Particular Software

Welcome

NServiceBus v6 API

webinar

Particular Software

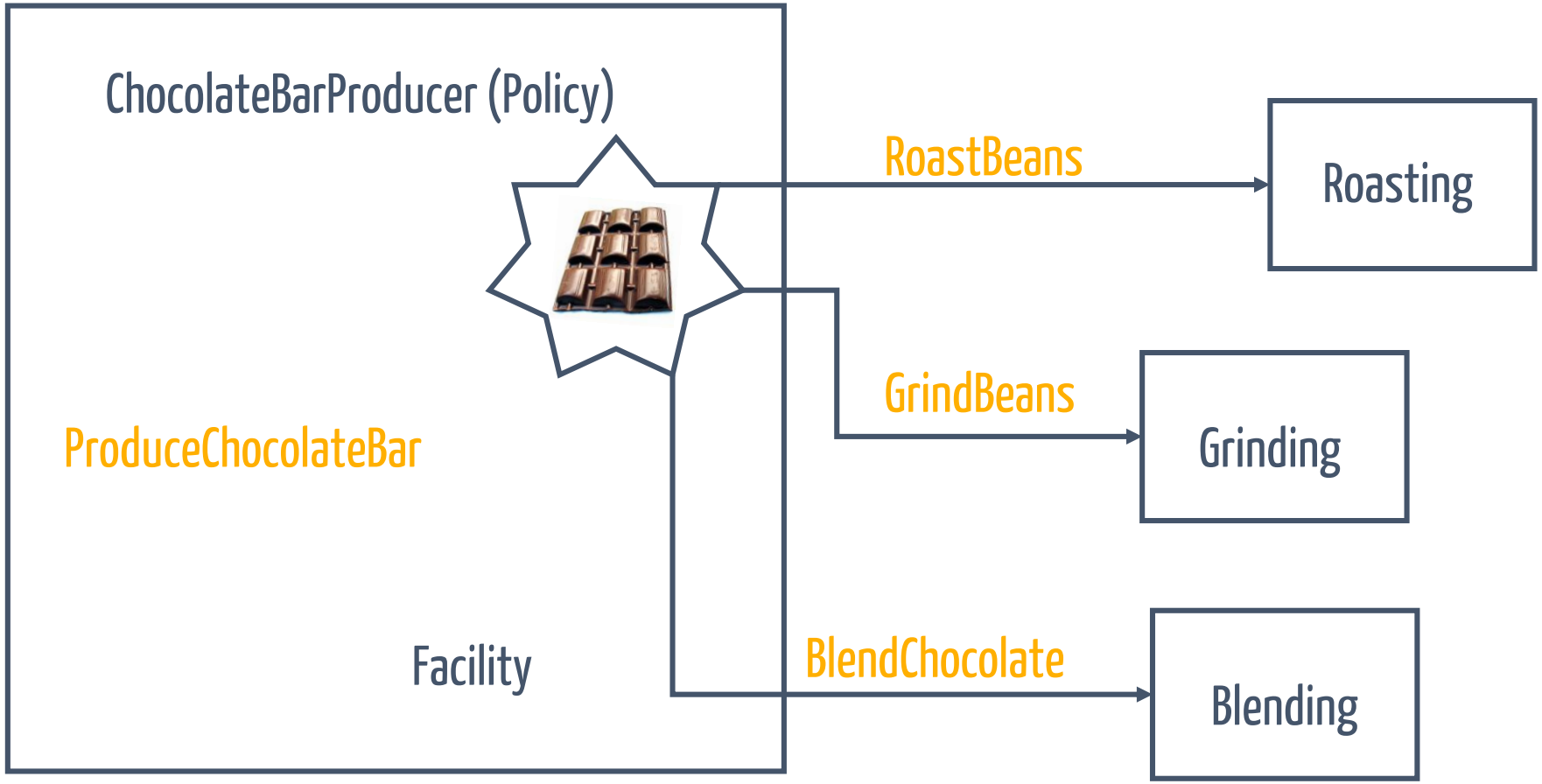
Solution Architect
Enthusiastic Software Engineer
Microsoft MVP for systems integration

@danielmarbach
particular.net/blog
planetgeek.ch

chocolate

factory

https://commons.wikimedia.org/wiki/File:Flag-map_of_Switzerland.svg


ChocolateBarProducer (Policy)
RoastBeans
Roasting
GrindBeans
Grinding
ProduceChocolateBar
Facility
BlendChocolate
Blending

await Demo

chocolate

as-a-service

```
class VanillaHandler : IHandleMessages<AcquireVanilla> {

  void Handle(AcquireVanilla message)
  {
    AcquireVanillaFromGovernmentService();
    StoreVanillaUsageInDatabase();
  }

}
```

```
class VanillaHandler : IHandleMessages<AcquireVanilla> {

  void Handle(AcquireVanilla message)
  {
    AcquireVanillaFromGovernmentAsync().Result;
    DownloadRecipeFromBlobStorageAsync().Wait();
    InsertVanillaUsageInDocumentDBAsync().Result;
    StoreTelemetryDataInEventHubAsync().Wait();
  }

}
```

```
class VanillaHandler : IHandleMessages<AcquireVanilla> {

  void Handle(AcquireVanilla message)
  {
    await AcquireVanillaFromGovernmentServiceAsync();
    await DownloadRecipeFromBlobStorageAsync();
    await InsertVanillaUsageInDocumentDBAsync();
    await StoreTelemetryDataInEventHubAsync();
  }

}
```

Error CS4033 The 'await' operator can only be used within an async method. Consider marking this method with the 'async' modifier and changing its return type to 'Task'.

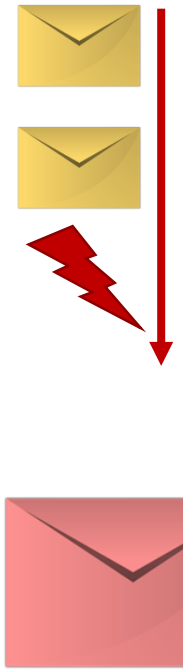
ALT + ENTER

```
class VanillaHandler : IHandleMessages<AcquireVanilla> {

  async void Handle(AcquireVanilla message)
  {
    await AcquireVanillaFromGovernmentServiceAsync();
    await DownloadRecipeFromBlobStorageAsync();
    await InsertVanillaUsageInDocumentDBAsync();
    await StoreTelemetryDataInEventHubAsync();
  }

}
```

Void Handler

```
AcquireVanillaFromGovernmentServiceAsync().Result;
DownloadRecipeFromBlobStorageAsync().Wait();
InsertVanillaUsageInDocumentDBAsync().Result;
StoreTelemetryDataInEventHubAsync().Wait();
```

Void Handler



```
AcquireVanillaFromGovernmentServiceAsync().Result;
DownloadRecipeFromBlobStorageAsync().Wait();
InsertVanillaUsageInDocumentDBAsync().Result;
StoreTelemetryDataInEventHubAsync().Wait();
```


Async void Handler

```
await AcquireVanillaFromGovernmentServiceAsync();

await DownloadRecipeFromBlobStorageAsync();

await InsertVanillaUsageInDocumentDBAsync();

await StoreTelemetryDataInEventHubAsync();
```

Async void Handler

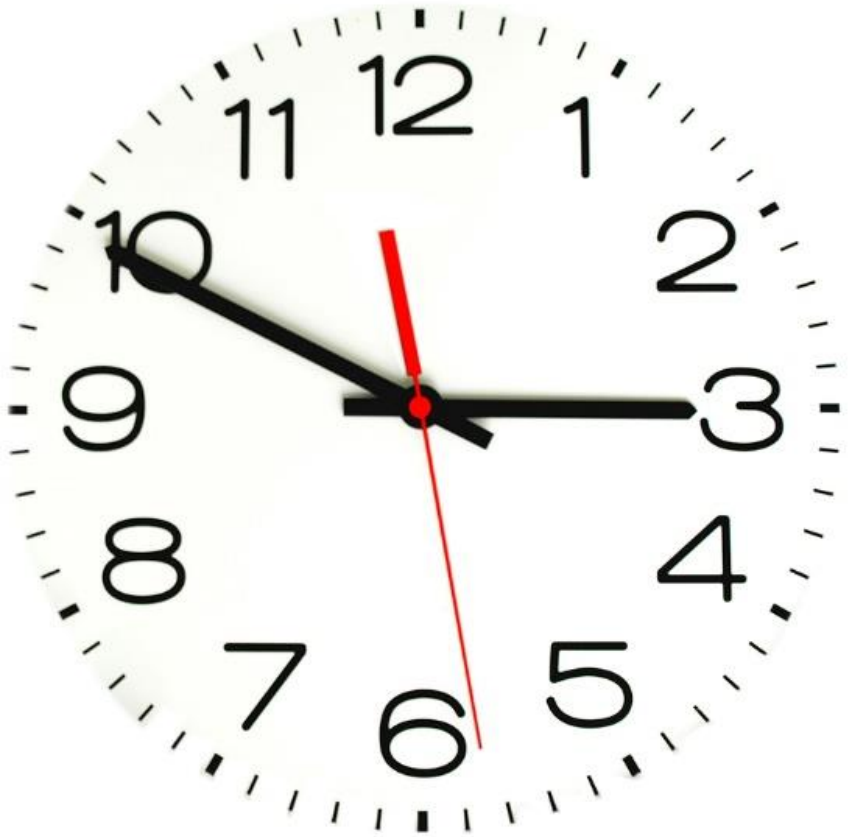
```
await AcquireVanillaFromGovernmentServiceAsync();
await DownloadRecipeFromBlobStorageAsync();
await InsertVanillaUsageInDocumentDBAsync();
await StoreTelemetryDataInEventHubAsync();
```

Engine

Version 5

async

it's time



particular.net/blog/async-await-its-time

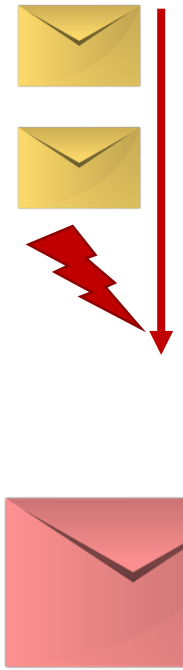
```
class VanillaHandler : IHandleMessages<AcquireVanilla> {

  Task Handle(AcquireVanilla message)
  {
  }

}
```

```
class VanillaHandler : IHandleMessages<AcquireVanilla> {

  async Task Handle(AcquireVanilla message)
  {
    await AcquireVanillaFromGovernmentServiceAsync();
  }

}
```

```
class VanillaHandler : IHandleMessages<AcquireVanilla> {

  async Task Handle(AcquireVanilla message)
  {
    await AcquireVanillaFromGovernmentServiceAsync();
    await DownloadRecipeFromBlobStorageAsync();
    await InsertVanillaUsageInDocumentDBAsync();
    await StoreTelemetryDataInEventHubAsync();
  }

}
```

```
class VanillaHandler : IHandleMessages<AcquireVanilla> {

  Task Handle(AcquireVanilla message)
  {
    var t1 = AcquireVanillaFromGovernmentServiceAsync();
    var t2 = DownloadRecipeFromBlobStorageAsync();
    var t3 = InsertVanillaUsageInDocumentDBAsync();
    var t4 = StoreTelemetryDataInEventHubAsync();
    return Task.WhenAll(t1, t2, t3, t4);
  }

}
```

Async Task Handler



```
await AcquireVanillaFromGovernmentServiceAsync();
await DownloadRecipeFromBlobStorageAsync();
await InsertVanillaUsageInDocumentDBAsync();
await StoreTelemetryDataInEventHubAsync();
```


But there are not only asynchronous handlers

```
class AnotherHandler : IHandleMessages<AnotherMessage> {

  Task Handle(AnotherMessage message)
  {
    DoSomethingSynchronous();
  }

}
```

```
class AnotherHandler : IHandleMessages<AnotherMessage> {

  async Task Handle(AnotherMessage message)
  {
    DoSomethingSynchronous();
  }

}
```

Warning CS1998 This async method lacks 'await' operators and will run synchronously. Consider using the 'await' operator to await non-blocking API calls, or 'await Task.Run(...)' to do CPU-bound work on a background thread.

```
class AnotherHandler : IHandleMessages<AnotherMessage> {

  Task Handle(AnotherMessage message)
  {
    DoSomethingSynchronous();
    return Task.CompletedTask; // for .NET 4.6

    return Task.FromResult(0); // for .NET 4.5
  }

}
```

Ambient

state

The dangers of ThreadLocal

Written by Daniel Marbach on December 01, 2015 • Comments

Languages and frameworks evolve. We as developers have to learn new things constantly and unlearn already-learned knowledge. Speaking for myself, unlearning is the most difficult part of continuous learning. When I first came into contact with multi-threaded applications in .NET, I stumbled over the ThreadStatic attribute. I made a mental note that this attribute is particularly helpful when you have static fields that should not be shared between threads. At the time that the .NET Framework 4.0 was released, I discovered the ThreadLocal class and how it does a better job assigning default values to thread-specific data. So I unlearned the `ThreadStaticAttribute`, favoring instead `ThreadLocal<T>`.

Fast forward to some time later, when I started digging into `async/await`. I fell victim to a belief that thread-specific data still worked. So I was wrong, again, and had to unlearn, again! If only I had known about AsyncLocal earlier.

Let's learn and unlearn together!

Community

Courses and Events

Blog

Discussion group

Community champions

Ambient

state

Context

floating state

www.insideevs.com

```
class VanillaHandler : IHandleMessages<AcquireVanilla> {

  async Task Handle(AcquireVanilla message, IMessageHandlerContext context)
  {
    var vanilla = await webService.AcquireVanilla(message.LotNumber);

    await context.Publish(new VanillaAcquired());
  }

}
```

await Demo

APIs are in early preview

Guidance

docs.particular.net

Engine

Version 6

www.hdwallbackgrounds.com

Slides, Links...

github.com/danielmarbach/03-10-2016-AsyncWebinar

Particular Software

await Q & A

Particular
Software

Thanks