Week 2: Development Tasks
**Unit 2: Model Reuse**

@sap/cds/common

For applications, it is recommended to use the common types and aspects provided through *@sap/cds/common.*

This brings the following benefits:

- Concise and comprehensible models
- Foster interoperability between all applications
- Proven best practices captured from real applications
- Streamlined data models with minimal entry barriers
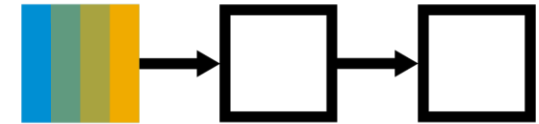- Optimized implementations and runtime performance

and more ...
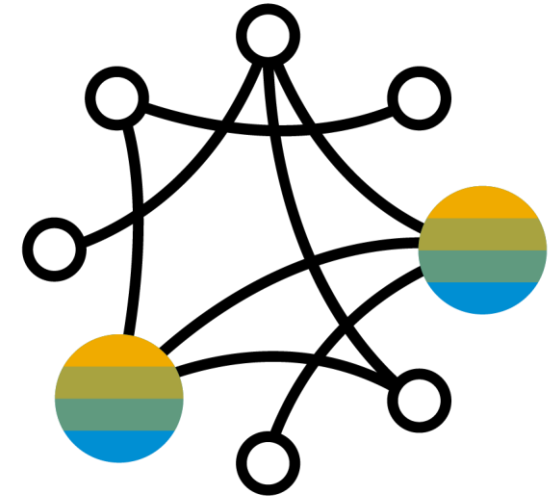
Common aspects

@sap/cds/common defines the following aspects
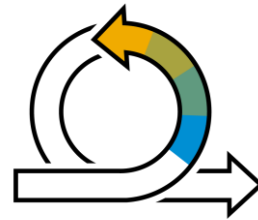for use in your entity definitions:

- cuid
- managed
- temporal

Common aspects

Use **cuid** as a convenient shortcut to add canonical, universally unique primary keys to your entity definitions.

```
using { cuid } from '@sap/cds/common';

entity Books : cuid {
  title : String;
  author : String;
}
```
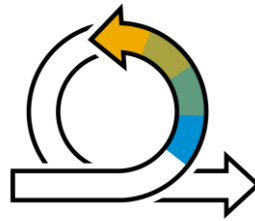
```
entity Books {
  key ID : UUID;
  title : String;
  author : String;
}
```

Model Reuse
Common aspects

Use **managed** to add four elements to capture *created by/created at* and latest *changed by/changed at* management information for records

```
using { managed } from '@sap/cds/comm
on';
entity Books : managed {
  title : String;
  author : String;
}
```



```
entity Books {
  modifiedAt: DateTime;
  createdAt: DateTime;
  createdBy: String;
  modifiedBy: String;
  title : String;
  author : String;
}
```

Common aspects

Use **temporal** to take advantage of the built-in support that handles date-effective records and time slices, including time travel.
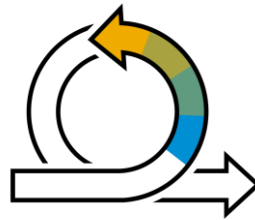
```
using { temporal } from '@sap/cds/common';
entity WorkAssignments : temporal {
  employee : String;
  descr : String;
}
```



```
entity WorkAssignments {
  validFrom : DateTime @cds.valid.from;
  validTo: DateTime @cds.valid.to;
  employee : String;
  descr : String;
}
```
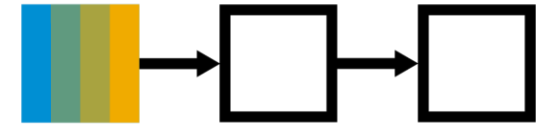
Common types

@sap/cds/common provides predefined easy-to-use types for:

- Countries
- Currencies
- Languages

Common types

The reuse type Country is a simple managed association to the code list for countries as follows:

```
context sap.common {
  aspect CodeList {
      name : localized String(255);
      descr : localized String(1000);
  }

  entity Countries : CodeList {
      key code : String(3);
  }
}

type Country : Association to sap.common.Countries;
```

Common types

The type **Country** acts as a code list table for the country element in your domain model:

```
type Country : Association to sap.common.Countries;
```

```
using { Country } from '@sap/cds/common';
entity Addresses {
  street : String;
  town : String;
  country : Country;
}
```

Model Reuse
using directives

using directives allows to import definitions from other CDS models.

- using foo.bar.scoped.Bar from './contexts';
  entity Car : Bar {}

- using foo.bar.scoped.nested from './contexts';
  entity Moo : nested.Zoo {} *// Zoo is a definitions from foo.bar.scoped.nested*

- using foo.bar.scoped.nested as specified from './contexts';
  entity Zoo : specified.Zoo {} *// Zoo is a definitions from foo.bar.scoped.nested*

- using { Foo as Moo, sub.Bar } from './base-model'; *// Multiple named imports*
  entity Boo : Moo;
  entity Car : Bar;

Model resolution

Working like the "require" function in Node.js and "import" of modules in ES6.

- Names starting with "./" or with "../" are resolved relative to the current model
- If the reference itself is a directory, the reference is loaded from a .../index.<json|cds> file in that directory
- Others are absolute references, fetched for in "node_modules" folders with either .json or .cds suffixes

Demo
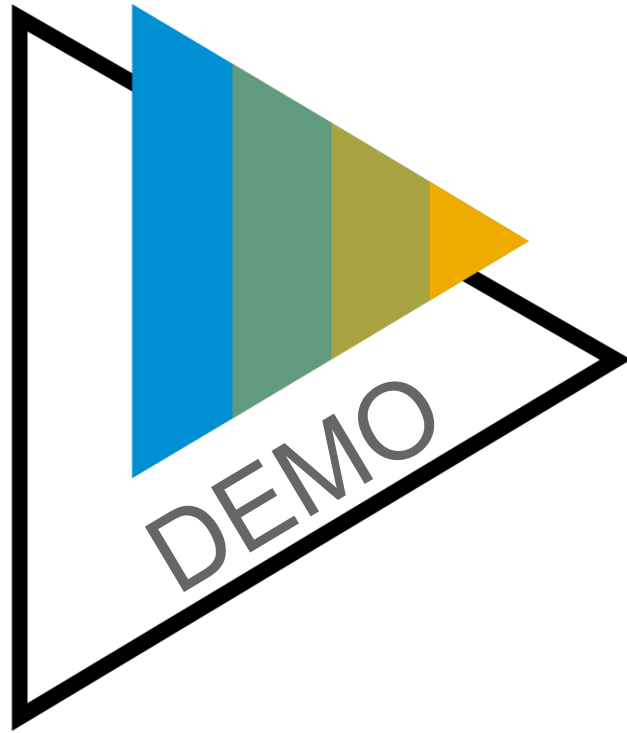
What you've learned in this unit

- @sap/cds/common – predefined common:
  - Aspects
  - Types
- *using* directives
- Model resolution
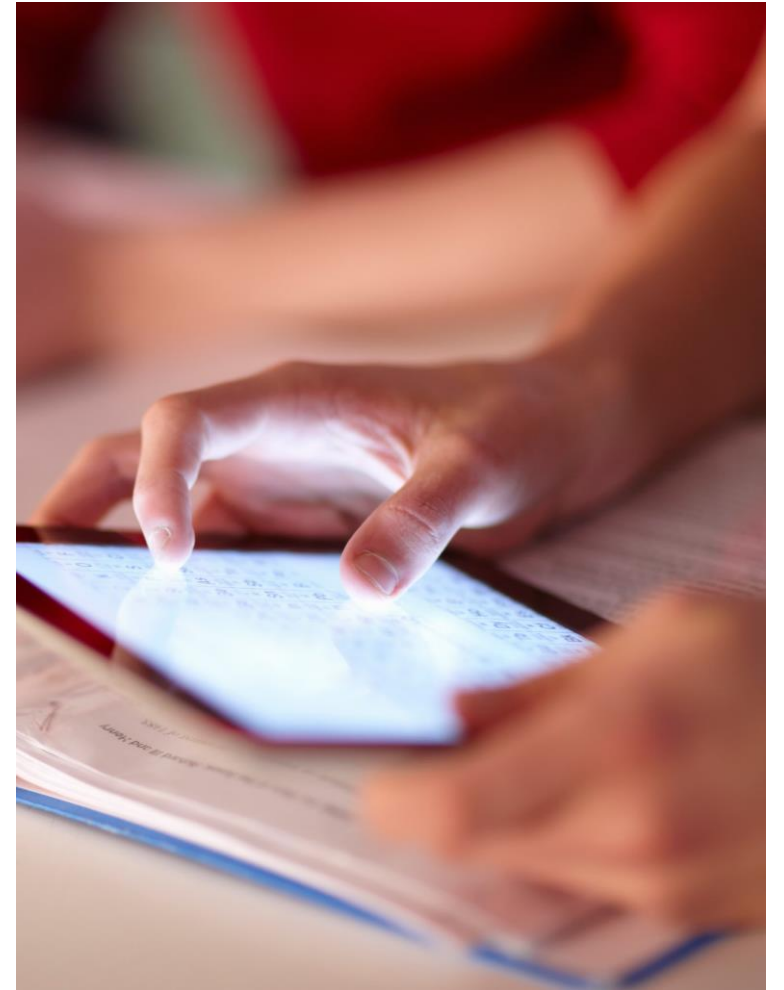
- [Official documentation](#)
- [@sap/cds/common](#)
- [Require function in Node.js](#)
- [ES6 imports](#)
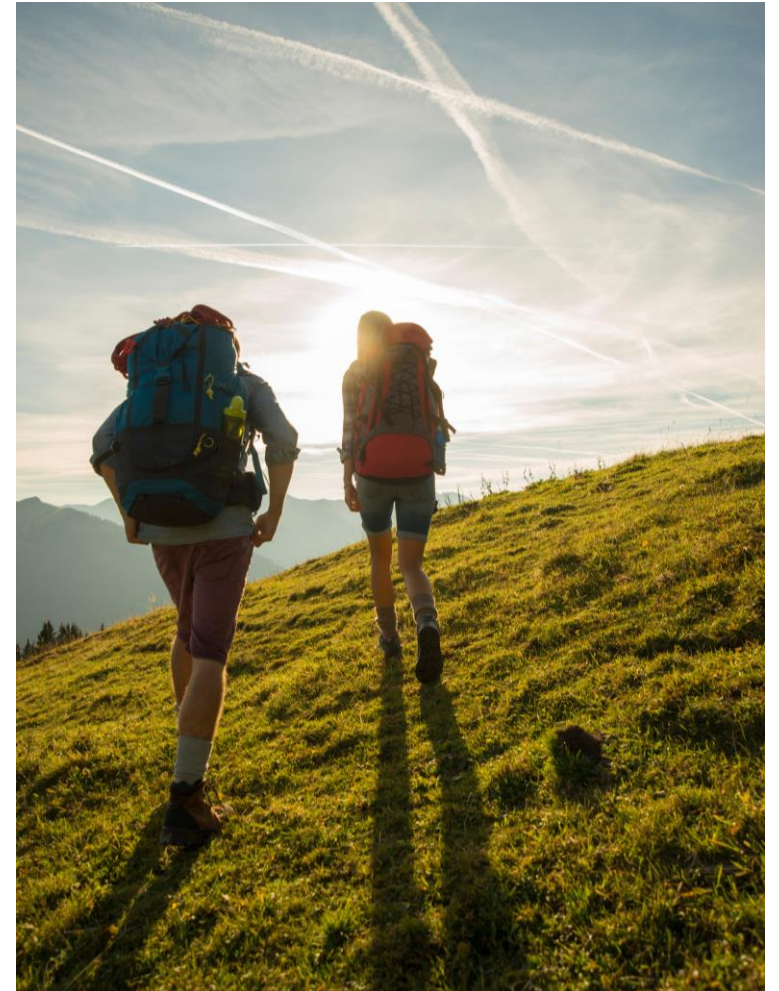
Thank you.

Contact information:

open@sap.com

Follow all of SAP

www.sap.com/contactsap

THE BEST RUN SAP