**

Паттерн Абстрактная фабрика – предоставляет интерфейс создания семейств взаимосвязанных или взаимозависимых объектов без указания их конкретных классов. Шаблон реализуется созданием абстрактного класса, который представляет собой интерфейс для создания компонентов системы (например, для оконного интерфейса он может создавать окна и кнопки), затем пишутся классы, реализующие этот интерфейс.

**Применяется в случаях:** когда программа должна быть независимой от процесса и типов создаваемых новых объектов; когда необходимо создать семейства или группы взаимосвязанных объектов исключая возможность одновременного использования объектов из разных этих наборов в одном контексте

5. Расскажите о взаимодействии jsp-servlet-jsp.