سطح پیشرفته

رابطه بین مدل منطقی و فیزیکی در معماری نرم‌افزار

بخش سوم

مهندس شهرام رمضانی
Shahram.ramezani@yahoo.com

بعضی از معماری‌ها قصد دارند که مفاهیم مدل منطقی n-layer و مدل فیزیکی n-tier را ادغام کنند. چنین تصمیمی نویدبخش طراحی ساده‌تری است اما در عمل کاربردی نخواهد بود، زیرا منجر به استفاده از معماری‌های منطقی یا فیزیکی ناکارآمد در طراحی نرم‌افزار خواهد شد که انطباق کامل با نیازهای برنامه‌نویس نخواهد داشت.

مدل منطقی

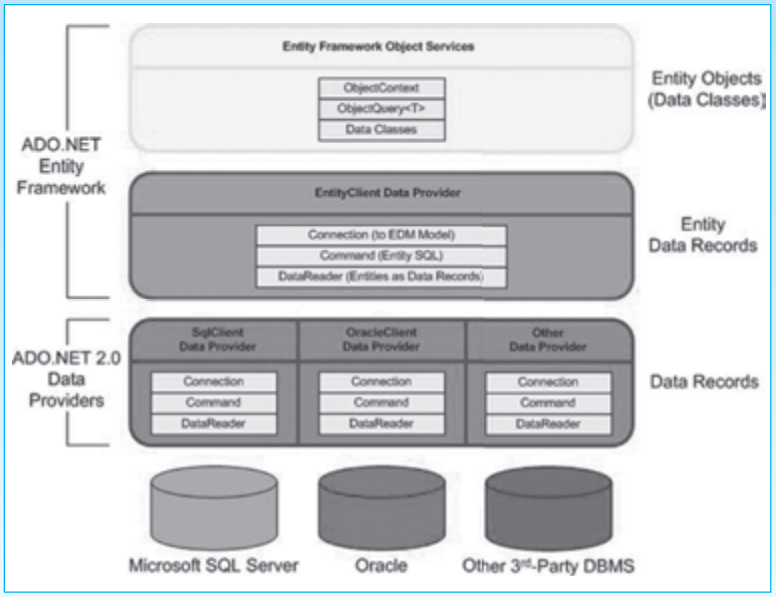
آغاز فرایند تولید نرم‌افزار با یک معماری منطقی به واسطه شفاف‌سازی نقش اجزای تشکیل‌دهنده سیستم و تفکیک عملکرد هر یک، امکان فعالیت تیمی برنامه‌نویسان در روند تولید و سهولت در نگهداری سیستم را به ارمغان می‌آورد.

این معماری منطقی جهت انعطاف‌پذیری در انتخاب معماری فیزیکی متناسب باید از لایه‌های کافی برخوردار باشد.

امروزه تفکیک انجام‌شده توسط مدل سنتی سه لایه منطقی که مبتنی‌بر لایه‌های Business Logic، Interface و Data Management است، کافی به‌نظر نمی‌رسد.

لایه Interface به دو قسمت Browser و Web Server و لایه Business Logic هم معمولاً بین Client و Web Server یا Application Server به‌صورت فیزیکی تقسیم شده‌اند.

مدل‌های معروف و پراستفاده‌ای مانند Model-View-Controller



گاهی افراد رابطه بین لایه‌های Presentation و Business Logic را برای تعاملات تحت شبکه بهینه طراحی نمی‌کنند.

به‌عنوان مثال از Data Binding استفاده می‌کنند که مستلزم تعامل بیش از حد بین لایه‌ها در قالب فراخوانی و استفاده از Propertyها، Methodها و Eventها است.

لایه‌های منطقی مجزا لزوماً در قالب لایه‌های فیزیکی متفاوت توزیع نخواهند شد و این امر باید در طراحی بهینه تعاملات بین لایه‌ها لحاظ شود که نتیجه آن ایجاد تعادل بین بازدهی و انعطاف‌پذیری خواهد بود.

و Facade-Data-Logic، لایه Business Logic سنتی را به چند قسمت تقسیم می‌کنند.

قوانین حاکم بر معماری منطقی

• معماری منطقی باید دربرگیرنده لایه‌هایی باشد که با هدف ساماندهی اجزای سیستم در نقشه کاملاً مجزا ایجاد شده‌اند.

• معماری منطقی باید حداقل به تعداد لایه‌های معماری فیزیکی آتی دارای لایه‌های تفکیک شده باشد.

با توجه به قوانین گفته شده، اکثر برنامه‌های مدرن دارای چهار تا شش لایه منطقی مجزا هستند.

ارتباط بین لایه‌ها

تفکیک سیستم به لایه‌های مختلف به معنای عملکرد نامرتبط و نامتعامل لایه‌های منطقی توزیع‌شده در لایه‌های فیزیکی متفاوت نیست.

کد هر لایه با لایه‌ای که بلافاصله قبل یا بعد از آن قرار گرفته است، در ارتباط خواهد بود.

اگر این ارتباط به‌درستی طراحی نشده باشد، امکان قرارگرفتن یک بستر شبکه‌ای بین لایه‌های فیزیکی وجود نخواهد داشت.

به‌عنوان مثال مرز شبکه‌ای بین لایه‌های Business Logic و Data باید با بازدهی بسیار بالایی طراحی شود. اکثر برنامه‌های امروزی بین لایه Data و سایر لایه‌ها دارای یک بستر شبکه‌ای بوده و فناوری‌های مدرن Data Access امروزی از بازدهی بالایی در این سناریو برخوردار هستند.

مدل فیزیکی

با اطمینان از داشتن لایه‌های منطقی کافی که منجر به انعطاف‌پذیری سیستم خواهد شد، می‌توان برنامه را در قالب لایه‌های فیزیکی مناسب با درنظرگرفتن موارد زیر تنظیم کرد:

• بازدهی
• توسعه‌پذیری
• تحمل‌پذیری خطا
• امنیت
• توزیع‌پذیری

باید در نظر داشته باشید که بعضی از فناوری‌ها طراحی فیزیکی خاصی را به برنامه تحمیل می‌کنند. به عنوان مثال استفاده از معماری فیزیکی دو لایه برای برنامه‌های Silver Light یا Windows Phone امکان‌پذیر نیست، زیرا هر نوع کد Data Access باید روی Application Server اجرا شود. در این حالت در صورتی‌که توزیع یک، سه یا چهار لایه مانعی نداشته باشد، فناوری از توزیع دو لایه ممانعت می‌کند.

تعدد لایه‌های فیزیکی منجربه موارد زیرخواهد شد:

• کاهش بازدهی
• افزایش توسعه‌پذیری
• بهبود امنیت
• تحمل‌پذیری بیشتر خطا
• توزیع‌پذیری ساده‌تر

در قسمت بعد نقش طراحی فیزیکی در موارد فوق را تشریح خواهیم کرد.