Week 2: Development Tasks

Unit 5: Custom Logic

Custom Logic

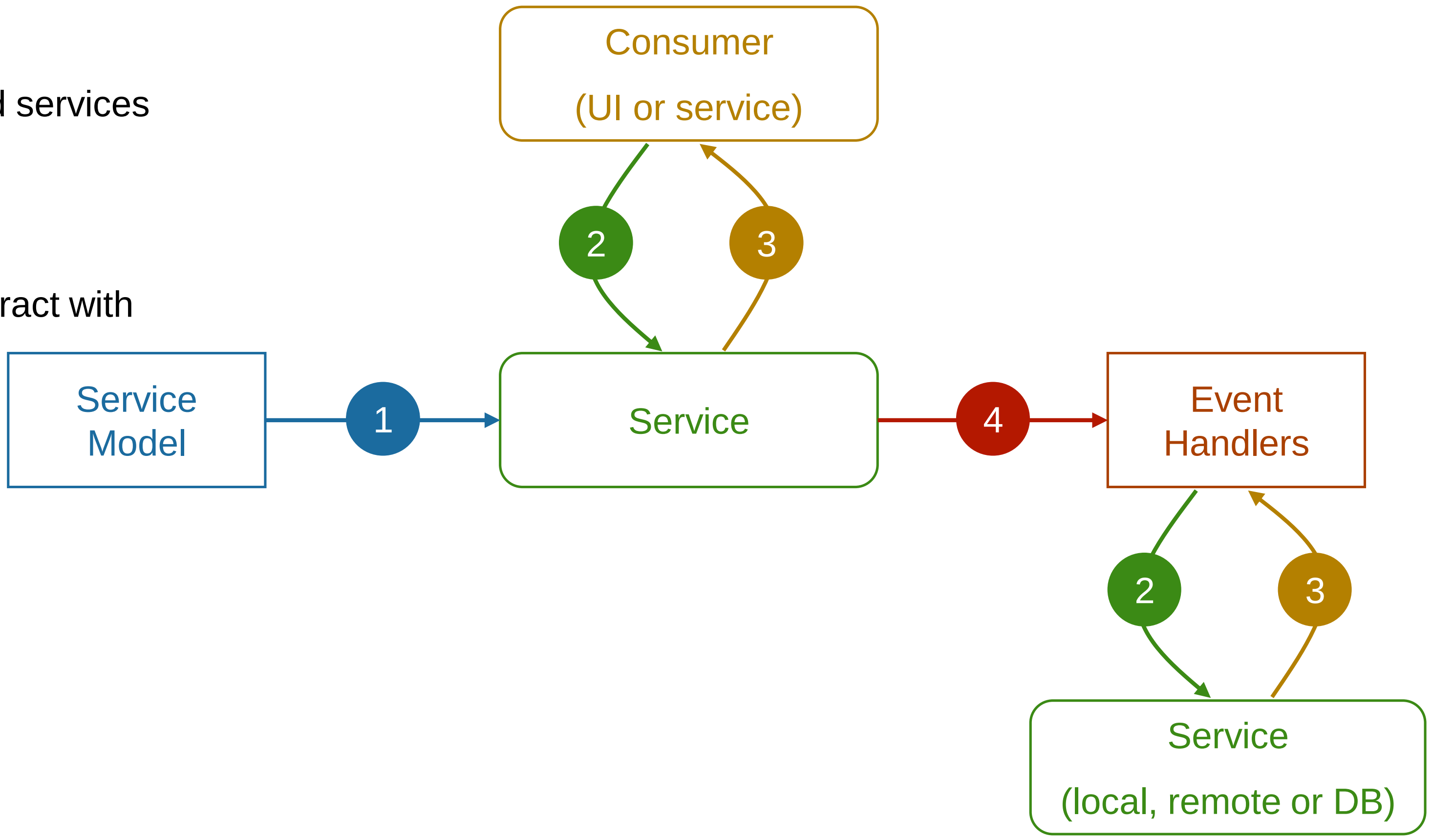
Service API overview

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

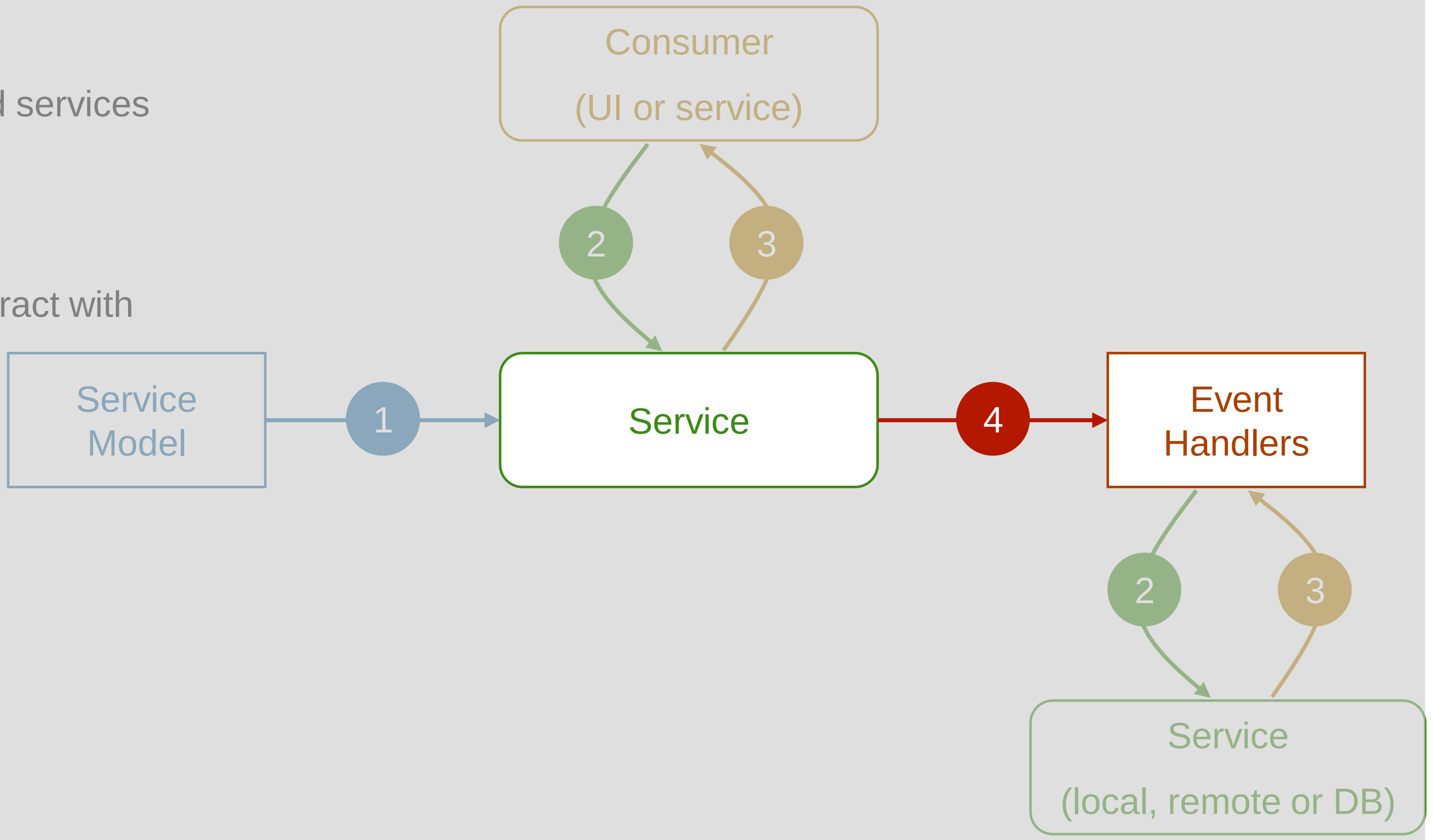
Service API overview

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



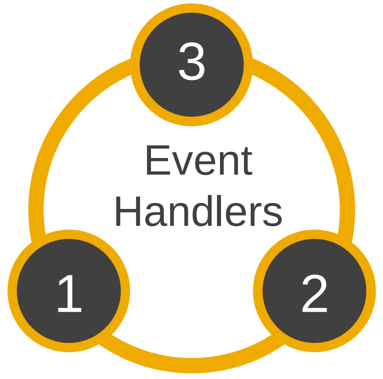
Custom Logic

Custom event handlers

.before

e.g. verify if stock is sufficient

```
// Use reflection to get the csn definition 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.ql(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

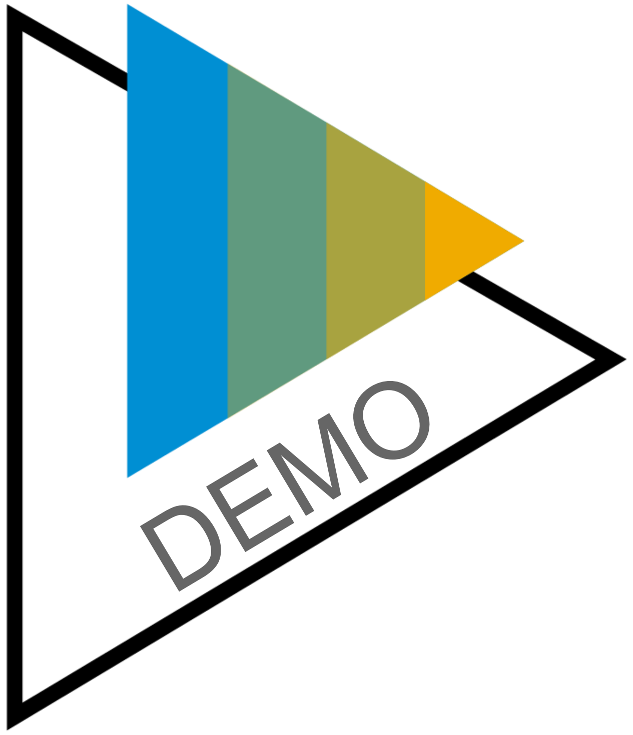
req.on
Use this method to register handlers, executed when the whole request is finished

```
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



Custom Logic

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)=> {...}
```

See documentation for more

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() …*

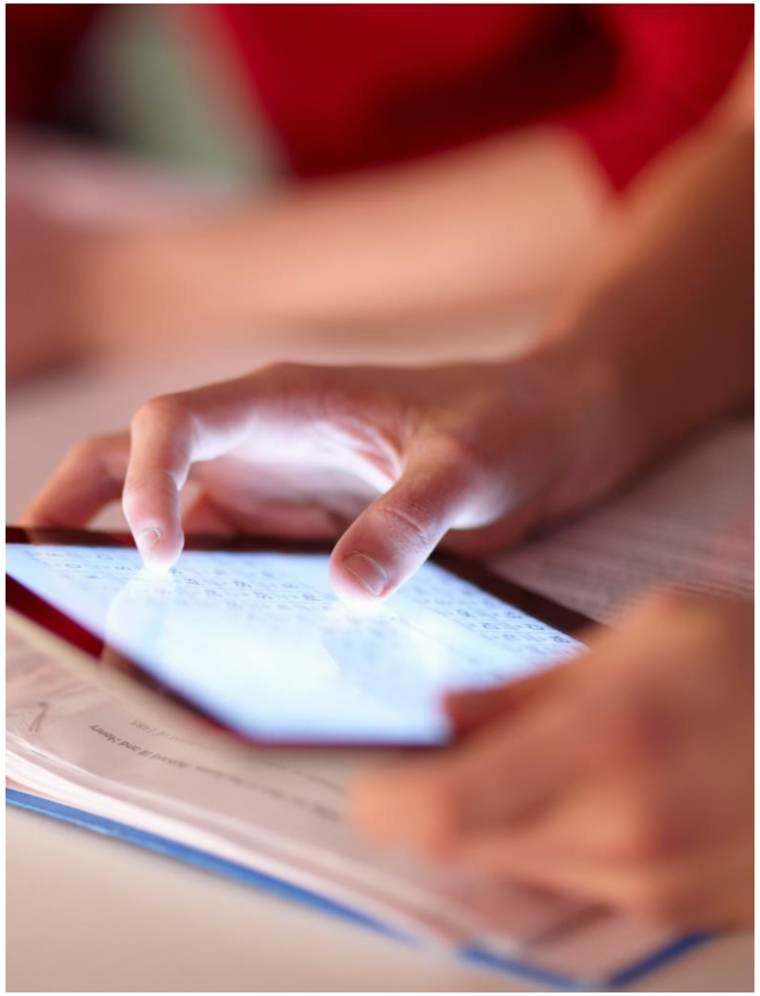
Inline function passed to result of cds.connect()

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

Custom Logic

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

THE BEST RUN SAP