Трислоен модел.MVC.

**1.Трислоен модел.MVC.**

Всички съвременни бизнес приложения имат достъп до различен вид данни като част от тяхната основна функционалност. След като релационните бази данни добиха широка популярност, IT индустрията се преориентира от еднослоен модел (mainframe) към модела клиент-сървър. При него са налице клиент, който включва презентационна логика и голямата част от бизнес логиката на приложението, както и сървър включващ в себе си хранилище за данни (data storage) и част от бизнес логиката под формата на готови и съхранени заявки към базата данни. От началото на 90-те години клиент-сървър моделът започна да излиза от употреба поради високите разходи за поддръжка и невъзможността за разпределение на отговорности при възникване на проблеми.

Трислойната архитектура обикновено се състои от презентационен слой (потребителски интерфейс) на най-високо ниво в приложението. Той служи за прякото взаимодействие с потребителя и изпращането на заявки към бизнес слоя. Не трябва да има директна връзка между този слой и слоят за бази данни. Бизнес слоят служи за обработка на данните и работните процеси. Той комуникира както с презентационния слой, така и с базите данни. Слоят за данни комуникира само с бизнес логиката и служи за съхранение данни и тяхното използване. Често отделните модули се разработват и на различни платформи. Разработена е от Джон Дж. Донован, в неговата компания OEC (Open Environment Corporation), основана в Кеймбридж, Масачузетс.

Освен стандартните предимства на модулния софтуер, трислойната архитектура е проектирана да позволява да бъде променен или заменен независимо, който и да е от трите слоя, за да отговаря на промени в изискванията или технологиите. Например, промяна на операционната система на презентационния слой би засегнала само кода за потребителският интерфейс.

**2.Принципи за правилно разделяне на компонентите на приложението по слоеве.**

Съществуват много важни и интересни концепции в платформата Джумла. Една от тях е, че Joomla компонентите използват готов дизайн, т. нар. Model-View-Controller (MVC) модел от версия 1.5 насам, излязла през 2008 г. - цели 10 години, което в Интернет е огромен период. Основната концепция на модела MVC е разделянето на приложението в отделни области: контролер, модел и изглед. Това е съществено технологично предимство спрямо кода на популярния WordPress, където всичко е смесено на едно място, и продължава да е така дори и сега**.**

* Контролерът (Controller) контролира главната програмна логика, решава кои задачи трябва да бъдат изпълнени и зарежда подходящия изглед. За по-задълбочено приложение са създадени и под-контролери, които контролират конкретни области на приложението.
* Областта модел (Model) се грижи за обработката на данните. Моделът трябва да е независим от контролера, не трябва да извиква методи в контролера или изгледа. Функцията на модела не е да определя логиката на програмирането, а да обработва данните.
* Изгледът (View) се извиква от контролера, получава данните от модела, както и обработва изхода чрез разширен шаблон.

Наличието на логическа система за организация прави кода по-лесен за разбиране. Добре написаното разширение (extension) трябва да е ясно и логично, за да може и непрограмиращият потребител да прочете кода и да придобие груба представа за това, което прави.

Едно от най-трудните неща в програмирането е въпросът от къде да се започне, което означава, че съсредоточаване върху една задача в даден момент.

**3.Слой за данни.**

Слой за достъп до данни ( DAL ) в компютърния софтуер е слой от компютърна програма , който осигурява опростен достъп до данни, съхранени в постоянно хранилище от някакъв вид, като например релационна база данни на обекти . Този акроним се използва предимно в средите на Microsoft .

Например, DAL може да върне препратка към обект (от гледна точка на обектно-ориентираното програмиране ) заедно с неговите атрибути вместо ред от полета от таблица на база данни . Това позволява клиентските (или потребителските) модули да бъдат създадени с по-високо ниво на абстракция . Този вид модел може да бъде реализиран чрез създаване на клас методи за достъп до данни, които директно препращат към съответния набор от съхранени процедури в базата данни. Друга реализация може потенциално да извлича или записва записи във или от файлова система. DAL скрива тази сложност на основното хранилище на данни от външния свят.

Например, вместо да използвате команди като вмъкване , изтриване и актуализиране за достъп до конкретна таблица в база данни, клас и няколко съхранени процедури могат да бъдат създадени в базата данни. Процедурите ще бъдат извикани от метод в класа, който ще върне обект, съдържащ исканите стойности. Или командите за вмъкване, изтриване и актуализиране могат да бъдат изпълнени в прости функции като registeruser или loginuser , съхранени в слоя за достъп до данни.

Също така методите на бизнес логиката от дадено приложение могат да бъдат нанесени към слоя за достъп до данни. Така например, вместо да прави заявка в база данни, за да извлече всички потребители от няколко таблици, приложението може да извика един метод от DAL, който абстрахира тези извиквания на базата данни.

Приложенията, използващи слой за достъп до данни, могат да бъдат или зависими от сървъра на базата данни, или независими. Ако слоят за достъп до данни поддържа множество типове бази данни, приложението може да използва каквито и бази данни да може да разговаря DAL. И при двата случая наличието на слой за достъп до данни осигурява централизирано местоположение за всички повиквания към базата данни и по този начин улеснява пренасянето на приложението към други системи за бази данни (ако приемем, че 100% от взаимодействието с базата данни се извършва в DAL за даден приложение).

**4.Слой за услуги.**

В интелигентните мрежи (IN) и клетъчните мрежи сервизният слой е концептуален слой в рамките на архитектурата на доставчик на мрежова услуга. Той има за цел да предостави междинен софтуер , който обслужва услуги и приложения с добавена стойност на трети страни на по-висок приложен слой . Слоят на услугата предоставя сървъри с възможности , притежавани от доставчик на телекомуникационни мрежови услуги, достъпни чрез отворени и сигурни интерфейси за програмиране на приложения (API) от сървъри на приложния слой, притежавани от доставчици на съдържание от трети страни . Сервизният слой също така осигурява интерфейс към основните мрежи на по-нисък ресурсен слой. [1]Долните слоеве могат също да бъдат наречени контролен слой и транспортен слой (транспортният слой също се нарича слой за достъп в някои архитектури).

Концепцията за ниво на услугата се използва в контексти като интелигентни мрежи (IN), WAP , 3G и IP мултимедийна подсистема (IMS). Той е дефиниран в модела 3GPP Open Services Architecture (OSA), който използва повторно идеята за Parlay API за сървъри на трети страни.

В софтуерния дизайн, например ориентираната към услуги архитектура , концепцията за ниво на услугата има различно значение.

**5.Слой за потребителски интерфейс.**

Слоят на потребителския интерфейс представлява всички събития (и свързани неща), случващи се между браузъра и сървъра. Има много събития, включително, но не само, следните:

* Задействане на HTTP заявката
* Получаване на отговора
* Изтегляне на ресурсите
* Изобразяването им в браузъра
* Всякакво изпълнение на JavaScript код