Defining and Packaging ABL services for PASOE

Beyond The Code Series

Peter Judge
pjudge@progress.com

Application Architectures

Service Interfaces

Deployment Levels

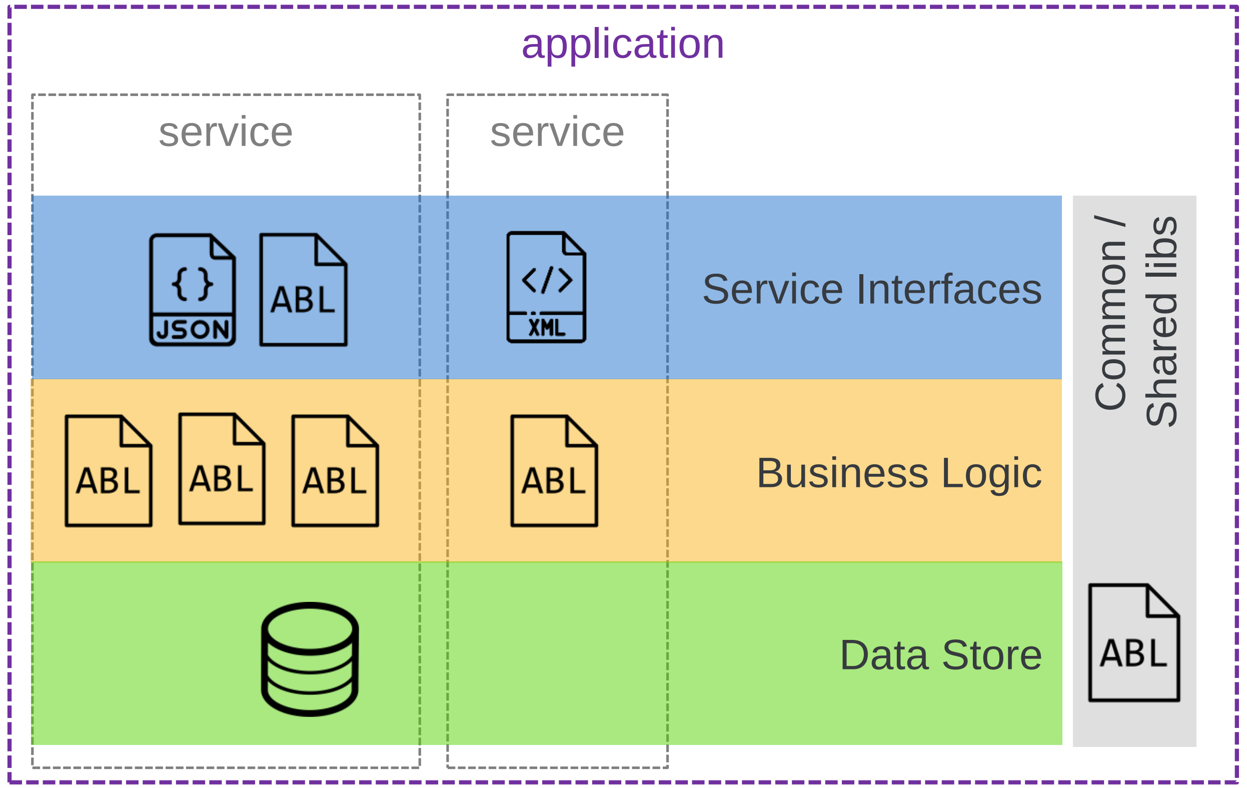
Deployment packages

This is NOT how-to

- Write applications
- Secure applications
- Monitor/administer instances

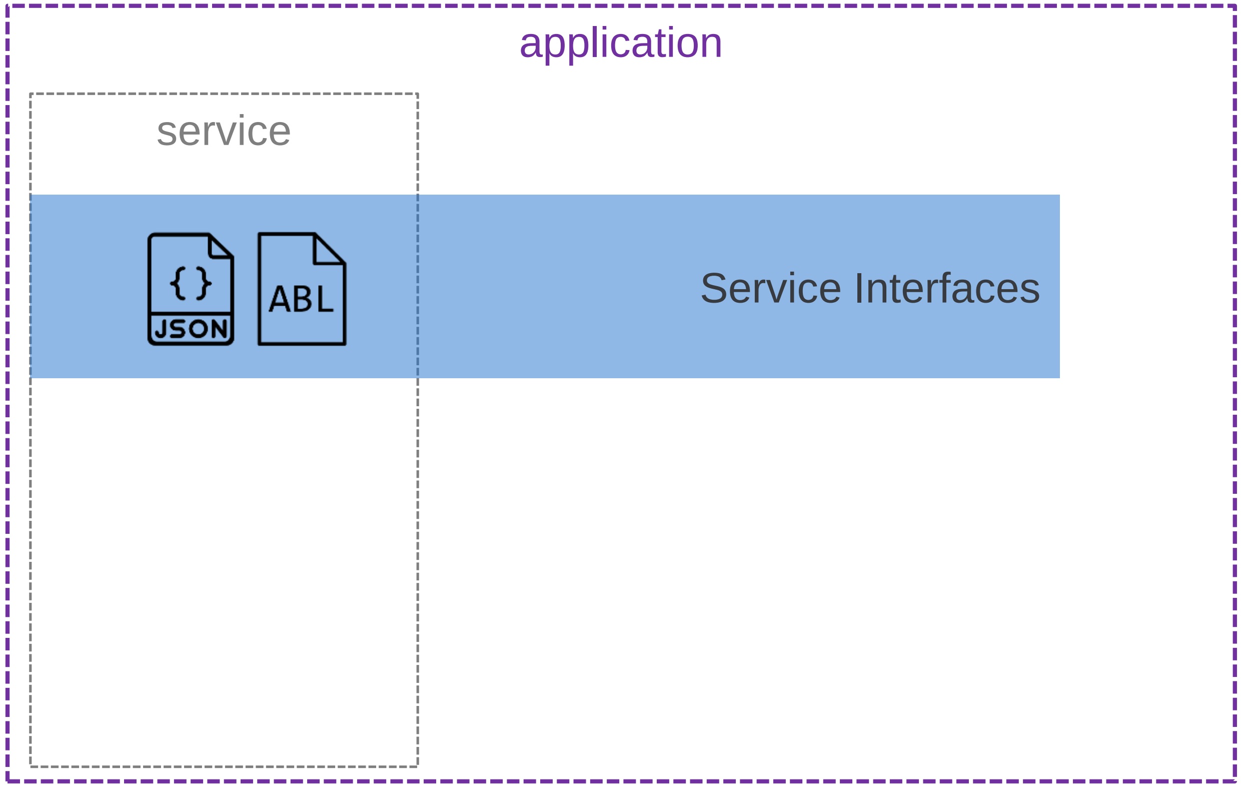
Application Architecture

Business Application Architecture



- An application consists of at least this set of logical components
  - Service Interfaces
  - Business Logic
  - Data Store(s)
  - Common libraries, incl $DLC
- Deployable artifacts are
  - ABL code, loose or in PLs
  - Service descriptors
  - Databases and/or schema (.DF) & data (.D)
  - Security configuration
  - Other configuration files
  - Scripts to tailor, set env vars

Business Application Architecture



A service consists of a client-specific API and a Service Interface

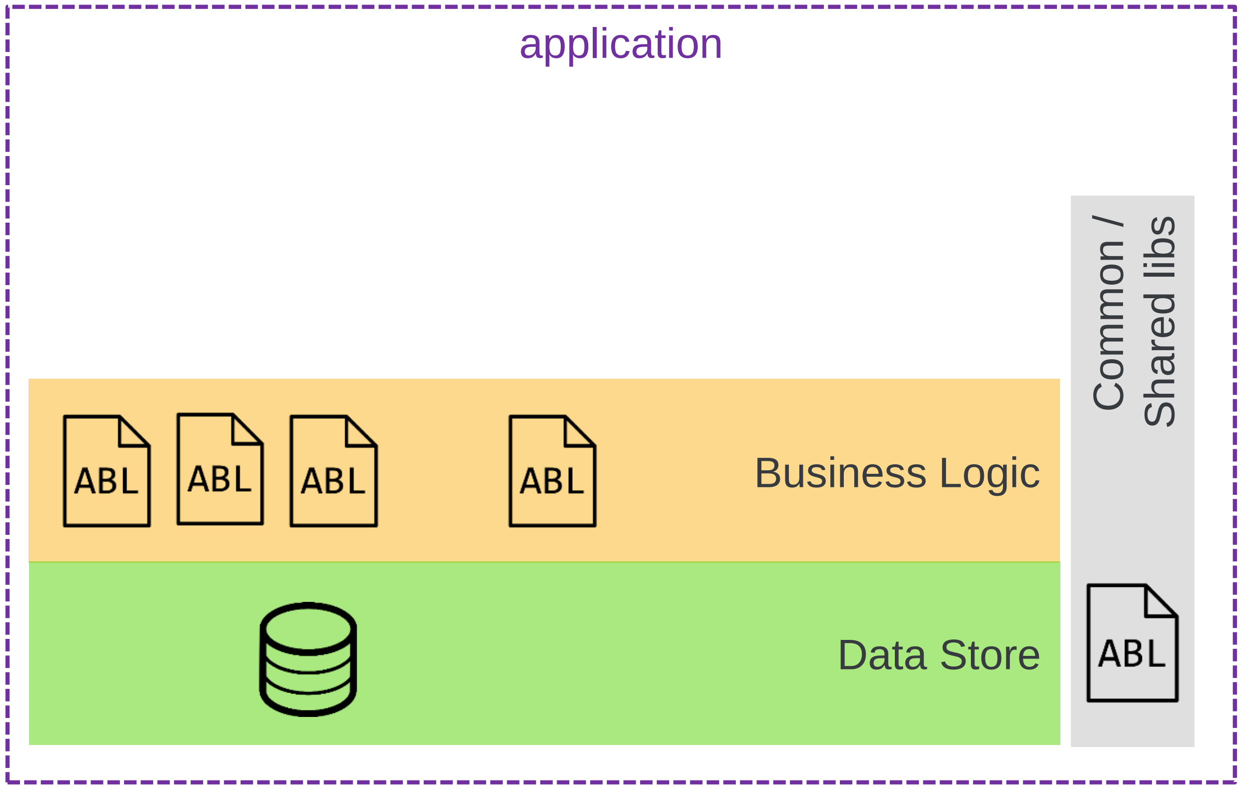
- WebApp provides the API
- ABL Service provides the Service Interfaces

The API is a set of endpoints (URIs), methods, schemas, protocols that specific clients know how to talk.

Service interfaces provide the translation layer between a request and the underlying business services

- Provide authentication and authorization and error handling
- Translate input / output formats to and from business logic

Business Application Architecture



Business domain logic like tax calculations, master data maintenance, order entry, MRP …

Systems-of-record data in one or more OE databases

Common or shared libraries typically contain generic code that is needed for an application but provides no direct business value

Service Interface Approaches

Data Object (REST)

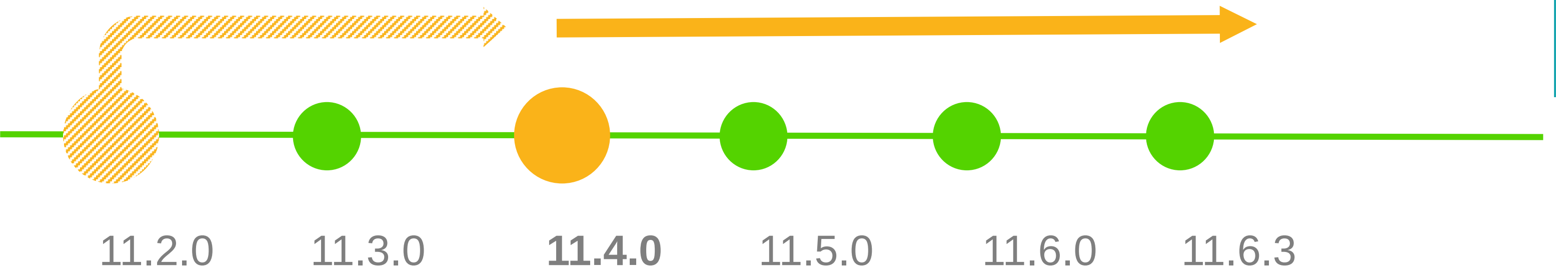
Data Object (WebHandler)

REST (Mapped RPC)

WebHandler

DataObjectHandler

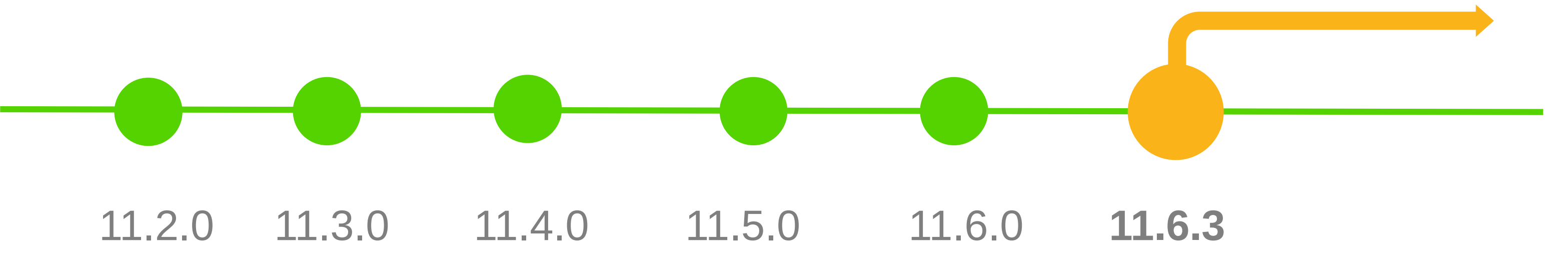
- Formerly Mobile Services
- Annotate certain methods (w/ particular signatures)
- Very prescriptive
  - Programming model
  - URI paths
  - Content types (JSON)
- Uses REST transport
- Creates Data Service Catalog as public API
- Service descriptors in java



Service Interface Approaches

- Data Object (REST)
- **Data Object (WebHandler)**
- REST (Mapped RPC)
- WebHandler
- DataObjectHandler

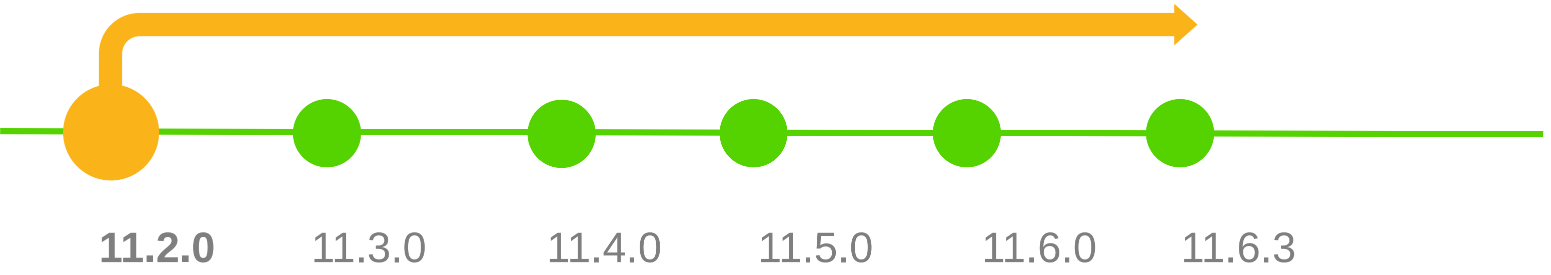
- WEB-transport variant
- Annotate certain methods (w/ particular signatures)
- Still very prescriptive
- More flexibility in mapping
- Creates Data Service Catalog as public API
- Service descriptors in ABL



Service Interface Approaches

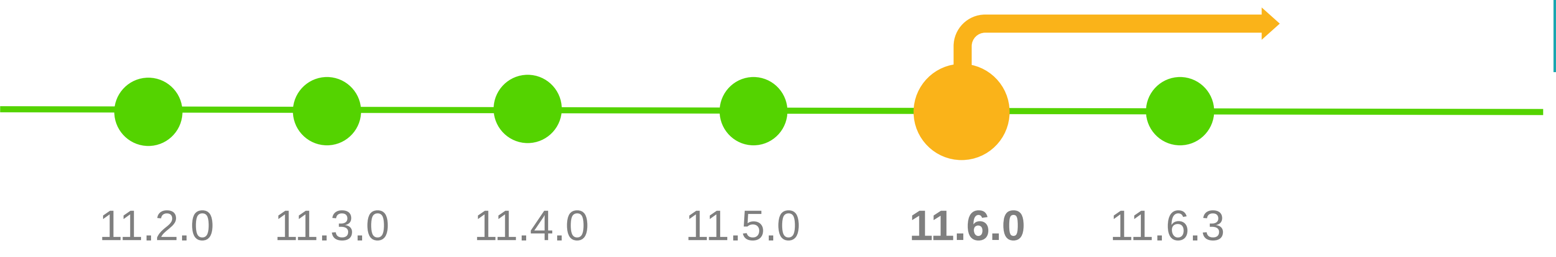
- Data Object (REST)
- Data Object (WebHandler)
- **REST (Mapped RPC)**
- WebHandler
- DataObjectHandler

- Formerly REST Services
- Graphical mapping tool
- Uses REST transport
- Flexible in URI paths
- Limited content types (JSON)
- Service descriptors in java



Service Interface Approaches

- Data Object (REST)
- Data Object (WebHandler)
- REST (Mapped RPC)
- **WebHandler**
- DataObjectHandler



- Associate an OOABL WebHandler class with a URI pattern
- Uses WEB transport
- VERY flexible, URI is all yours
- Do whatever you want in code/ABL

- Service descriptors in ABL
- In-the-box versions
  - **`OpenEdge.Web.WebHandler`**
  - `OpenEdge.Web.CompatibilityHandler`
  - `OpenEdge.Web.DefaultHandler`
  - `OpenEdge.Web.PingWebHandler`

Service Interface Approaches

Data Object (REST)

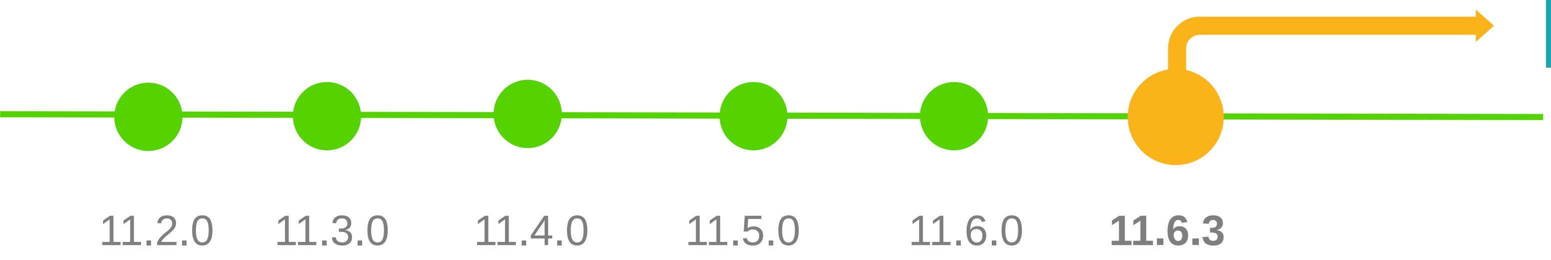
Data Object (WebHandler)

REST (Mapped RPC)

WebHandler

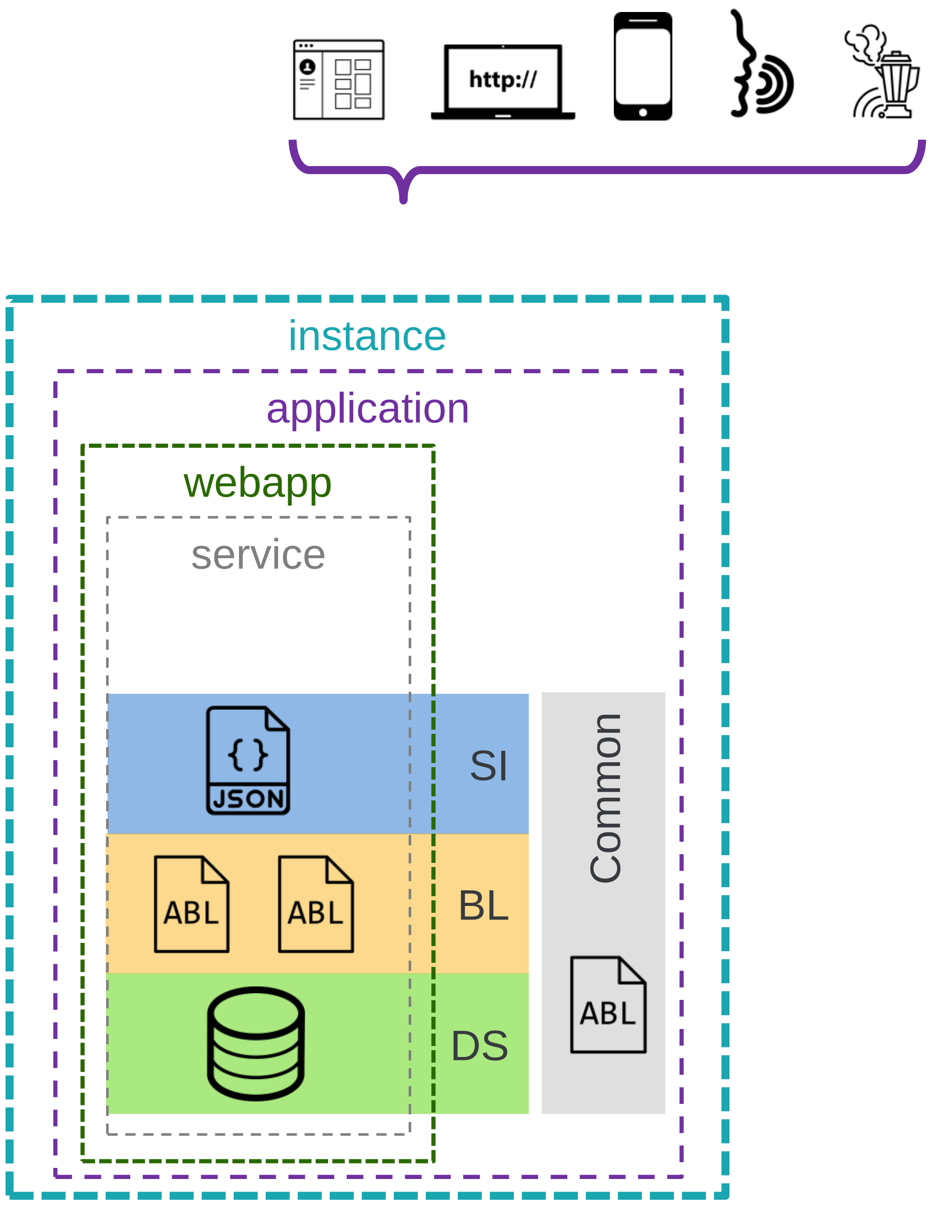
DataObjectHandler

- Pre-built generic WebHandler
- Effectively a WebHandler-based version of Mapped RPC
- Mapping defined in JSON file
- Very flexible
  - Programming model
  - URI paths
  - Content types
- Service descriptors in ABL



Instance application configuration

Basic 1+1+1

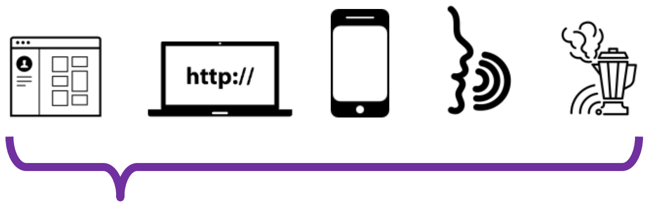


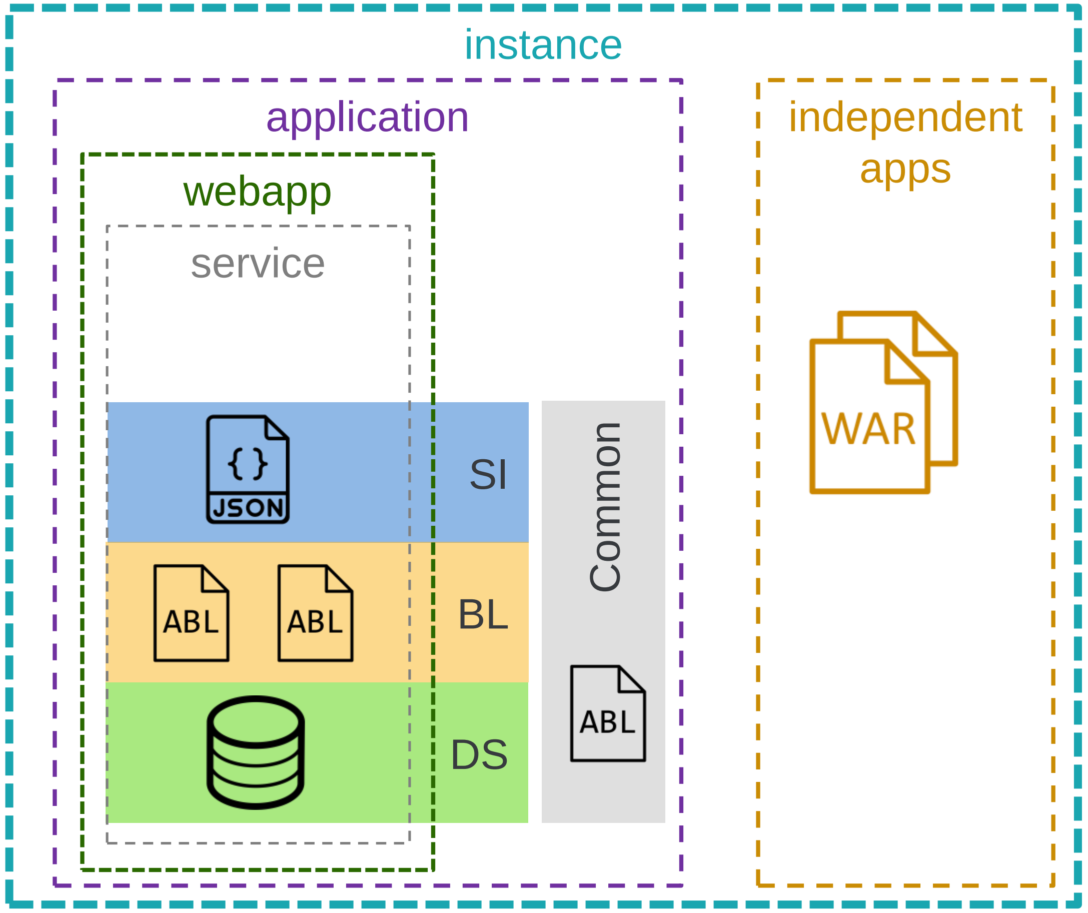
The simplest runnable PASOE configuration

- 1 instance
- 1 application
- 1 webapp and 1+ services

- What you get when you create an instance using `tcman create`

Extras (all configs)





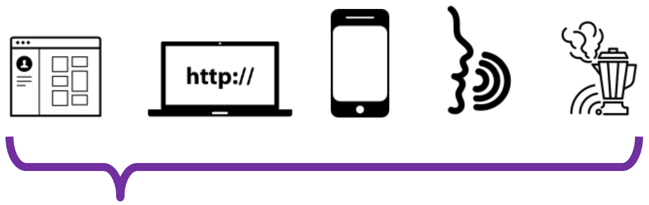
Extend functionality using independent* apps

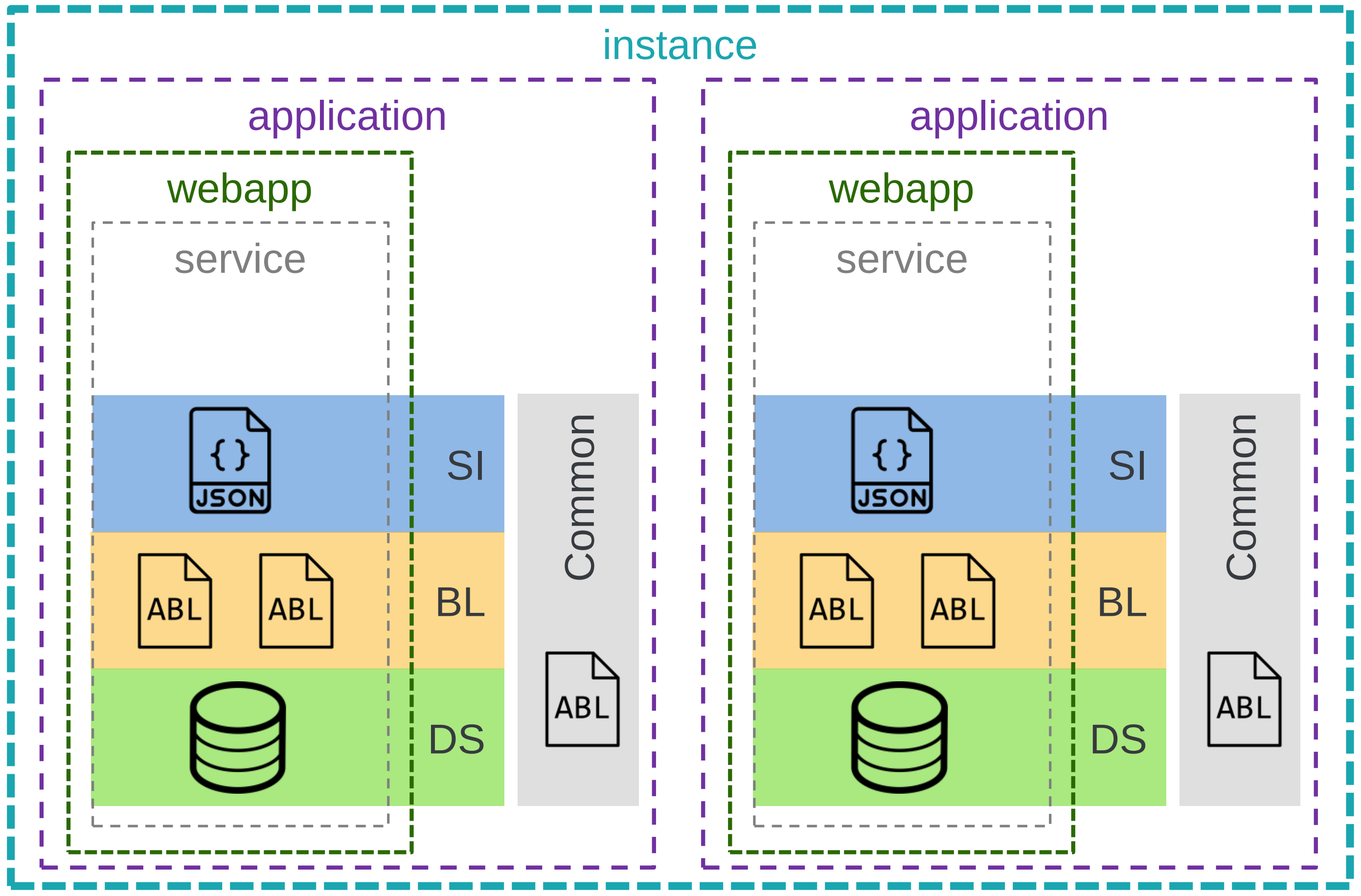
- `oemanager`
- `oehealth`
- `oedbg`
- (Tomcat) `manager`
- Corticon
- Web UI

* don't run ABL business logic

Some are shipped in `$DLC/servers/pasoe/extras`

n Apps

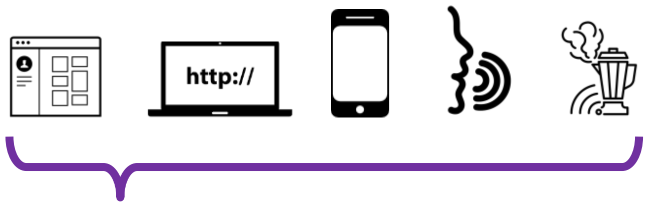


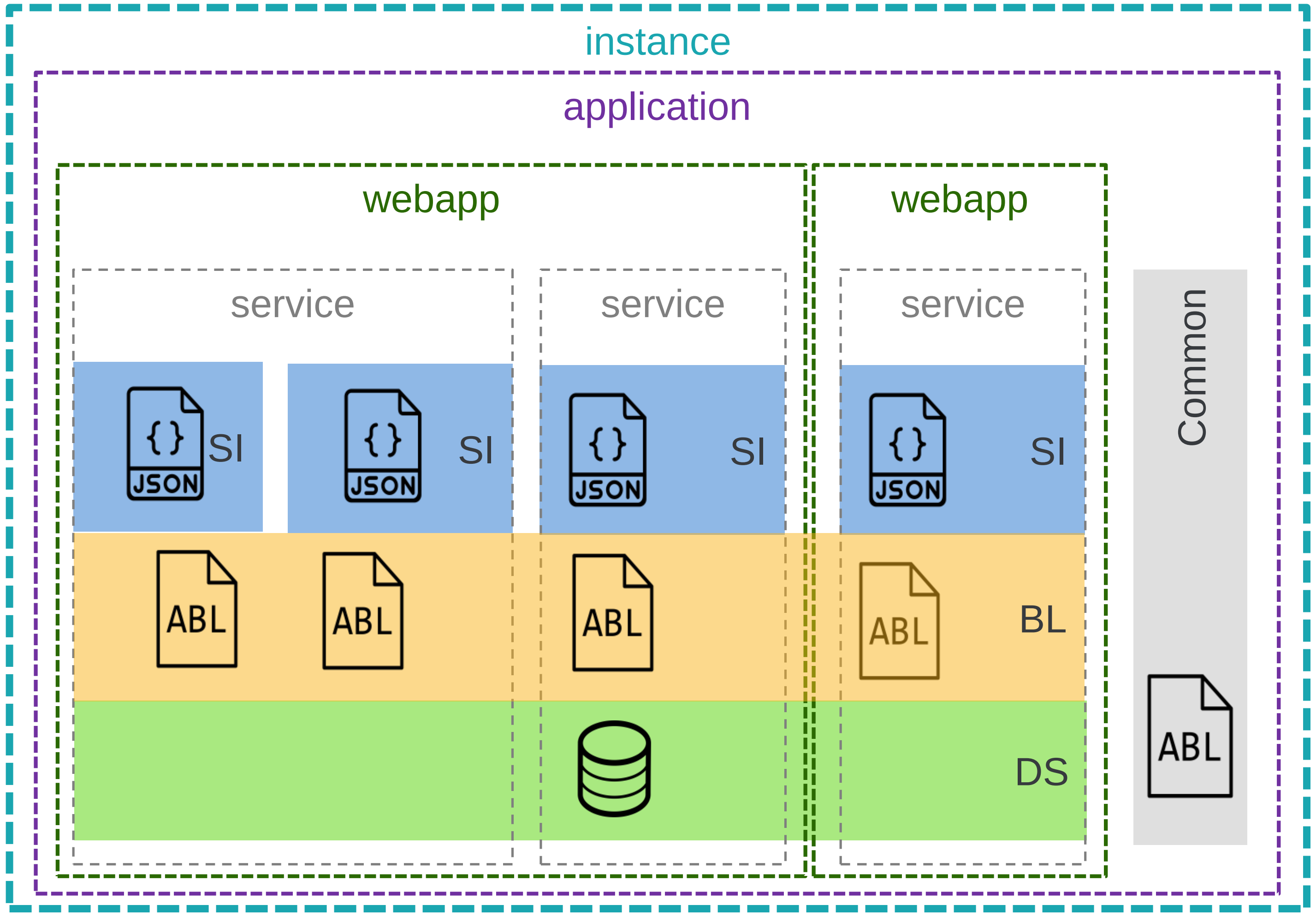


An instance can support many applications, where *n* is bound by CPU & memory

- 1 app per instance
  - Max scalability, flexibility, simplicity
- Many apps per instance
  - "Family" of apps vendor
  - Developer environment
  - Small deployments (limited resources: physical & people)

m Webapps





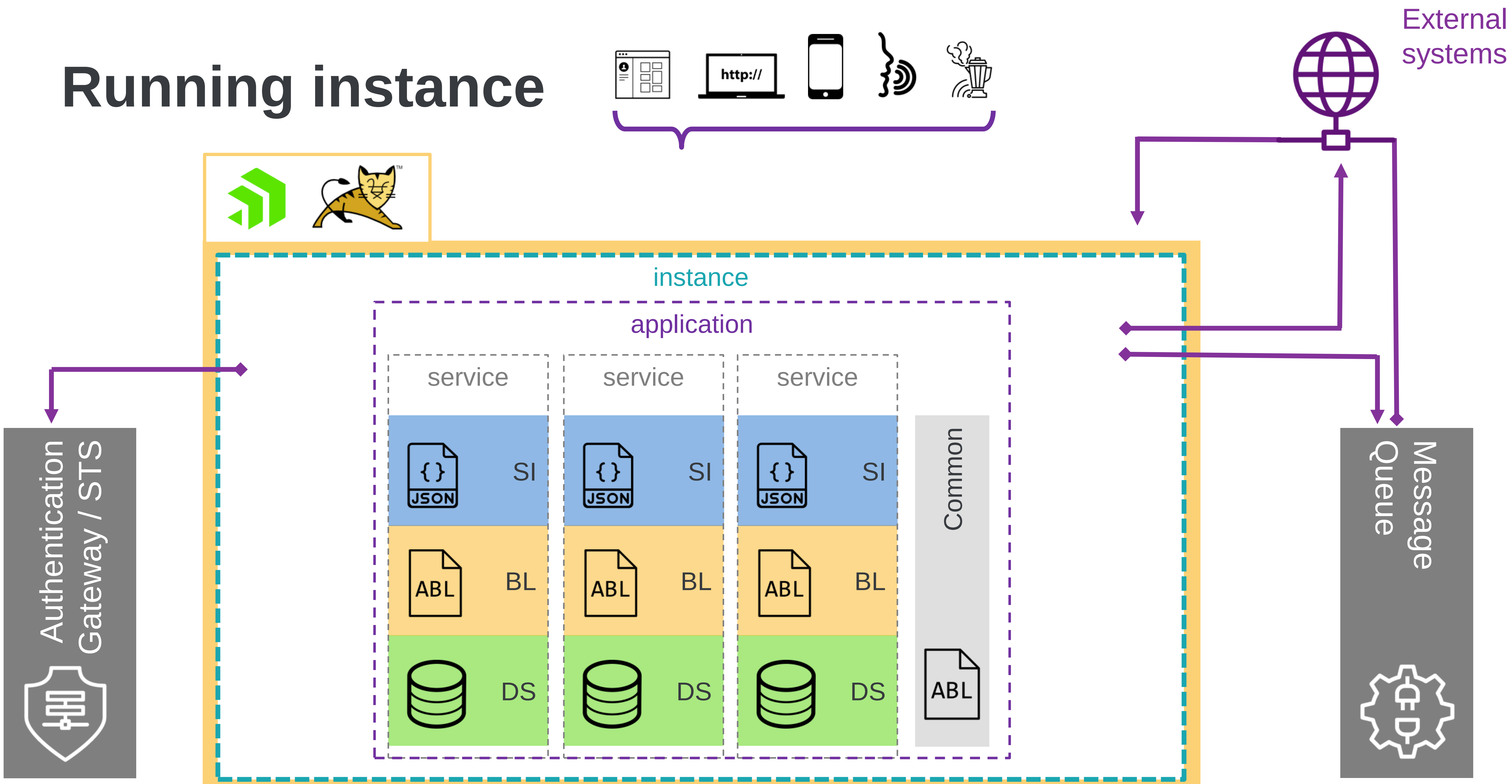
Services form part of URL space

- Service boundaries are authentication boundaries, enforced by webapp
- Webapps contain many ABL services

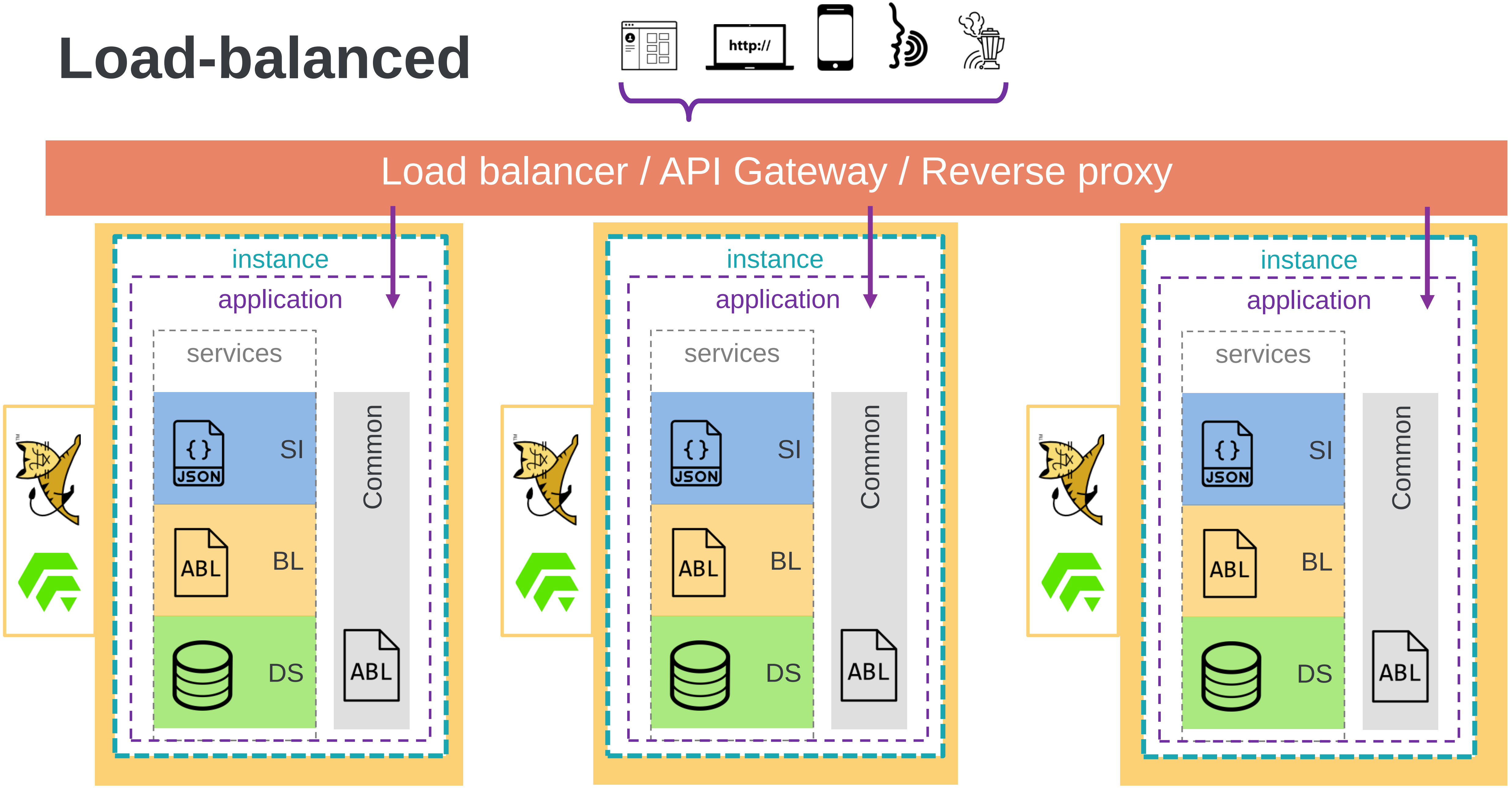
Apps may also have many services for separation-of-concerns

- Service definitions are responsibility of developers and devops / admins

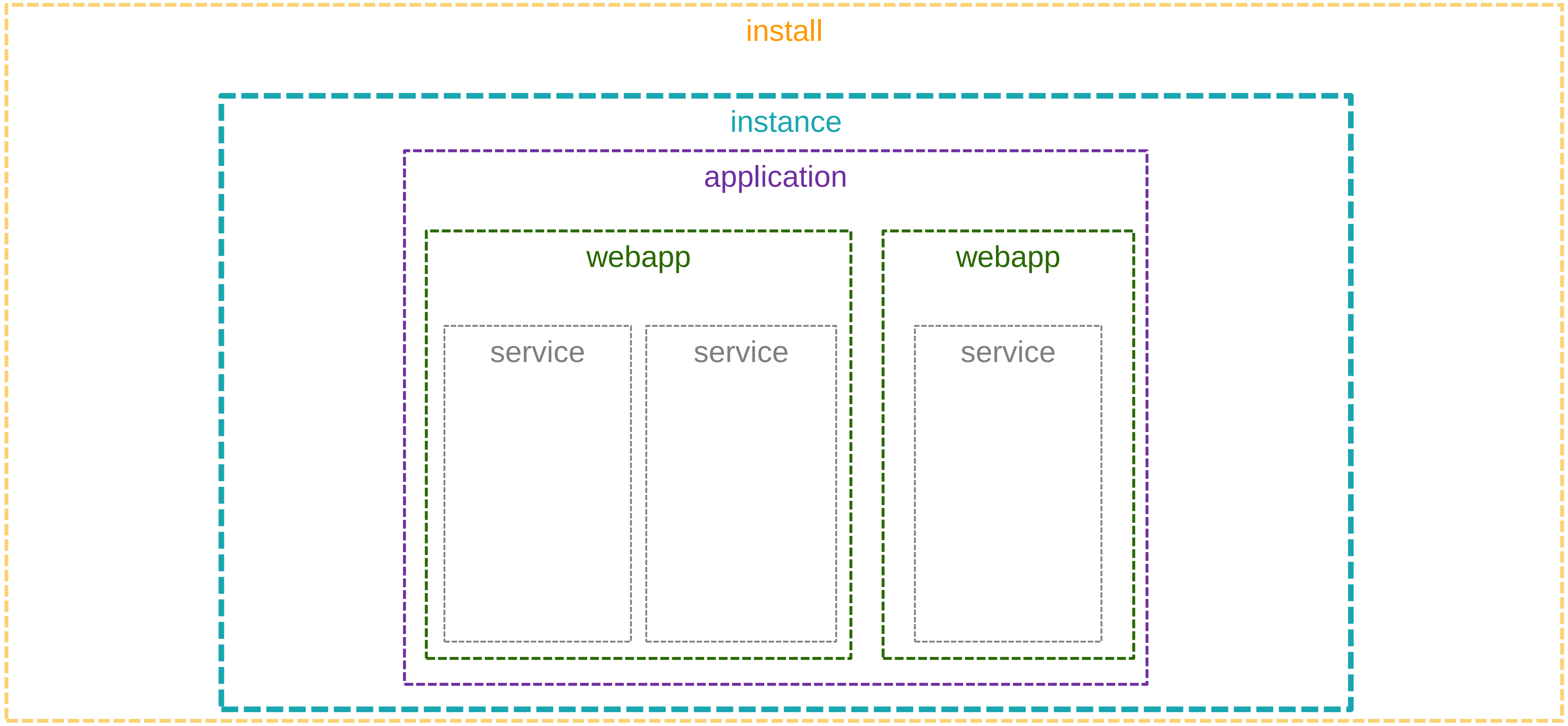
Running instance



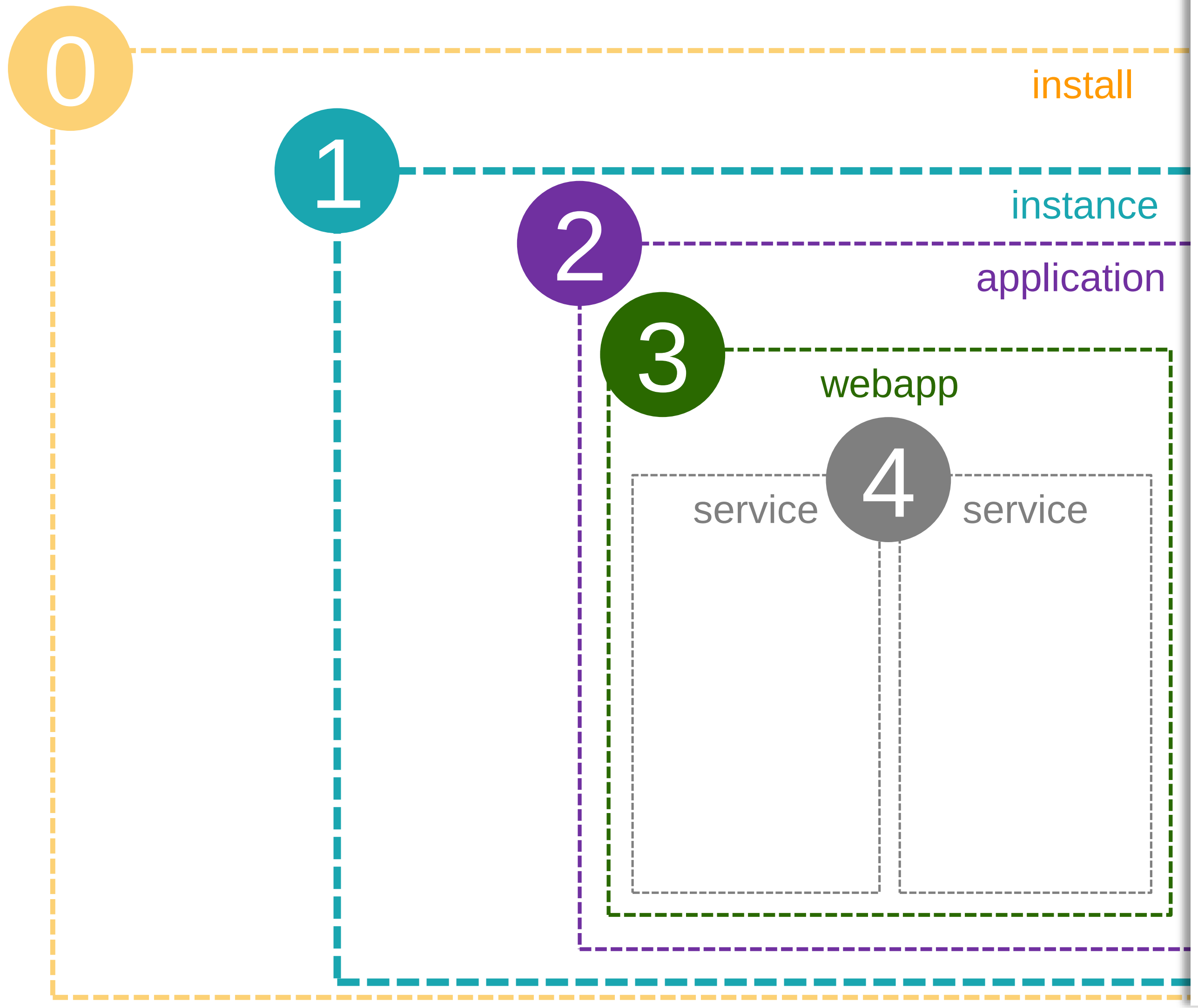
Load-balanced



Deployment levels



Deployment levels



0 Install

Affects all of the instances on a machine/container.

`$DLC`, you all know and love it; runs ABL and the DB.

`$CATALINA_HOME` contains the Tomcat exe's

1 Instance

An instance runs one or more business applications. It may also run other Tomcat webapps and controls load balancing & failover.

2 ABL Application

A business application, as defined by a PROPATH, database connection(s), agent configuration(s) and their executable AVM sessions. It contains one or more ABL webapps

3 ABL Webapp

A secured set of services that provide access into the ABL application. A webapp provides a name and (primarily) authentication services, and contains one or more ABL services

4 ABL Service

A service interface (often ABL code) into the ABL Application's business logic; also provides authorization in the webapp and ABL

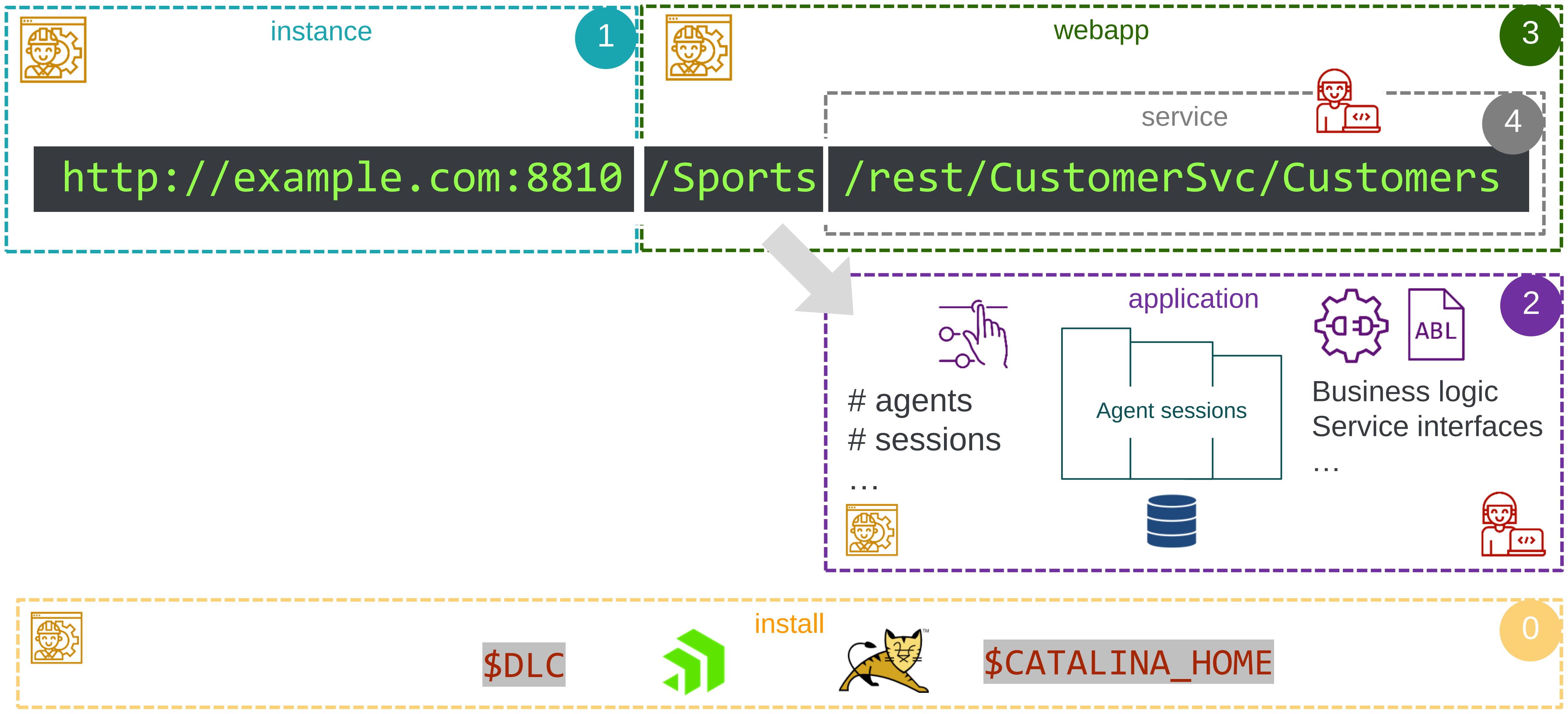
Why so many levels?

We don't know how you are planning to deploy your app(s). Nor do we want to force you into a specific model.

How to decide?

- Do you have more than one Classic AppServer and/or WebSpeed Server today as part of the application?
- How are these defined?
  - Business function
  - Technology / client type
  - "Reasons" / history

Deployment levels & URL space



(what goes into the) Packages

What do we want from a package?

- Smallest logical/common-sense unit
  - Separation of concerns … each level has own package
  - Composable into one or more business application
  - Independently built & versioned

- Packages should not be tightly-coupled to OE versions
  - Should only contain your-application-specific stuff

- Consumable in, producable from a CI/CD pipeline
  - Check-in and –out all artifacts (aka Infrastructure-as-Code)

1

Instance

Package	Deploy	
1. Zip up (working) instance's entire folder structure	`tcman register` Run tailoring	A

1. Manifest (version, name, etc)
2. `openedge.properties`
3. `catalina.properties`
4. Logging & other properties
5. Instance-common ABL
6. ABL applications packages
7. Standard webapps
8. Scripts for events, tailoring
	`tcman create`
	`oeprop -f <merge-file>`
	`tcman config`
	Copy ABL code to `/openedge`
	Copy files to `/webapps`, `/bin`
	Deploy ABL applications
	Run tailoring
		B

Instance zip vs. overlay package

A

Complete package

✓ Fast, easy, works

✓ Runnable

○ Tied to a specific OpenEdge version

– Eg Logging config, Spring config

B

Overlay package

✓ Version portable

✓ Smaller (fewer files)

✓ Composed of deployable sub-packages

○ More work (creation, testing)

Why not use both?

Create an overlay package, check it in, and use it to create a deployable zip

2

ABL Application

Package

1. Manifest (version, name, etc)
2. `openedge.properties`
3. `oeablSecurity.properties`
4. Business logic
5. ABL webapps packages
6. Scripts for events, tailoring

```
<abl-app-name>_startup.{bat|sh}
<abl-app-name>_started.{bat|sh}
<abl-app-name>_stopping.{bat|sh}
<abl-app-name>_shutdown.{bat|sh}
```

Deploy

Use an existing instance

```
tcman deploy \
-a <webapp-name> $dlc/…/oeabl.war\
<abl-app-name>
```

```
oeprop -f <merge-file>
```

```
secprop -f <merge-file>
```

Deploy ABL webapps

Copy scripts to `/bin`

Copy ABL code to `/ablapps/<abl-app-name>/openedge`

Run tailoring & restart

ABL WebApp

Package | **Deploy**

A

1. Export PDSOE project as ABL webapp (`.WAR`)

`tcman deploy`

Run tailoring

B

1. Manifest (version, name, etc)
2. `openedge.properties`
3. `oeablSecurity.properties`
4. Static files
5. ABL Services packages
6. Scripts for tailoring

```
tcman deploy –a <webapp-name>
      $dlc/…/oeabl.war
```

`oeprop –f <merge-file>`

`secprop –f <merge-file>`

Copy static files to `/static`, `/`

Deploy ABL services

Run tailoring

ABL Service

Package

1. Manifest (version, name, etc)
2. Transport-specific service descriptor (PAAR / GEN / WSM / handlers / JSON)
3. `openedge.properties`
4. `oeablSecurity.csv`
5. Service-interface ABL code
6. Static files
7. Scripts for events, tailoring

Deploy

```
deploySvc –a <webapp-name> <descriptor>
```

```
oeprop –f <merge-file>
```

```
secprop –f <merge-file>
```

Copy static files to `/static`

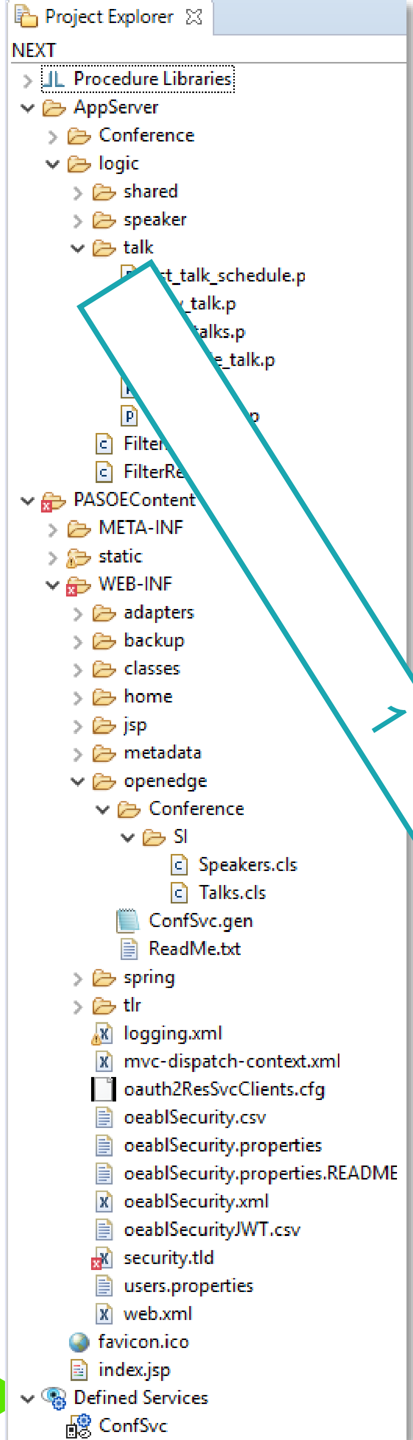
Copy ABL code to static files to `/WEB-INF/openedge`

Run tailoring

Where does it come from …
Package sources

PDSOE

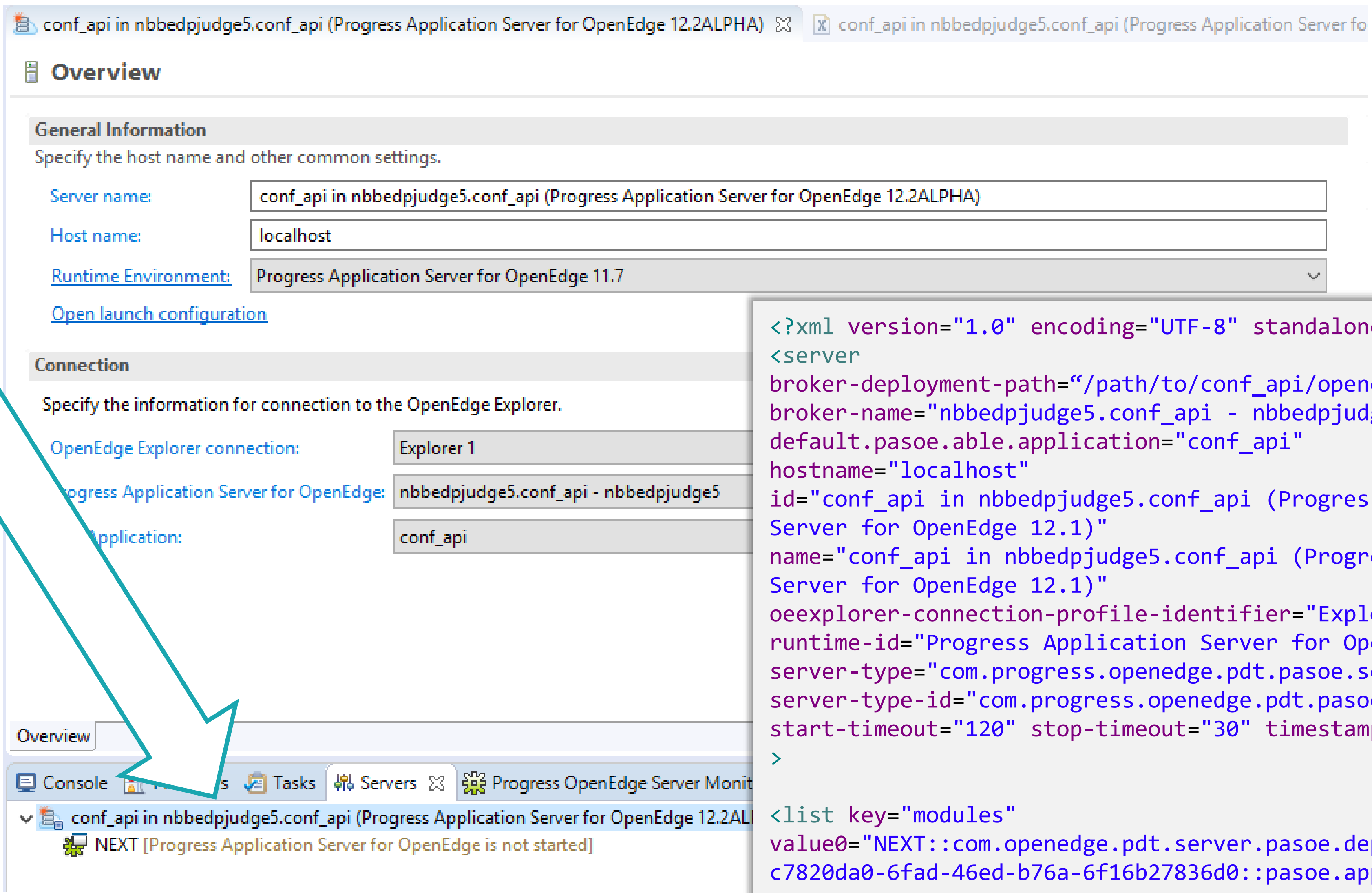
1



Project Explorer

NEXT

- Procedure Libraries
- AppServer
  - Conference
  - logic
    - shared
    - speaker
    - talk
      - t_talk_schedule.p
      - _talk.p
      - alks.p
      - e_talk.p
      - p
    - Filte
    - FilterRe
- PASOEContent
  - META-INF
  - static
  - WEB-INF
    - adapters
    - backup
    - classes
    - home
    - jsp
    - metadata
    - openedge
      - Conference
        - SI
          - Speakers.cls
          - Talks.cls
      - ConfSvc.gen
      - ReadMe.txt
    - spring
    - tlr
    - logging.xml
    - mvc-dispatch-context.xml
    - oauth2ResSvcClients.cfg
    - oeablSecurity.csv
    - oeablSecurity.properties
    - oeablSecurity.properties.README
    - oeablSecurity.xml
    - oeablSecurityJWT.csv
    - security.tld
    - users.properties
    - web.xml
  - favicon.ico
  - index.jsp
- Defined Services
  - ConfSvc



conf_api in nbbedpjudge5.conf_api (Progress Application Server for OpenEdge 12.2ALPHA) | conf_api in nbbedpjudge5.conf_api (Progress Application Server fo

Overview

General Information

Specify the host name and other common settings.

Server name:	conf_api in nbbedpjudge5.conf_api (Progress Application Server for OpenEdge 12.2ALPHA)
Host name:	localhost
Runtime Environment:	Progress Application Server for OpenEdge 11.7

Open launch configuration

Connection

Specify the information for connection to the OpenEdge Explorer.

OpenEdge Explorer connection:	Explorer 1
ogress Application Server for OpenEdge:	nbbedpjudge5.conf_api - nbbedpjudge5
Application:	conf_api

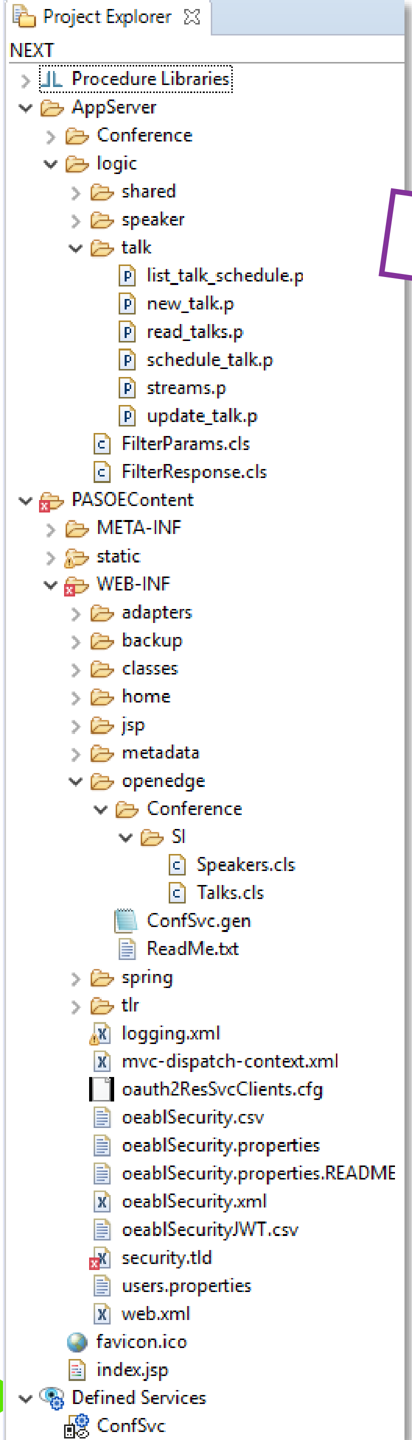
Overview

Console | Tasks | Servers | Progress OpenEdge Server Monit

conf_api in nbbedpjudge5.conf_api (Progress Application Server for OpenEdge 12.2AL

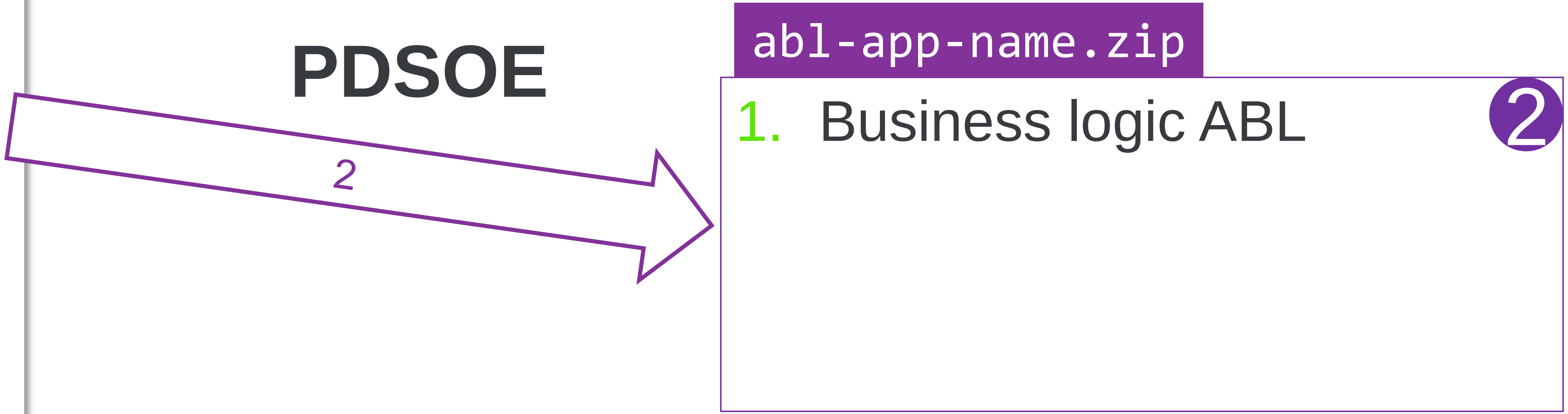
NEXT [Progress Application Server for OpenEdge is not started]

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<server
broker-deployment-path="“/path/to/conf_api/openedge"
broker-name="nbbedpjudge5.conf_api - nbbedpjudge5"
default.pasoe.able.application="conf_api"
hostname="localhost"
id="conf_api in nbbedpjudge5.conf_api (Progress Application
Server for OpenEdge 12.1)"
name="conf_api in nbbedpjudge5.conf_api (Progress Application
Server for OpenEdge 12.1)"
oeexplorer-connection-profile-identifier="Explorer_1"
runtime-id="Progress Application Server for OpenEdge 12.1"
server-type="com.progress.openedge.pdt.pasoe.servertype"
server-type-id="com.progress.openedge.pdt.pasoe.servertype"
start-timeout="120" stop-timeout="30" timestamp="2“
>

<list key="modules"
value0="NEXT::com.openedge.pdt.server.pasoe.deployable.component:
c7820da0-6fad-46ed-b76a-6f16b27836d0::pasoe.appserver::11.6" />

</server>
```

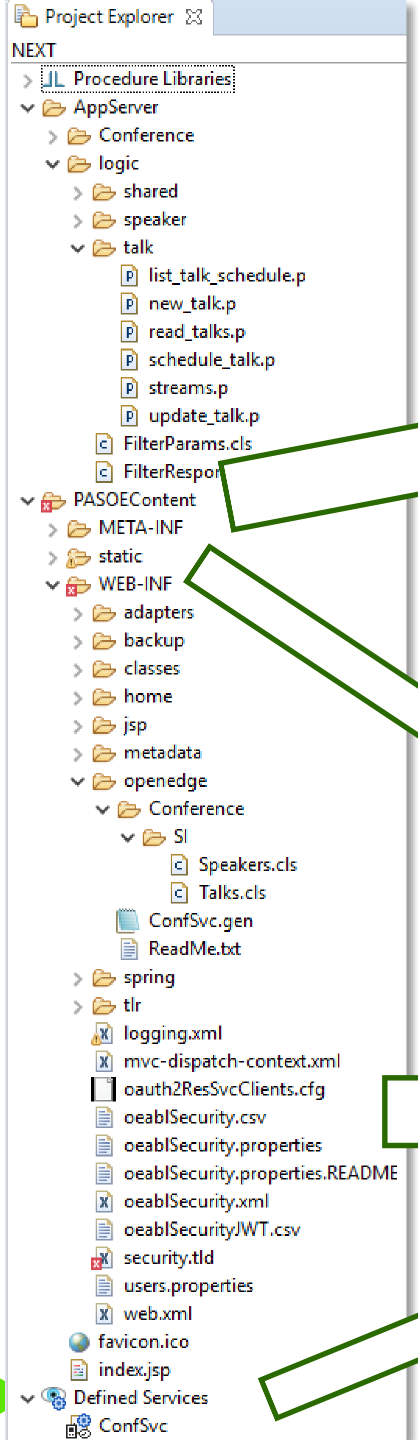

Project Explorer

NEXT
- Procedure Libraries
- AppServer
  - Conference
  - logic
    - shared
    - speaker
    - talk
      - list_talk_schedule.p
      - new_talk.p
      - read_talks.p
      - schedule_talk.p
      - streams.p
      - update_talk.p
    - FilterParams.cls
    - FilterResponse.cls
- PASOEContent
  - META-INF
  - static
  - WEB-INF
    - adapters
    - backup
    - classes
    - home
    - jsp
    - metadata
    - openedge
      - Conference
        - SI
          - Speakers.cls
          - Talks.cls
        - ConfSvc.gen
        - ReadMe.txt
    - spring
    - tlr
    - logging.xml
    - mvc-dispatch-context.xml
    - oauth2ResSvcClients.cfg
    - oeablSecurity.csv
    - oeablSecurity.properties
    - oeablSecurity.properties.README
    - oeablSecurity.xml
    - oeablSecurityJWT.csv
    - security.tld
    - users.properties
    - web.xml
  - favicon.ico
  - index.jsp
- Defined Services
  - ConfSvc

PDSOE

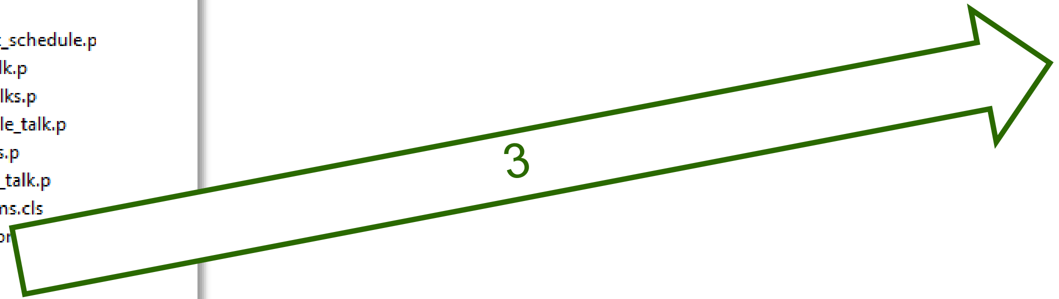


- PDSOE projects will only publish ABL code
- ABL Service-faceted projects have an `AppServer` folder (default name) for ABL


Project Explorer

NEXT
- Procedure Libraries
- AppServer
  - Conference
  - logic
    - shared
    - speaker
    - talk
      - list_talk_schedule.p
      - new_talk.p
      - read_talks.p
      - schedule_talk.p
      - streams.p
      - update_talk.p
    - FilterParams.cls
    - FilterRespo
- PASOEContent
  - META-INF
  - static
  - WEB-INF
    - adapters
    - backup
    - classes
    - home
    - jsp
    - metadata
    - openedge
      - Conference
        - SI
          - Speakers.cls
          - Talks.cls
      - ConfSvc.gen
      - ReadMe.txt
    - spring
    - tlr
    - logging.xml
    - mvc-dispatch-context.xml
    - oauth2ResSvcClients.cfg
    - oeablSecurity.csv
    - oeablSecurity.properties
    - oeablSecurity.properties.README
    - oeablSecurity.xml
    - oeablSecurityJWT.csv
    - security.tld
    - users.properties
    - web.xml
  - favicon.ico
  - index.jsp
- Defined Services
  - ConfSvc

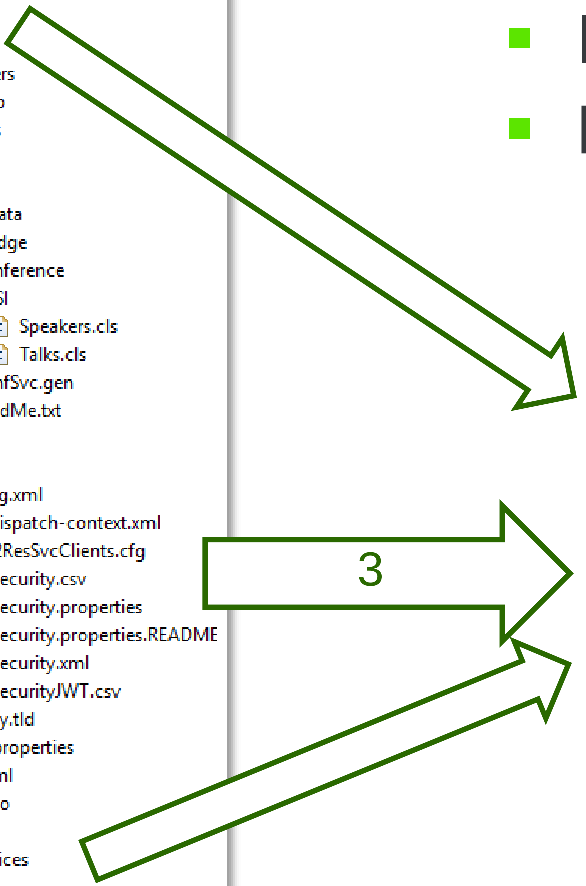
PDSOE



NEXT.war ③

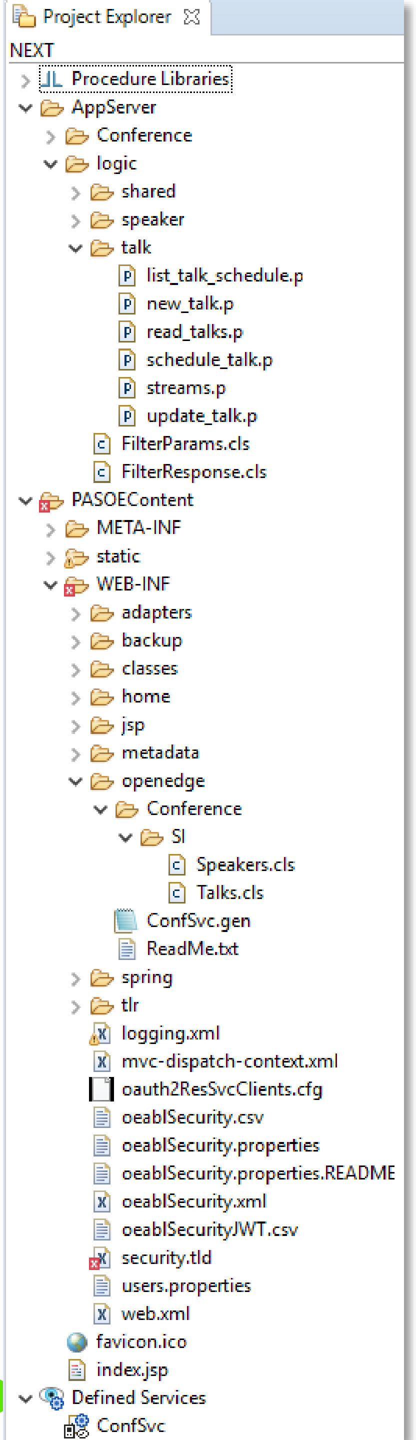
1. PASOEContent/**
2. Defined Services

- Includes selected ABL Services
- Deploy WAR to instance (dev mode)



NEXT.zip ③

1. WEB-INF/openedge
2. WEB-INF/tlr
3. Static files


Project Explorer

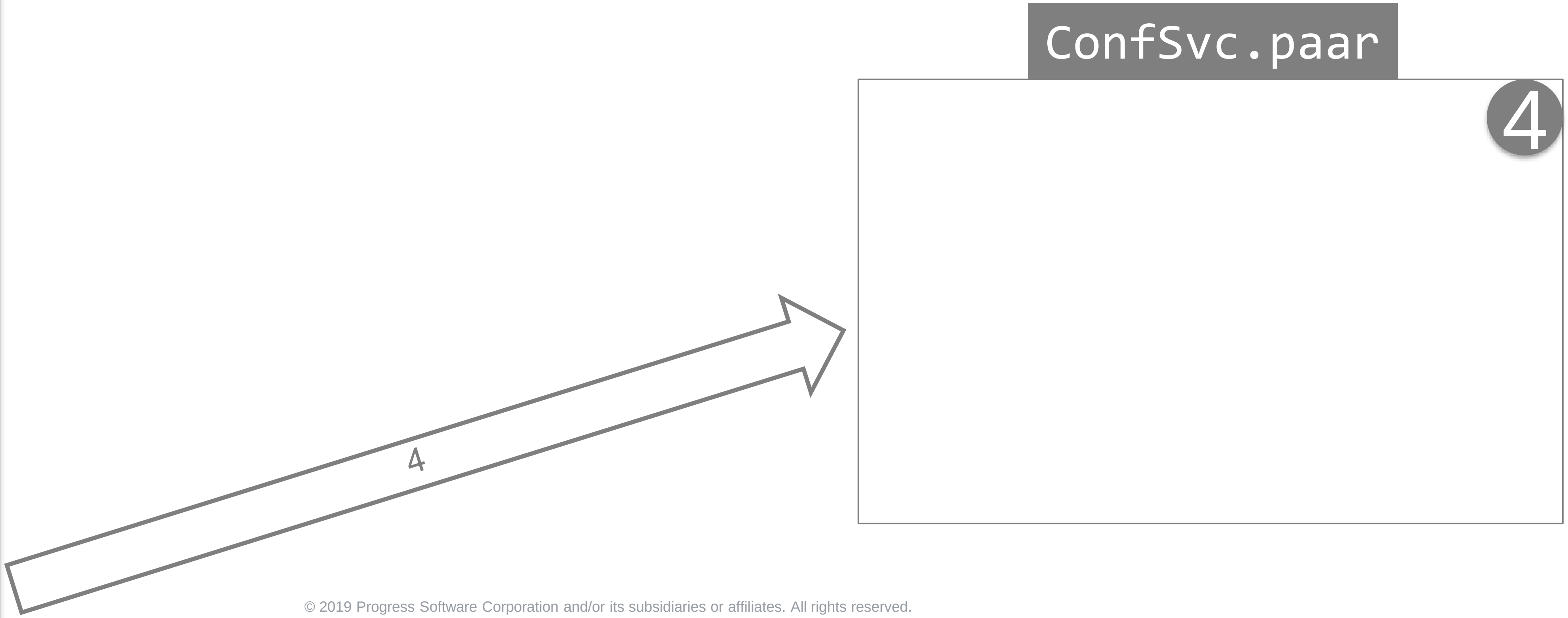
NEXT

- Procedure Libraries
- AppServer
  - Conference
  - logic
    - shared
    - speaker
    - talk
      - list_talk_schedule.p
      - new_talk.p
      - read_talks.p
      - schedule_talk.p
      - streams.p
      - update_talk.p
    - FilterParams.cls
    - FilterResponse.cls
- PASOEContent
  - META-INF
  - static
  - WEB-INF
    - adapters
    - backup
    - classes
    - home
    - jsp
    - metadata
    - openedge
      - Conference
        - SI
          - Speakers.cls
          - Talks.cls
      - ConfSvc.gen
      - ReadMe.txt
    - spring
    - tlr
    - logging.xml
    - mvc-dispatch-context.xml
    - oauth2ResSvcClients.cfg
    - oeablSecurity.csv
    - oeablSecurity.properties
    - oeablSecurity.properties.README
    - oeablSecurity.xml
    - oeablSecurityJWT.csv
    - security.tld
    - users.properties
    - web.xml
  - favicon.ico
  - index.jsp
- Defined Services
  - ConfSvc

PDSOE

- Can export REST services incrementally as a PAAR file from the `PaarGeneration` task

  https://docs.progress.com/bundle/developer-studio-olh/page/Packaging-REST-services.html



Where does it go?
Instance (physical) targets

Install

```
/usr/bin/oe121
├── bin
├── servers
│   ├── pasoe
│   │   ├── bin
│   │   ├── common
│   │   ├── extras
│   │   ├── lib
```

- `$DLC/bin` contains the agent executable `_mproapsv`
- The default `$CATALINA_HOME` is `$DLC/servers/pasoe`
  - Contains binaries used to run the instance
- `$DLC/servers/pasoe/extras` contains webapps (incl. baseline ABL webapp)
- See info about instances by running `tcman env`

Instance structure

```
oepas1
├── ablapps
├── bin
├── common
├── conf
├── logs
├── openedge
├── temp
├── webapps
└── work
```

- The instance is the runtime *context* of applications
  - Scripts/binaries
  - Configuration
  - Logs
- Hooked up to `$DLC` executables via `tcman register` command
- A container of applications and standard webapps
  - Some top-level folders contain stuff from the (grand)children

    `/conf`, `/bin`, `/logs`, `/temp`

ABL apps structure

```
ablapps/
├── abl-app-1
│   ├── conf
│   ├── openedge
│   └── temp
└── abl-app-2
```

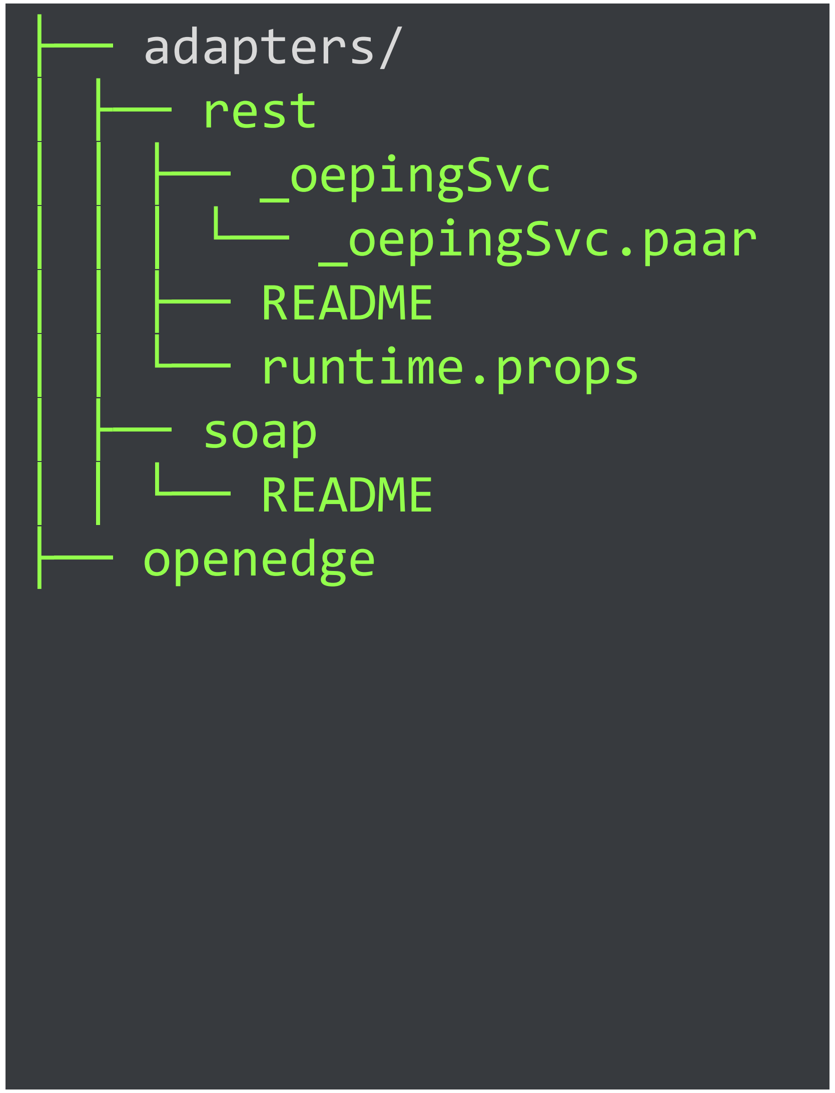
- The ABL application is the smallest deployment unit that can be run on its own
  - Runtime configuration - # agents, # sessions, PROPATH, db connections etc
  - Security configuration – default authentication model for all its contents
- Contributions to the instance
  - Scripts/binaries
  - **Configuration**
  - Logs
- A container of ABL webapps that are deployed into the instance's `/webapps` folder

ABL webapps structure

```
webapps/
├── oemanager
└── ROOT
    ├── META-INF
    ├── static
    │   ├── auth
    │   ├── error
    │   └── images
    └── WEB-INF
        ├── adapters
        │   ├── rest
        │   └── soap
        ├── backup
        ├── classes
        ├── jsp
        ├── metadata
        ├── openedge
        ├── spring
        └── tlr
```

- Exposes the application's business logic via URL ⚠ *the only way to call/run ABL*
- Provides (enforces) the service
  - Authentication
  - Authorization (provided by ABL service)
- Contributions to the instance
  - Configuration
- A container of ABL services, grouped by "transport"
  - `/static` is a special case

ABL services structure



```
├── adapters/
│   ├── rest
│   │   ├── _oepingSvc
│   │   │   └── _oepingSvc.paar
│   │   ├── README
│   │   └── runtime.props
│   ├── soap
│   │   └── README
├── openedge
```

- ABL Service are the API that exposes service interfaces
  - Descriptor (`.PAAR`, `.GEN`, `.WSM`, `.handlers`)
  - Service interface code (ABL or other)
  - Authorization configuration
- Contributions to ABL webapp
  - ABL code `/WEB-INF/openedge/*`
  - Intercept-urls `/WEB-INF/oeablSecurity.csv`
- Contributions to instance
  - Configuration `adapterEnabled=1`
- Categorised by "transport"
  - **APSV**, **WEB**, REST, SOAP, STATIC

Instance folder structure

1

```
oepas1
├── ablapps
├── bin
├── common
├── conf
├── logs
├── openedge
├── temp
├── webapps
└── work
```

2

```
ablapps/
├── abl-app-1
│   ├── conf
│   ├── openedge
│   └── temp
└── abl-app-2
```

3

```
webapps/
├── oemanager
└── ROOT
    ├── META-INF
    ├── static
    │   ├── auth
    │   ├── error
    │   └── images
    └── WEB-INF
        ├── adapters
        │   ├── rest
        │   └── soap
        ├── backup
        ├── classes
        ├── jsp
        ├── metadata
        ├── openedge
        ├── spring
        └── tlr
```

4

```
├── adapters/
│   ├── rest
│   │   ├── _oepingSvc
│   ├── soap
├── openedge
```

Progress

Instance properties

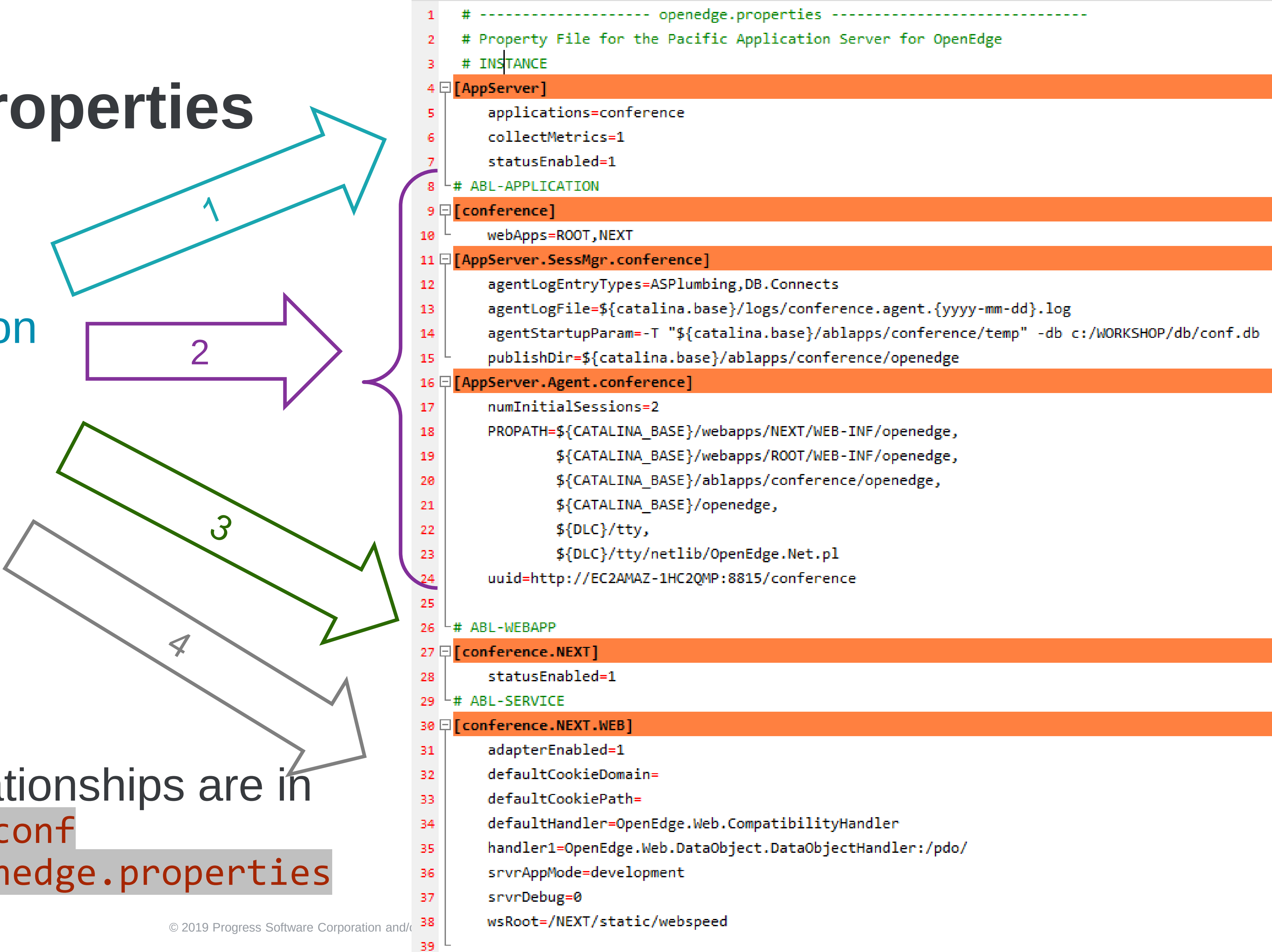
0 Install

1 Instance

2 ABL Application

3 ABL Webapp

4 ABL Service



```
# -------------------- openedge.properties -----------------------------
# Property File for the Pacific Application Server for OpenEdge
# INSTANCE
[AppServer]
    applications=conference
    collectMetrics=1
    statusEnabled=1
# ABL-APPLICATION
[conference]
    webApps=ROOT,NEXT
[AppServer.SessMgr.conference]
    agentLogEntryTypes=ASPlumbing,DB.Connects
    agentLogFile=${catalina.base}/logs/conference.agent.{yyyy-mm-dd}.log
    agentStartupParam=-T "${catalina.base}/ablapps/conference/temp" -db c:/WORKSHOP/db/conf.db
    publishDir=${catalina.base}/ablapps/conference/openedge
[AppServer.Agent.conference]
    numInitialSessions=2
    PROPATH=${CATALINA_BASE}/webapps/NEXT/WEB-INF/openedge,
            ${CATALINA_BASE}/webapps/ROOT/WEB-INF/openedge,
            ${CATALINA_BASE}/ablapps/conference/openedge,
            ${CATALINA_BASE}/openedge,
            ${DLC}/tty,
            ${DLC}/tty/netlib/OpenEdge.Net.pl
    uuid=http://EC2AMAZ-1HC2QMP:8815/conference

# ABL-WEBAPP
[conference.NEXT]
    statusEnabled=1
# ABL-SERVICE
[conference.NEXT.WEB]
    adapterEnabled=1
    defaultCookieDomain=
    defaultCookiePath=
    defaultHandler=OpenEdge.Web.CompatibilityHandler
    handler1=OpenEdge.Web.DataObject.DataObjectHandler:/pdo/
    srvrAppMode=development
    srvrDebug=0
    wsRoot=/NEXT/static/webspeed

```

Parent-child relationships are in `$CATALINA_BASE/conf /openedge.properties`

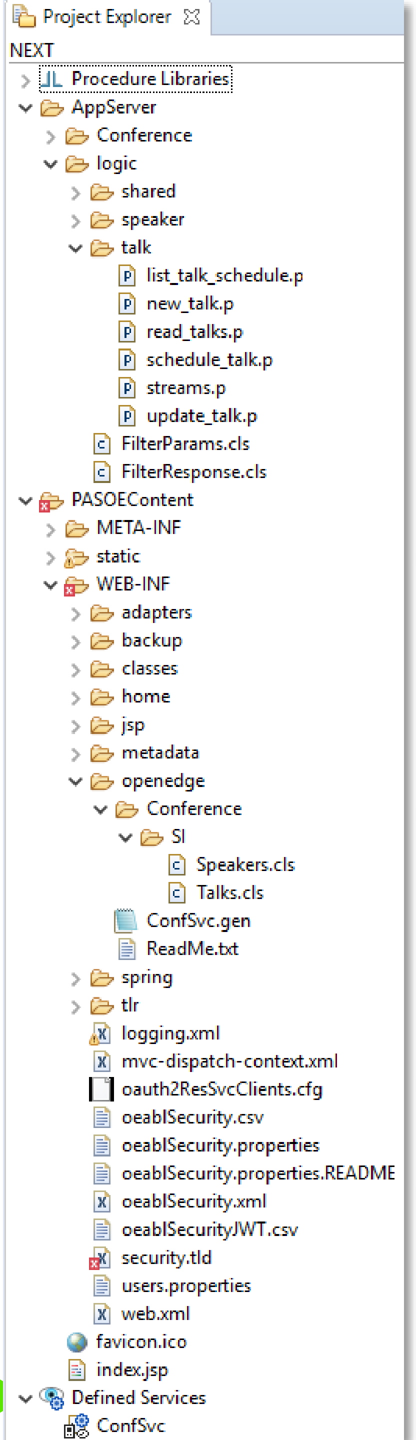
Looking forward …

- Well-defined packages for each level
  - Manifests, tailoring, etc
- Creating CLI-based tooling
  - `export` flows to create packages with a defined structure
  - `deploy` (and `undeploy` and `patch`) to apply those package contents to an instance
- Ant-based tailoring on each of these operations, for each level

`_export`	`_exported`
`_deploy`	`_deployed`
`_undeploy`	`_undeployed`
`_patch`	`_patched`

- We see the primary use-cases around automation (CI/CD)

  … but we also want this flow to work from IDEs and other developer tooling


Project Explorer

NEXT
- Procedure Libraries
- AppServer
  - Conference
  - logic
    - shared
    - speaker
    - talk
      - list_talk_schedule.p
      - new_talk.p
      - read_talks.p
      - schedule_talk.p
      - streams.p
      - update_talk.p
    - FilterParams.cls
    - FilterResponse.cls
- PASOEContent
  - META-INF
  - static
  - WEB-INF
    - adapters
    - backup
    - classes
    - home
    - jsp
    - metadata
    - openedge
      - Conference
        - SI
          - Speakers.cls
          - Talks.cls
      - ConfSvc.gen
      - ReadMe.txt
    - spring
    - tlr
    - logging.xml
    - mvc-dispatch-context.xml
    - oauth2ResSvcClients.cfg
    - oeablSecurity.csv
    - oeablSecurity.properties
    - oeablSecurity.properties.README
    - oeablSecurity.xml
    - oeablSecurityJWT.csv
    - security.tld
    - users.properties
    - web.xml
  - favicon.ico
  - index.jsp
- Defined Services
  - ConfSvc

2

3

4

4

3

4

Better

2

1. Manifest (version, name, etc)
2. `openedge.properties`
3. `oeablSecurity.properties`
4. App-common ABL
5. Scripts for events, tailoring

4

1. Manifest (version, name, etc)
2. Transport-specific service descriptor (`PAAR` / `GEN` / `WSM` / `handlers` / `JSON`)
3. `openedge.properties`
4. `oeablSecurity.csv`
5. Service-interface ABL code
6. Static files
7. Scripts for events, tailoring

3

1. Manifest (version, name, etc)
2. `openedge.properties`
3. `oeablSecurity.properties`
4. Static files
5. ABL Services packages
6. Scripts for tailoring

Conclusion

- Described the component parts of an instance, at a logical level, for defining deployment packages

- Described what should be in a deployment package and how to apply them to an instance

But wait, there’s more!

WEDNESDAY

385: Deploying Applications with the Docker Container for Progress Application Server for OpenEdge

09:45 Roy Ellis, PSC

430: Beyond the Code: Implementing DevOps and CI/CD Techniques for Cloud Apps

11:00 Edsel Garcia, PSC

Questions?

pjudge@progress.com

Progress®