



## Week 2: Development Tasks

# Unit 5: Custom Logic

## Service API overview

1

### Construct, Reflection API

API mostly used when bootstrapping provided services or when connecting to required ones

2

### Querying API

Synchronous APIs used by consumers to interact with a service

3

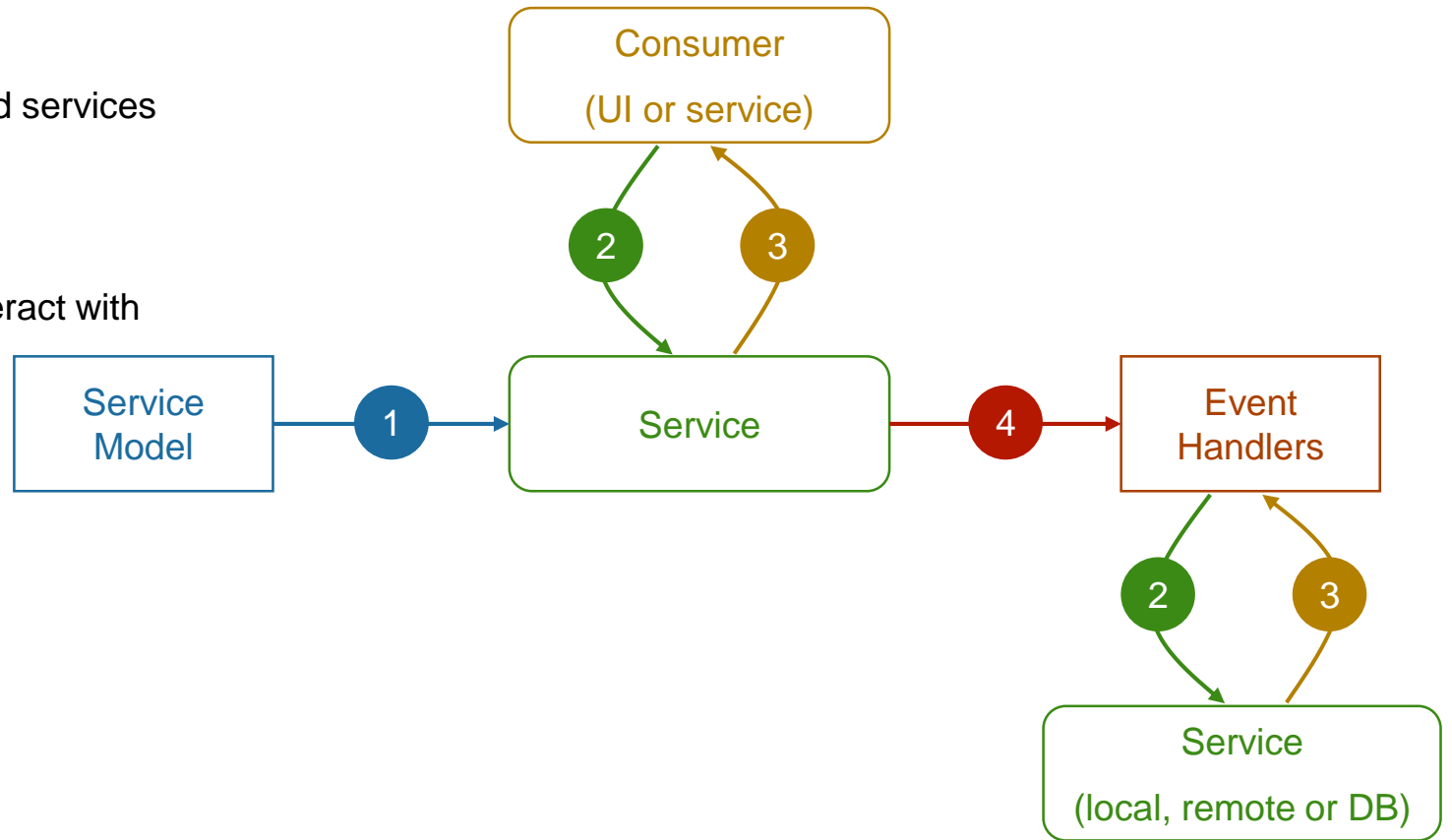
### Messaging API

Asynchronous APIs used by consumers to interact with a service

4

### Event Handling

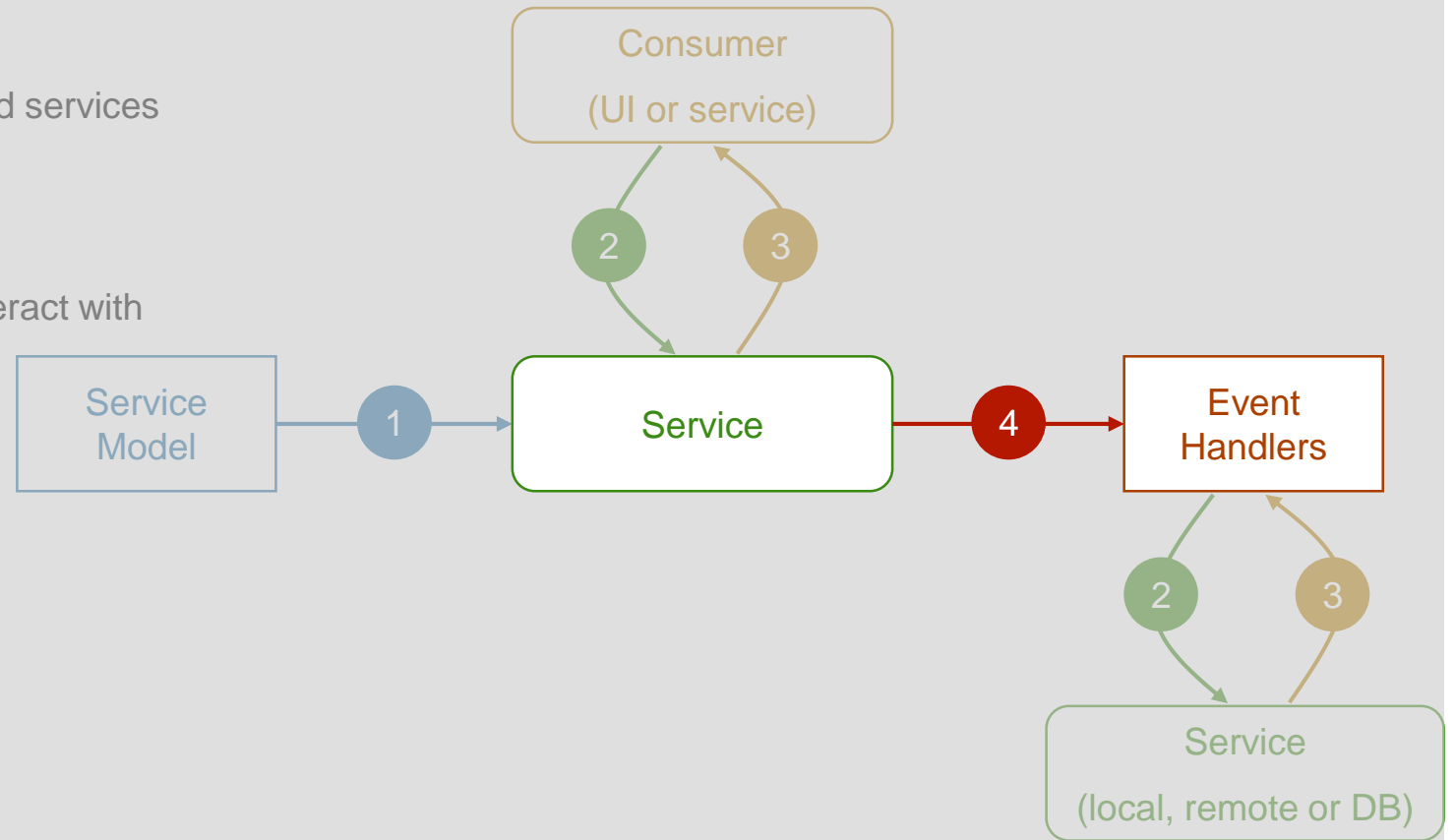
Used to register custom event handler



## Custom Logic

# Service API overview

- 1 Construct, Reflection API**  
API mostly used when bootstrapping provided services or when connecting to required ones
- 2 Querying API**  
Synchronous APIs used by consumers to interact with a service
- 3 Messaging API**  
Asynchronous APIs used by consumers to interact with a service
- 4 Event Handling**  
Used to register custom event handler



# Custom Logic

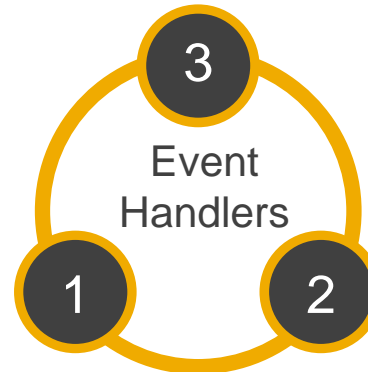
## Custom event handlers



### **.before**

e.g. verify if stock is sufficient

```
// Use reflection to get the csndefinition of Books
const { Books } = cds.entities
// Reduce stock of books upon incoming orders
srv.before('CREATE', 'Orders', async (req) => {
  const tx = cds.transaction(req, order = req.data;
  if (order.Items) {
    const affectedRows = await tx.run(order.Items.map(item =>
      UPDATE(Books).where({ ID: item.book_ID })
        .and('stock >=', item.amount)
        .set('stock -=', item.amount)
    ))
    if (affectedRows.some(row => !row))
      req.error(409, 'Sold out')
  }
})
```



### **.after**

e.g. apply discount if overstocked

```
// Add some discount for overstocked books
srv.after('READ', 'Books', (each) => {
  if (each.stock > 111)
    each.title += ' -- 11% discount!'
})
```



### **.on**

e.g. call external review service

```
const ReviewsService = await cds.connect.to('ReviewsService')
const { Reviews } = ReviewsService.entities
const { Books } = srv.entities

// delegate requests to reviews service
srv.on('READ', 'Reviews', async (req) => {
  const { SELECT } = cds.q((req)
  const results = await SELECT.from(Reviews)
  return results
})
```



Generic handlers serve all standard CRUD requests, and registered custom handlers can add domain logic to application. Multiple handlers can be registered for same event, and single handlers can be registered for multiple events. See [documentation](#) for more.

# Custom Logic

## Request object



### req.method

HTTP method of incoming request, e.g. POST, GET, PUT...



### req.target

Refers to the current request's target entity definition, if any



### req.query

Captures the incoming request as a CQN query object



### req.data

Captures all query parameters as well as http post bodies as a single object



### req.error

Returns an error message to the client. If the first argument is a number, it is used as the HTTP response header code



### req.on

Use this method to register handlers, executed when the whole request is finished

```
srv.before(['CREATE', 'UPDATE'], 'Orders', (req) => {  
  const tx = cds.transaction(req)  
  const order = req.data  
  return Promise.all(order.Items.map(each => tx.run(  
    UPDATE(Books).where({ ID: each.book_ID })  
    .and(`stock >=`, each.amount)  
    .set(`stock -=`, each.amount)  
  ).then(affectedRows => {  
    if (!affectedRows) {  
      req.error(409, `insufficient stock`)  
    }  
  })))  
})
```

```
req.on('succeeded', () => {...}) // request succeeded
```

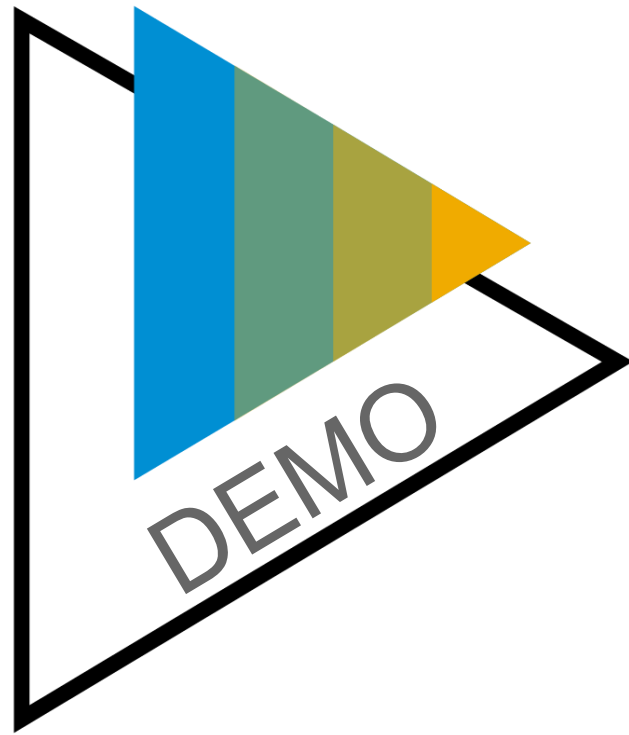
```
req.on('failed', () => {...}) // request failed
```

```
req.on('done', () => {...}) // request succeeded/failed
```



Event handlers registered via ***service.on***, ***service.before***, ***service.after*** all accept a request object as argument, which provides single point of API contact for accessing request input, reading/writing data, and sending responses.

# Custom Logic Demo



# Where to register event handlers

1

### **.js file**

.js file with same name as .cds file

```
// cat-service.cds
service CatalogService {...}
```

```
// cat-service.js
module.exports = (srv) => {...}
```

2

### **.js file and @impl**

.js file and annotation in .cds file

```
@impl: 'my-service.js'
service CatalogService {...}
```

```
// my-service.js
module.exports = (srv) => {...}
```

3

### **cds.serve().with(...)**

.js file or inline function passed to serve.with()

```
cds.serve('./cat-service').with('./cat-service.js')
// or
cds.serve('./cat-service').with(srv => srv.on(...))
```

4

### **cds.serve() ...**

Inline function passed to result of cds.serve()

```
const {CatalogService} = await cds.serve('./cat-service')

CatalogService.on('READ', 'Books', req => {...})
// or
CatalogService.impl(srv => srv.on(..))
```

5

### **cds.connect() ...**

Inline function passed to result of cds.connect()

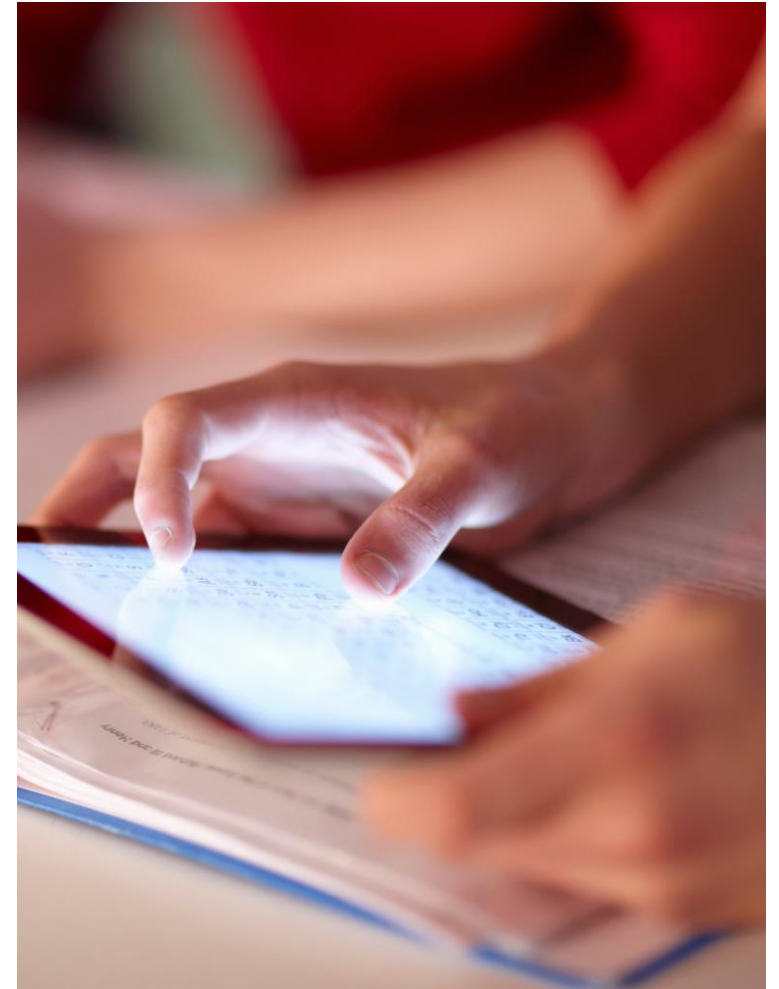
```
const {ExternalService} = await cds.connect('external-service')
ExternalService.on('some-event', evt => {...})
```



See [documentation](#) for more

## What you've learned in this unit

- How to [register custom event handlers](#) to modify your business application behavior
- The [request object](#) is available in all handlers. It provides single point of API contact to access request input, read/write data, and send response.
- There are different ways to wire CDS service models with event handler files or Javascript functions.
- Every active thing in SAP Cloud Application Programming Model is a service, including local or remote services, even databases





# Thank you.

**Contact information:**

**open@sap.com**

Follow all of SAP



[www.sap.com/contactsap](http://www.sap.com/contactsap)

© 2020 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.

The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platforms, directions, and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.

See [www.sap.com/copyright](http://www.sap.com/copyright) for additional trademark information and notices.