Week 2: Development Tasks

Unit 1: Domain Modeling

What is Domain Modeling

1. **Keep your domain models clean, concise, comprehensible by**

   Factoring out technical aspects

   Separating Concerns, e.g.

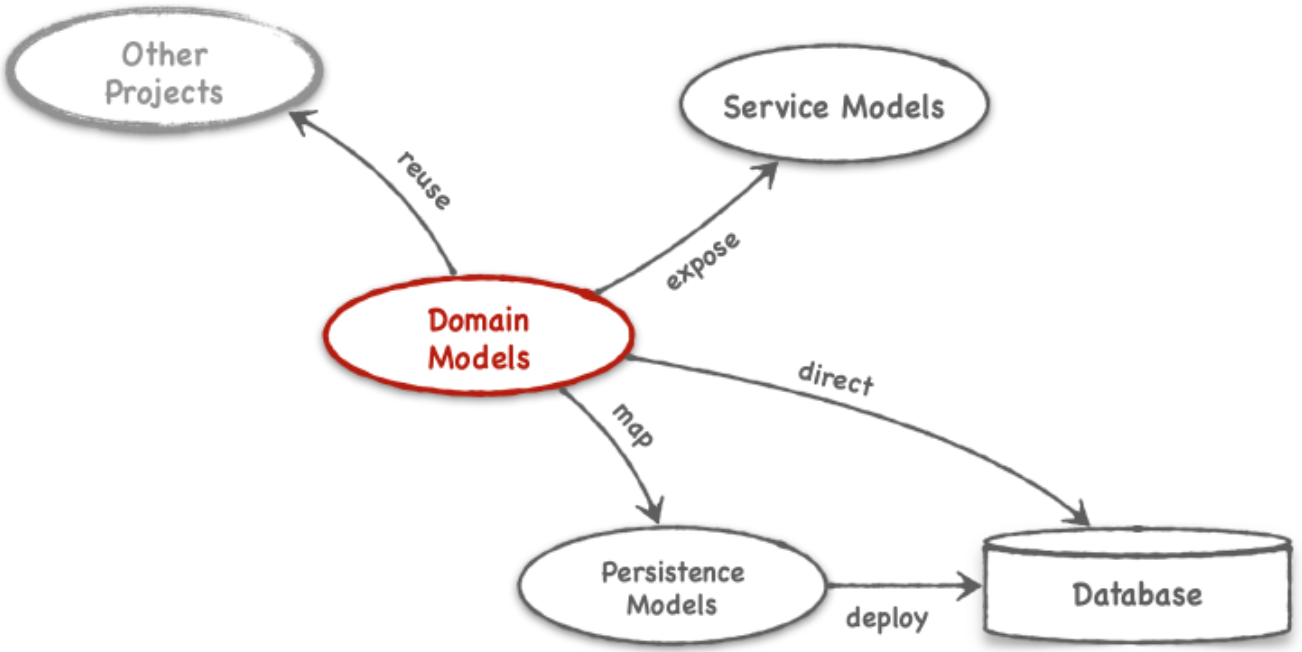
   - Fiori Markup
   - Authorization
   - Persistence

   … in same or different files or projects

2. **Intrinsic Extensibility**

   Project-level
   → Reuse and Extend

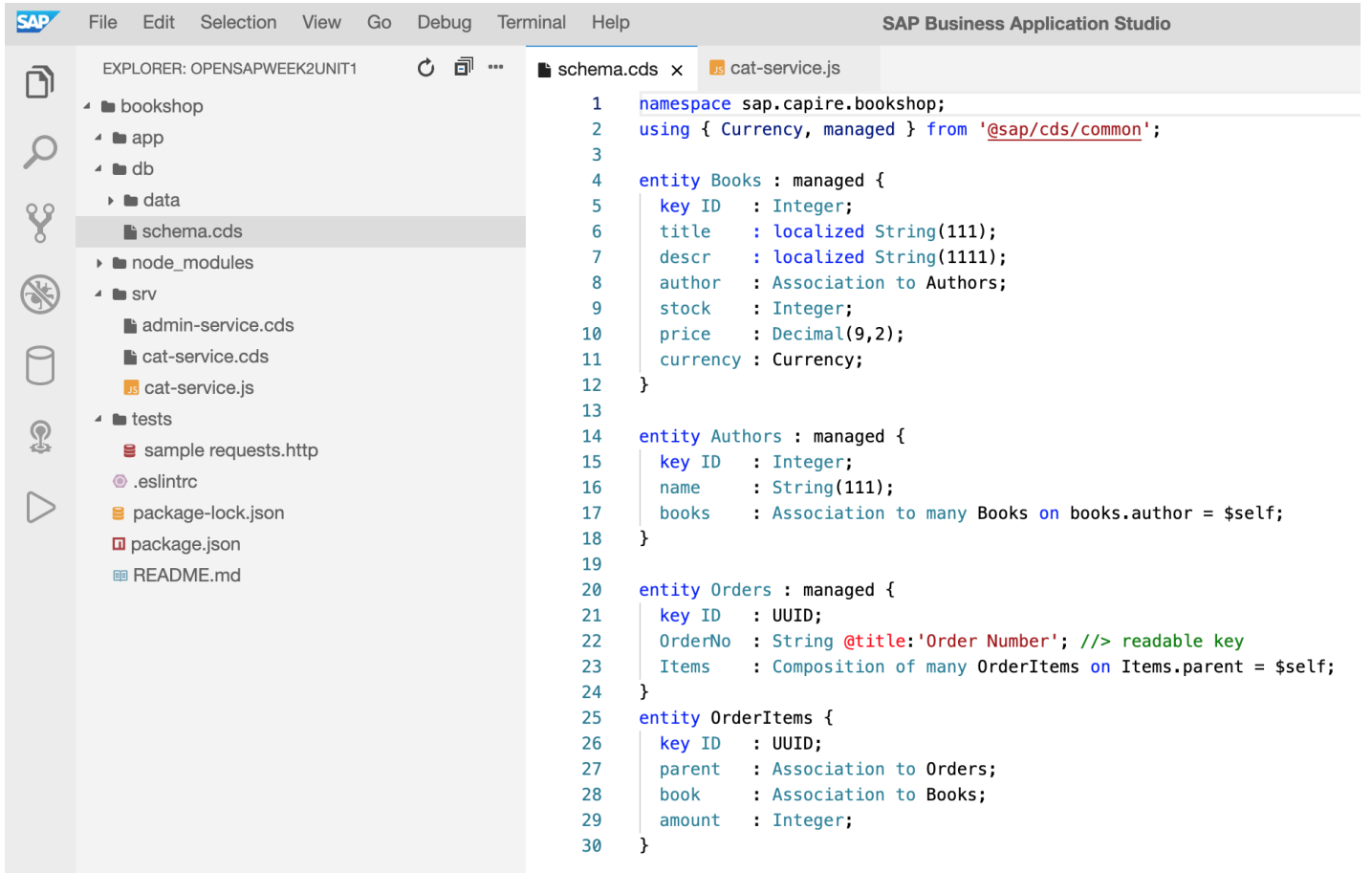
   SaaS-level
   → Dynamic Extensibility



Domain models

- Types, Entities and Associations
- Aspects
- Best Practices in Domain Modeling:
  - Naming Conventions
  - Namespaces
  - Avoid over Normalization
  - Simple, flat structures
  - Enterprise features

    e.g. localized data



Entities, Types and Associations (recap)

```
namespace 'our.library';

entity Books {
  key ID : UUID;
  title  : String;
  descr  : String;
  author : Association to Authors;
}

entity Authors {
  key ID : UUID;
  name   : String;
  books  : Association to many Books on books.author=$self;
  birth  : Date;
  death  : Date;
}
```

1. **Entities** represent data that can be read and written by consumers, uniquely identified by their primary keys.
2. **Types** describe the types of elements within entities.
3. **Associations** capture relationships between entities.

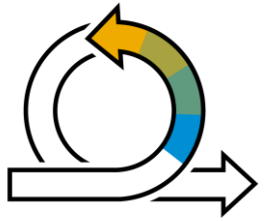
```
entity Books {
  key ID : UUID;
  title : String;
  genre : Genre;
  author : Association to Authors;
}
type Genre : String enum {
  Mystery; Fiction; Drama;
}
```

Aspects

The aspect '*additionalInfo*' can be reused in other entities as well:

```
aspect additionalInfo {
        genre : Genre;
        language : String(1000);
 }

 entity Books : additionalInfo {
   key ID : String(3);
        name: String(3);
}
```



```
entity Books {
   key ID : String(3);
        name : String(3);
        genre : Genre;
        language : String(1000);
}
```

Best practices: Naming Conventions

1. Naming Conventions

- Start ***entity*** and ***type*** names with ***uppercase letters*** – for example, ***Books***
- Start ***elements*** with a ***lowercase letter*** – for example, ***title***
- Use ***plural*** form for ***entities*** – for example, ***Authors***
- Use ***singular*** form for ***types*** – for example, ***Genre***

```
entity Books {
  key ID : UUID;
  title : String;
  genre : Genre;
  author : Association to Authors;
}
type Genre : String enum {
  Mystery; Fiction; Drama;
}
```

Best practices: Using Namespaces & Enterprise features

1. Using Namespaces

- Use namespaces if reusing models
- Services are rarely reused, so rarely need namespaces

```
namespace foo.bar;
entity Boo {}
entity Moo : Boo {}
```

..... is equivalent to:

```
entity foo.bar.Boo {}
entity foo.bar.Moo : foo.bar.Boo {}
```

2. Namespaces vs Contexts

- Use namespaces over top-level contexts

Do:	Don't:
namespace foo.bar; *entity Boo {}* *entity Car {}*	*context foo.bar {* *entity Boo {}* *entity Car {}* *}*

3. Using Enterprise Features

- Localized data
- Authentication & Authorization

```
...
  title  : localized String;
  descr  : localized String;
...
```

Best practices: Avoid over-normalization

Do

```
entity Contacts {
  key ID : UUID;
  name : String;
  emails : array of {
    kind : String;
    address : String;
    primary : Boolean;
  }
  phones : array of {...}
}
```

Don’t

```
entity Contacts {
  key ID : UUID;
  name : String;
  emails : Composition of many EmailAddresses on emails.contact=$self;
  phones : Composition of many PhoneNumbers on phones.contact=$self;
}
entity EmailAddresses {
  contact : Association to Contacts;
  key ID  : UUID;
  kind    : String;
  address : String;
  primary : Boolean;
}
entity PhoneNumbers {...}
```

Best practices: Favor simple, flat structures

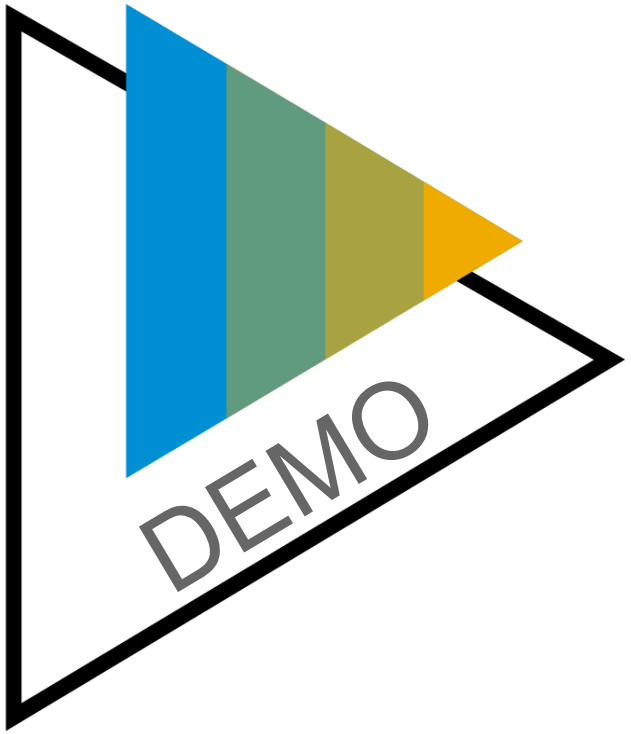
Do

```
entity Contacts {
  isCompany : Boolean;
  company : String;
  title : String;
  firstname : String;
  lastname : String;
  ...
}
```

Don’t

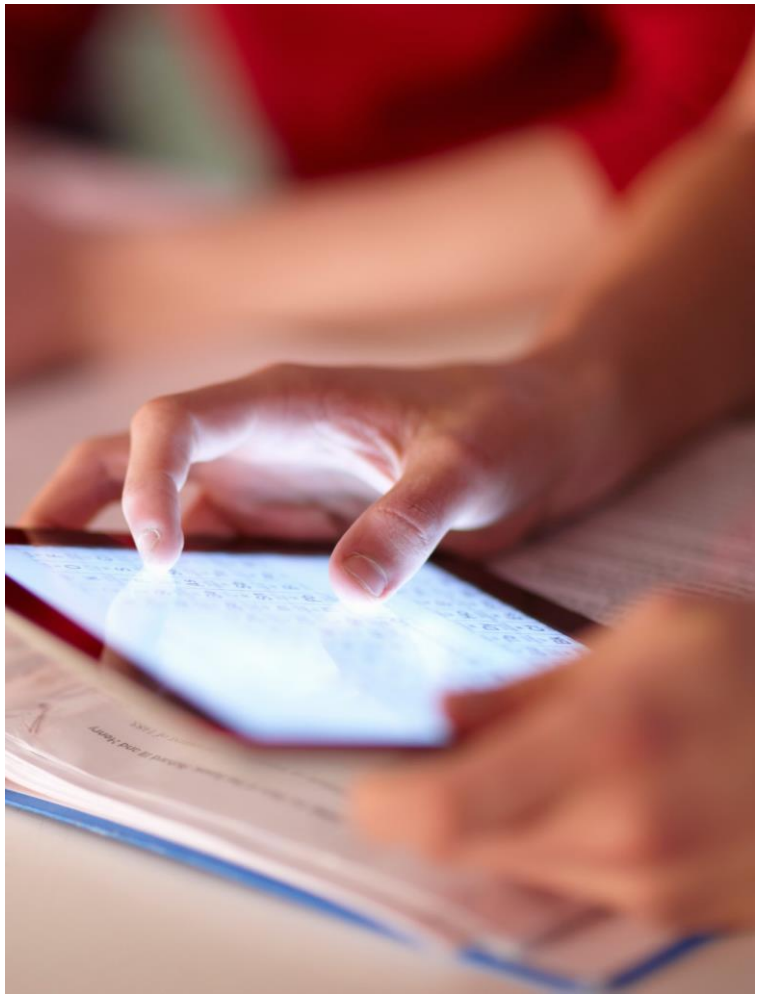
```
entity Contacts {
  isCompany : Boolean;
  companyData : {
    name : String;
  }
  personData : {
    title : {
      primary : String;
      secondary : String;
    }
    name : {
      firstname : String;
      lastname : String;
      ...
    }
  }
  ...
}
```

Domain models



What you've learned in this unit

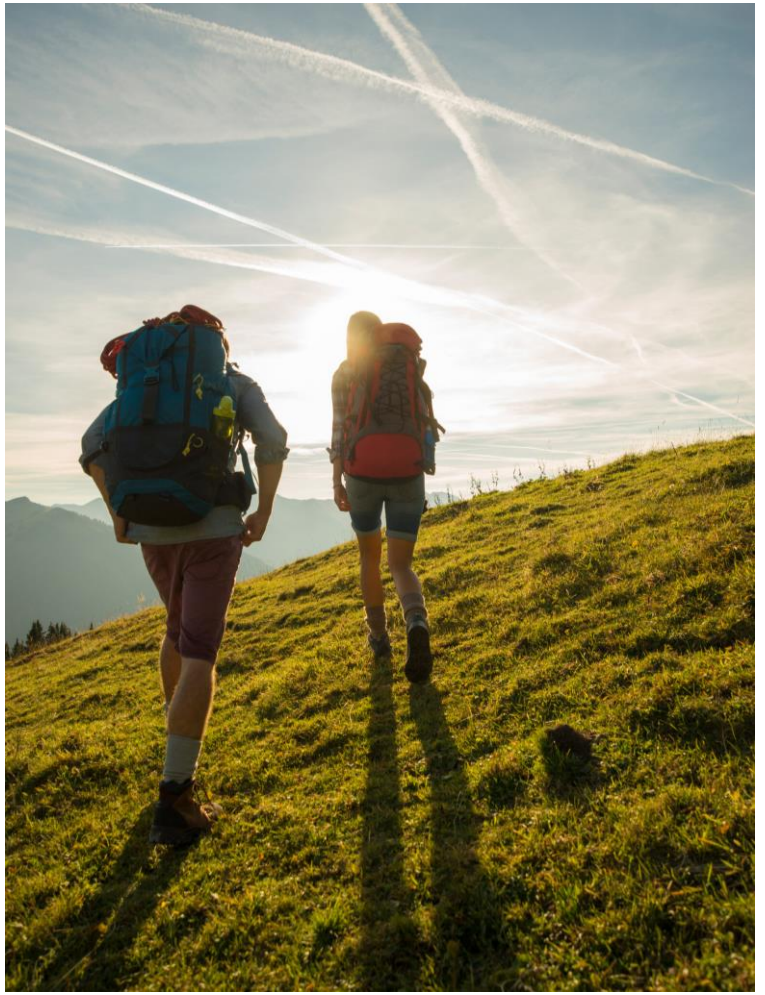
- Domain modeling with Core Data Services
- Types, Entities and Associations
- Aspects
- Best Practices in Domain Modeling:
  - ✓ Naming Conventions
  - ✓ Namespaces
  - ✓ Avoid over Normalization
  - ✓ Simple, flat structures
  - ✓ Enterprise features e.g. localized data



Further reading

Additional Material

- Official documentation
- Blogs: SAP Cloud Application Programming Model
- Starter Scenario
- TechEd Replay
- Hello World Tutorial



Thank you.

Contact information:

open@sap.com

Follow all of SAP

www.sap.com/contactsap

THE BEST RUN SAP®