عنوان مقاله: آشنایی با تفاوت Domain Events و Integration Events

نویسنده مقاله: تیم فنی نیک آموز

تاریخ انتشار: ۳ اسفند ۱۴۰۲

منبع: https://nikamooz.com/the-difference-between-domain-events-and-integration-events/

تفاوت Domain Events و Integration Events به‌عنوان دو رویداد کلیدی از **Domain Driven Design** و **میکروسرویس ها** (Microservices)، می‌تواند کمی گیج‌کننده باشد. با شناخت دقیق نقاط تمایز این دو رویداد، می‌توانید طراحی دامنه‌محور و معماری میکروسرویس را به‌صورت کارآمد پیاده‌سازی کنید. پیش از بررسی تفاوت Domain Events با Integration Events ، در ابتدا به بررسی چیستی هر یک و نحوه پیاده‌سازی و مزایای آن‌ها می‌پردازیم و پس از آن، نقاط تمایز آن‌ها را از نقطه نظرهای مختلف شرح می‌دهیم.

### Domain Events چیست؟

Domain Events یکی از مفاهیم بنیادی در DDD و معماری میکروسرویس تلقی می‌شود و نمایان‌گر رخدادهای مهم در یک دامنه یا **Bounded Context** خاص است. به بیان ساده‌تر، Domain Event ها فعالیت‌های معنادار تجاری را دریافت می‌کنند که در یک بخش خاصی از اپلیکیشن رخ داده‌اند. تمرکز اصلی در Domain Events روی هماهنگ‌سازی تغییرات وضعیت و منطق داخلی دامنه است. معمولاً این نوع از رویداد، توسط عملیات مدل دامنه Trigger می‌شوند. به‌عنوان مثال، زمانی که یک سفارش در سیستم تجارت الکترونیک (E-Commerce) ثبت می‌شود، ممکن است یک رویداد به نام «OrderSubmitted» تریگر شود. ازنظر Scope، رویدادهای دامنه برای مدل داده خاصی معنا می‌دهد و الزامی نیست که توسط سایر بخش‌ها درک شود.

### پیاده سازی Domain Events

به‌طور کلی، برای پیاده‌سازی Domain Events ، اقدامات زیر ضروری هستند:

#### ۱- تعریف Domain Events

در این مرحله، ابتدا تغییرات چشم‌گیر و مهم را تشخیص دهید. برای این کار، لازم است رویدادهایی را نمایش دهید که رخدادهای مهمی را در مدل دامنه شما نمایش می‌دهند. این موارد می‌تواند کنش یا تغییر وضعیت یا هرنوع رخدادی باشد که می‌تواند معنای خاصی داشته باشد. در ادامه، لازم است کلاس‌های مجزایی برای هر Domain Event ایجاد کنید. این کلاس‌ها باید به‌صورت زیر باشند:

- **تغییرناپذیر (Immutable):** رویدادهای دامنه (Domain Events) موارد غیرقابل تغییری را نشان می‌دهند که از قبل در دامنه رخ داده‌اند؛ به همین دلیل، لازم است داده‌های آن غیرقابل‌تغییر باشند و زمانی که ساخته شد، امکان ویرایش آن وجود نداشته باشد.
- **توصیفی:** استفاده از نام‌های شفاف و منسجم که اهداف Event را مشخص کند، ضرورت دارد. این موضوع، خوانایی و درک کدها را برای متخصصین دامنه و توسعه‌دهندگان بهبود می‌دهد.
- **دارای داده‌های مرتبط:** رویداد باید به‌گونه‌ای باشد که تنها اطلاعات ضروری را شامل شود. این یعنی لازم است از قرار دادن جزئیات غیرضروری در Event جلوگیری شود.

#### ۲- انتشار Event ها

زمانی که یک تغییر بزرگ در منطق دامنه اتفاق می‌دهد، Domain Event مربوطه را Raise کنید. معمولاً، این موضوع ازطریق سرویس‌های دامنه یا متدهای Aggregate Root انجام می‌شود. پس از آن با پیاده‌سازی یک مکانیزم، به انتشار Event های Raise شده بپردازید. مکانیزم مورد استفاده می‌تواند یک In-Memory Publisher ساده یا Event Bus باشد.

#### ۳- رسیدگی به Event ها

موارد زیر در گام رسیدگی به رویدادها حائز اهمیت است:

- **تعریف Event Handler ها:** در این گام، کلاس‌هایی را ایجاد کنید که وظیفه رسیدگی به Domain Event های خاصی را برعهده دارند. این Handler ها، براساس رویکرد موردنظر شما، یک رابط را پیاده‌سازی می‌کند یا از کلاس Base ارث‌بری خواهند کرد.
- **ثبت Handler ها:** توجه کنید که Event Handler ها را در Event Publisher یا Event bus رجیستر کنید؛ بدین شیوه، خیالتان راحت است که در زمان Raise شدن رویدادها، نوتیفیکشن دریافت خواهید کرد.
- **پیاده‌سازی منطق Event Handler:** درون هر Handler، لازم است کنش‌هایی (Actions) تعریف شوند که باید پس از دریافت Event رخ دهند. این فرآیند، آپدیت سایر Aggregate ها و تریگر شدن مواردی همچون ارسال نوتیفیکشن یا اجرای هر منطق تجاری خاصی را دربرمی‌گیرد.

### ملاحظات جانبی

بهتر است موارد زیر به‌عنوان ملاحظات جانبی رعایت شوند:

- **تغییرناپذیر:** همان‌طور که پیش‌تر به آن اشاره شد، Domain Events باید تغییرناپذیر باشد تا بدین طریق، سازگاری داده‌ها تضمین شده و استدلال درمورد تاریخچه رخدادها ساده‌سازی شود.
- **Event Sourcing:** رویداد دامنه می‌توانند به‌عنوان پایه‌ای برای Event Sourcing استفاده شود که در آن رویدادها ذخیره می‌شوند و برای بازسازی وضعیت فعلی دامنه شما استفاده خواهند شد.
- **همگام / ناهمگام:** مشخص کنید که می‌خواهید Event ها به‌صورت همزمان درون یک تراکنش یکسان بررسی شوند یا ازطریق صف یا سیستم پیام‌دهی و به‌صورت غیرهمزمان به آن‌ها رسیدگی شود.

### مزایای Domain Events

پیش از بررسی تفاوت Domain Events و Integration Events ، قصد داریم در این بخش به مزیت‌های Domain Events اشاره کنیم:

- **اتصالات سست (Loose Coupling):** در Domain Events ، بخش‌های مختلف اپلیکیشن به شکل مستقل و دارای Loose Coupling هستند. این یعنی، کامپوننت‌ها تنها از Event هایی مطلع هستند که به آن اجزا مرتبط است. چنین مشخصه‌ای، مزایایی همچون توسعه مستقل، تسهیل نگهداری و بهبود ماژولاریتی را به‌همراه دارد.
- **واکنش‌گرایی:** به‌واسطه Domain Events ، معماری واکنشی (Reactive Architecture) امکان‌پذیر می‌شود و کامپوننت‌ها این امکان را دارند که به‌صورت ناهمگام به تغییرات واکنش نشان دهند. این ویژگی در مقایسه با تعاملات همگام و Tightly Coupled، مقیاس‌پذیری و کنش‌گرایی را بهبود می‌بخشد.
- **نظارت‌پذیری:** Domain Events یک رکورد تاریخی از رخدادهای اپلیکیشن ایجاد می‌کنند. به‌واسطه وجود این Log، امکان پیگیری مشکلات، نظارت و ارزیابی Action های سیستم وجود دارد.
- **امکان تست اجزا:** شما می‌توانید با ایزوله‌سازی رفتار و منطق دامنه درون رویدادها، اجزا را به‌راحتی و به‌صورت مستقل تست کنید؛ به‌طور کلی، چنین اقدامی به بهبود کیفیت و نگهداری کدهای شما منجر می‌شود. برای این کار از روش‌های Unit Testing و Integration Testing استفاده می‌شود.
- **بهبود مشارکت:** Domain Event ها یک زبان مشترک (Ubiquitous Language) برای متخصصین دامنه و توسعه‌دهندگان ایجاد می‌کند که به کمک آن، می‌توانند تغییرات دامنه را مورد بحث قرار دهند.

### Integration Events چیست؟

Integration Events نقش بسزایی در برقراری ارتباط‌هایی فراتر از Boundry ها ایفا می‌کنند. درحقیقت، Integration Events پیام‌هایی هستند که از مرزهای یک دامنه خاص عبور می‌کنند تا سایر سیستم‌ها خارجی یا Context ها را درمورد یک رخداد جدید در آن دامنه مطلع کنند. به بیان ساده، می‌توان Integration Events را مانند پیام‌رسانی درنظر داشت که مانند یک پل، ارتباط میان اجزای مختلف سیستم را ایجاد می‌کند. هدف از وجود آن، ایجاد ارتباط فراتر از Context ها است و معمولاً توسط Domain Events یا سایر عملیات مدل دامنه تریگر می‌شود.


### پیاده سازی Integration Events

به‌طور کلی، برای پیاده‌سازی Integration Events ، مراحل زیر را دنبال کنید:

#### ۱- تعریف رویدادها

ابتدا باید مشخص کنید دقیقاً چه رخدادهایی در درون دامنه وجود دارند که می‌خواهید با سایر Context ها یا سیستم‌ها ارتباط داشته باشند. علاوه‌بر این، لازم است ساختار داده یا همان اسکیمای Event را تعریف کنید.

#### ۲- انتخاب مکانیزم ارتباطی

اکنون یک سیستم پیام‌رسانی مناسب انتخاب کنید. Message Broker هایی همچون Apache Kafka ، RabbitMQ و همچنین راه‌حل‌های مبتنی‌بر Cloud، مانند Amazon SQS یا Azure Service Bus گزینه‌های مشهوری در این زمینه به‌حساب می‌آیند. شایان ذکر است که شما باید براساس نیازمندی‌ها و مقدار کنش‌گرایی مدنظرتان، تعیین کنید که پیام‌رسانی به‌صورت همگام یا غیرهمگام باشد.

#### ۳- انتشار رویدادها

مکانیزم‌هایی را در سرویس‌های دامنه خود ایجاد کنید تا رویدادها را زمانی که رویدادهای محرک اتفاق می‌افتند، منتشر کنند. از API های سیستم پیام‌رسانی موردنظر استفاده کنید تا پیام‌های رویداد به‌همراه اسکیمای تعریف‌شده ارسال شود.

### ۴- ثبت و طراحی Event Handler

در سیستم‌های خارجی یا Context های دریافت‌کننده، Event Subscriber هایی را پیاده‌سازی کنید که برای Event های خاصی روی سیستم پیام‌رسان انتخابی اصطلاحاً «گوش» (Listen) می‌دهند. حال باید Event Handler هایی درون Subcriber ها طراحی شود؛ به کمک آن‌ها، داده‌های دریافتی مربوط به رویداد، پردازش شده و Action های لازم در Context مربوطه اجرا می‌شوند.

### ملاحظات جانبی

موارد زیر را به‌عنوان موارد جانبی در زمینه Integration Event در نظر داشته باشید:

- **Event Handling:** شما باید مکانیزم‌هایی را برای رسیدگی به خطاهای بالقوه مربوط به زمان انتشار و دلیوری رویداد پیاده‌سازی کنید. بدین طریق، بازیابی و قابل اکتفا بودن پیام تضمین می‌شود.
- **نظارت و Logging:** سیستم‌های Logging و مانیتورینگ راه‌اندازی کنید تا جریان رویدادها پیگیری شود و مشکلات بالقوه تشخیص داده شوند.
- **امنیت:** معیارهای امنیتی مناسبی را پیاده‌سازی کنید تا از امنیت کنترل دسترسی و ارتباط‌ها برای پیام‌های رویداد اطمینان بیابید.

### مزایای Integration Events

مزیت‌های Integration Events به شرح زیر است:

- **اتصالات سست:** Integration events ویژگی Loose Coupling را بین اجزای مختلف سیستم ترویج می‌کند؛ این یعنی، سیستم‌ها و سرویس‌ها می‌توانند به‌صورت جداگانه تکامل یابند و بدون تأثیرگذاری روی سایر آن‌ها، Scale شوند.
- **مقیاس‌پذیری:** Integration Events با جداسازی (Decoupling) ارتباطات از یک سرویس خاص، افزایش مقیاس هر کامپوننت را تسهیل می‌بخشد. این موضوع به‌طور خاص، در معماری میکروسرویس کاربردی است. برای آشنایی بیشتر با این معماری، پیشنهاد می‌شود **مقاله جامع میکروسرویس** را مطالعه کنید.
- **Event Sourcing:** می‌توان Integration Events را همراه با Event Sourcing استفاده کرد تا یک رکورد غیرقابل تغییر از تغییرات وضعیت درون سیستم نگهداری کرد. بدین طریق، این امکان وجود دارد که در هر زمان، وضعیت سیستم را بازسازی کرد.
- **پردازش بلادرنگ:** به‌دلیل امکان Trigger کردن بلادرنگ Action ها در سیستم دیگر، این قابلیت وجود دارد که داده‌ها را به‌صورت آنی و بلادرنگ پردازش کرد. این موضوع در مواردی همچون تشخیص کلاه‌برداری مناسب خواهد بود.
- **انعطاف‌پذیری:** به‌واسطه Integration Events ، امکان یکپارچه‌سازی با انواع سیستم‌های خارجی و سرویس‌ها وجود دارد و شما می‌توانید براساس نیازمندی‌ها، پروتکل‌های ارتباطی و Message Broker های مختلف را به کار ببرید.

### مقایسه Integration Events و Domain Events

در این بخش قصد داریم به بررسی تفاوت Domain Events و Integration Events بپردازیم و آن‌ها را از نقطه نظرهای مختلف بررسی کنیم. هرچند هر دوی این رویدادها نقش کلیدی در طراحی دامنه محور دارا هستند، اما هر یک مشخصه‌ها و اهداف مخصوص به خود را دارند. برای درک تفاوت آن‌ها، به موارد زیر توجه کنید:

### تفاوت Integration Events و Domain Events : اهداف

یک تفاوت Integration Events و Domain Events این است که در رویداد دامنه تمرکز روی هماهنگی تغییرات وضعیت و منطق داخلی دامنه است؛ در حالی که Integration Events به‌منظور ارتباطات فراتر از Context ها و تسهیل تعامل با سایر سیستم و Context ها به کار می‌رود.

### تفاوت Integration Events و Domain Events : تریگر

معمولاً رویدادهای دامنه توسط عملیات مدل دامنه در درون همان Bounded Context تریگر می‌شوند. در صورتی که Integration Events توسط رویدادهای دامنه یا عملیات مدل دامنه دیگری تریگر خواهند شد و به‌طور خاص، برای ارتباط Cross-Context هستند.

### تفاوت Integration Events و Domain Events : حوزه

Domain Events مخصوص مدل دامنه است و الزامی ندارد که توسط سایر بخش‌های سیستم درک و شناخته شود. ازسوی دیگر، Integration Events از Domain Evets وسیع‌تر است و تعاملات بین سیستم‌ها خارجی یا Context های دیگر را دربرمی‌گیرد.

### تفاوت Integration Events و Domain Events : ویژگی های مثبت

به‌طور کلی، Domain Events مزیت‌هایی همچون بهبود Decoupling منطق دامنه، بهبود سازگاری درون Context، امکان تست منطق دامنه را دارا است. به‌واسطه Integration Events، ویژگی‌های مثبتی همچون اتصالات سست میان کامپوننت‌ها، بهبود مقیاس‌پذیری هر یک از سرویس‌ها، افزایش استحکام سیستم و انعطاف‌پذیری در یکپارچگی با سایر سیستم‌ها فراهم می‌شود.

### تفاوت Integration Events و Domain Events : پیاده سازی

یکی دیگر از انواع تفاوت Integration Events و Domain Events این است که رویدادهای دامنه درون Bounded Context پیاده‌سازی می‌شوند و در آن‌ها، از پیام‌رسانی درون حافظه‌ای یا Event Dispatcher استفاده می‌شود. در Integration Events، ملاحظات دیگری همچون قالب پیام‌رسانی، مکانیزم‌های صف و همچنین مکانیزم‌هایی برای مشکلات بالقوه مورد نیاز است.

در این بخش به بررسی برخی از مهم‌ترین نقاط تفاوت Domain Events و Integration Events پرداخته شد تا شما با درک دقیق آن‌ها، به الگوهای ارتباطی کارآمد برای سیستم خود برسید.

### جمع بندی: تفاوت Domain Events و Integration Events

به‌واسطه آشنایی با تفاوت Domain Events و Integration Events ، می‌توانید یک سیستم ساختارمند و Decoupled ایجاد کنید. درحقیقت، شما با شناختن هر یک از این رویدادها، این فرصت را دارید که تصمیمات صحیحی درخصوص نحوه استفاده آن‌ها در DDD و میکروسرویس اتخاذ کنید.