OSGi™ Community Event 2011

Structuring Software Systems with OSGi

Ulf Fildebrandt

SAP AG

22.09.2011

Agenda

- OSGi: basic structure and declaration
- Interface Usage
- Layers
- Services and Layers
- Versioning
- Testing
- Assemblies

Agenda

- **OSGi: basic structure and declaration**
- Interface Usage
- Layers
- Services and Layers
- Versioning
- Testing
- Assemblies

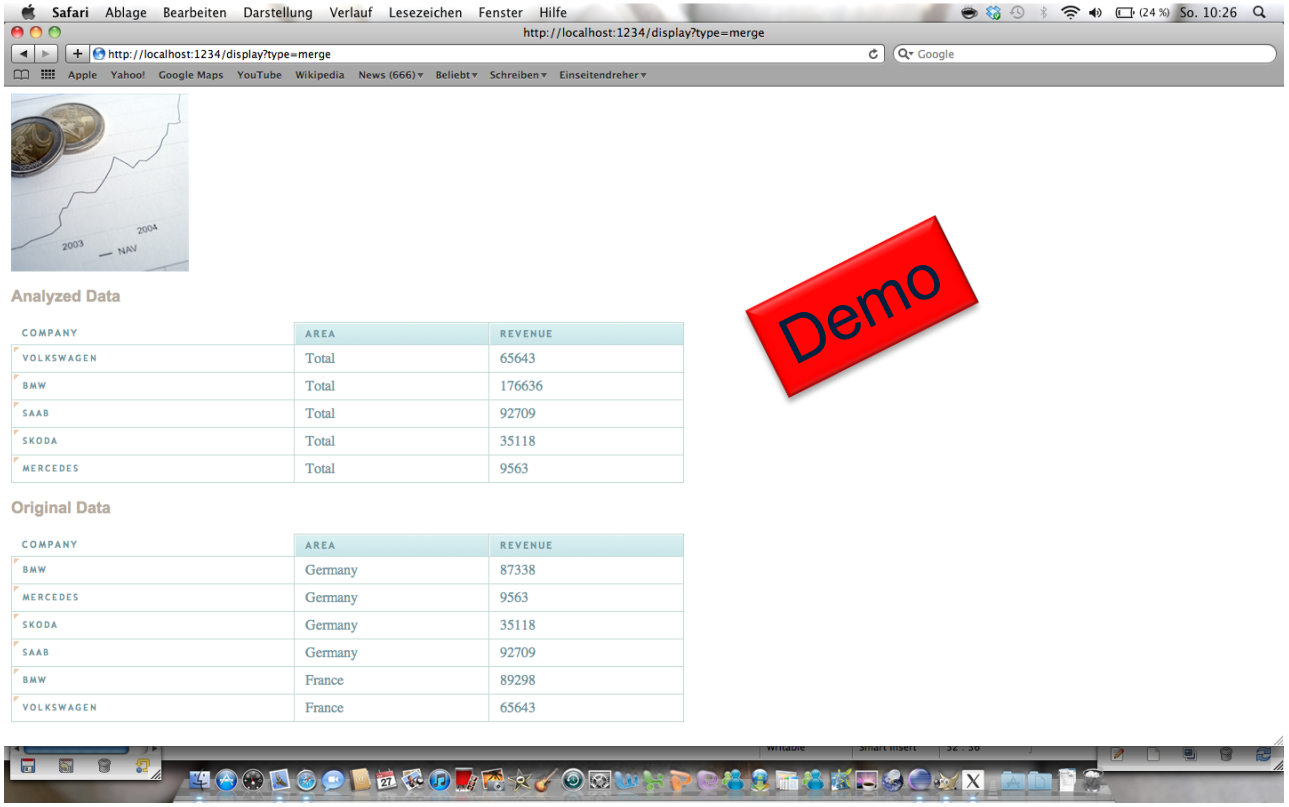
Principles of Modularity

- Decoupling
  - Logic has to be decoupled, otherwise the structure evolves into a monolithic block („architecture erosion“)
- Simplicity
  - Design the concepts as simple as possible, otherwise the system solves non-existing requirements
- Extensibility
  - Decoupling implies that frameworks are open for logic
  - Extensibility for logic is not the exception, but rather the common principle for all frameworks

OSGi basics

- **Bundles** are the atomic entities for structuring (containing binaries)
- **Services** are the communication means between bundles
- **Exported packages** are the public interface of bundles
- **Imported packages** define the usages of bundles

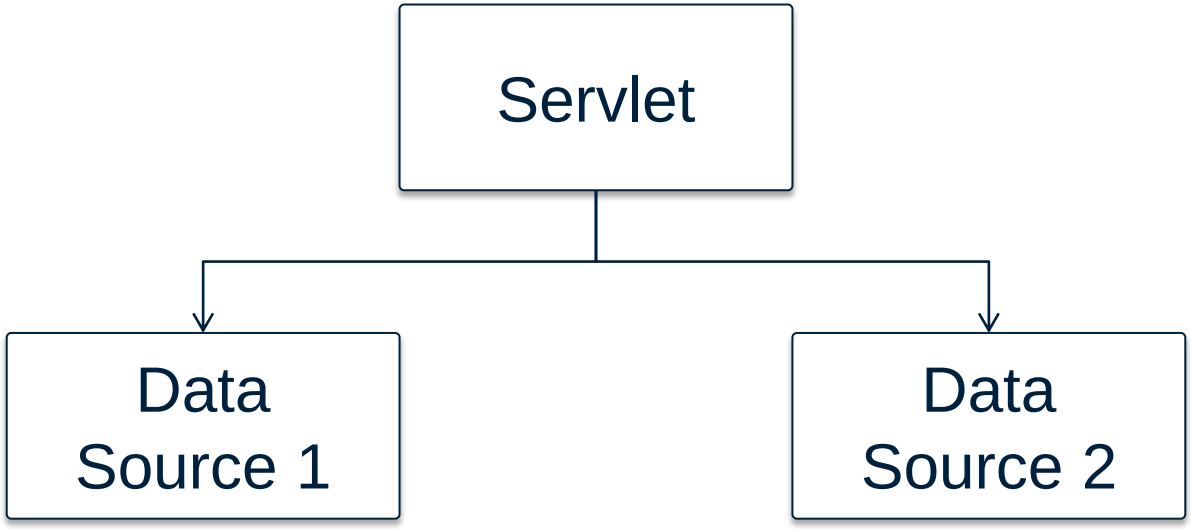
Example



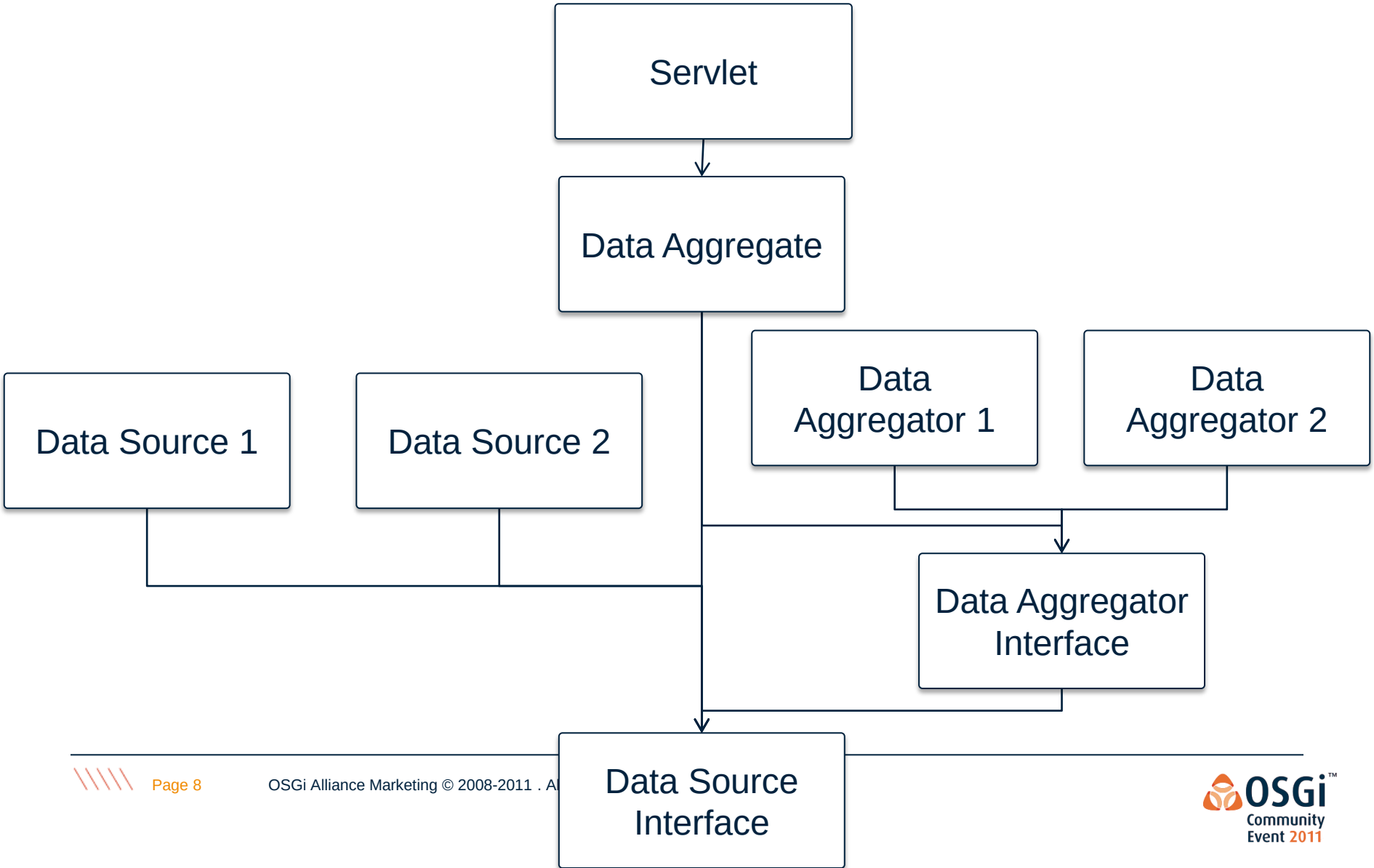
- Web application to analyze data
- OSGi bundles, services, and exported packages are used to structure the application

Structure of Application

- Fulfills the business requirements, but no modularity (decoupling, extensibility)



Structure Of Application



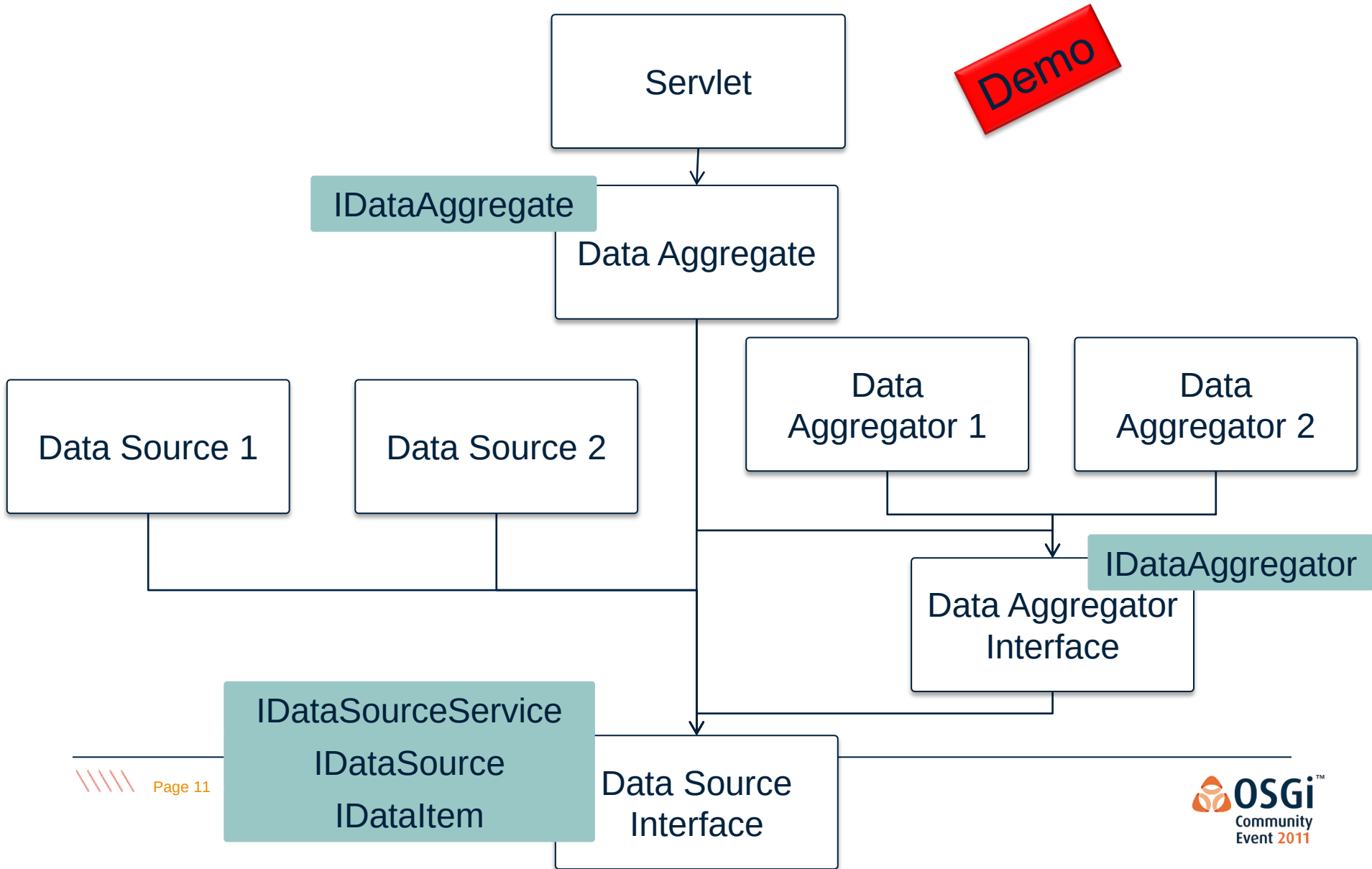
Agenda

- OSGi: basic structure and declaration
- **Interface Usage**
- Layers
- Services and Layers
- Versioning
- Testing
- Assemblies

Interface Usage

- Implementation should only use interfaces → Services in OSGi
- Services are difficult to implement in OSGi fulfilling all lifecycle requirements (Service Tracker, Service Listener) → „boilerplate coding“
- Frameworks: **Declarative Services** (DS) is part of the OSGi specification, Blueprint, etc.
  - DS are meant to declare the dependencies of services and service usage
  - DS is a mechanism to bring dependency injection into OSGi

Structure Of Application

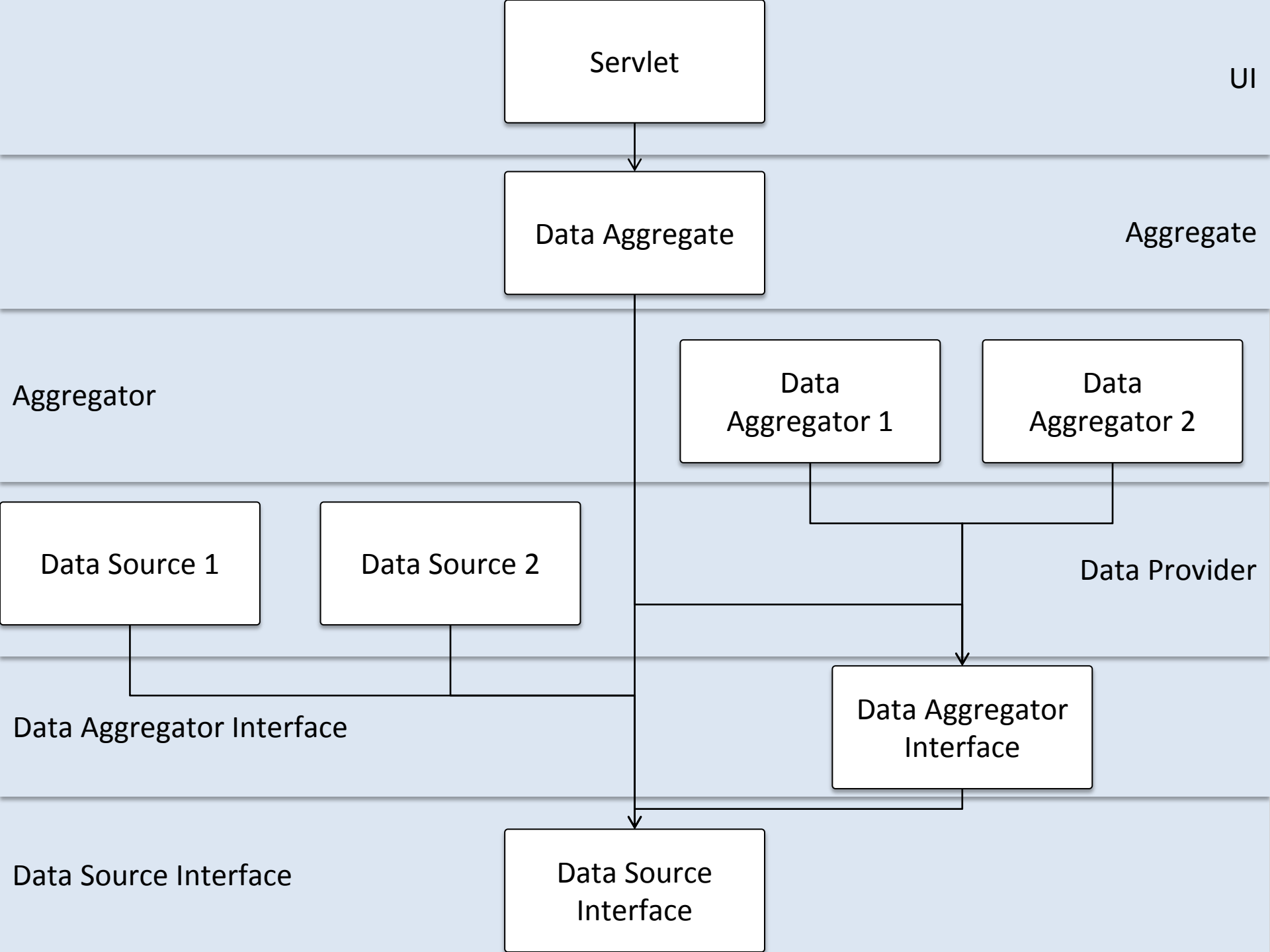


Agenda

- OSGi: basic structure and declaration
- Interface Usage
- **Layers**
- Services and Layers
- Versioning
- Testing
- Assemblies

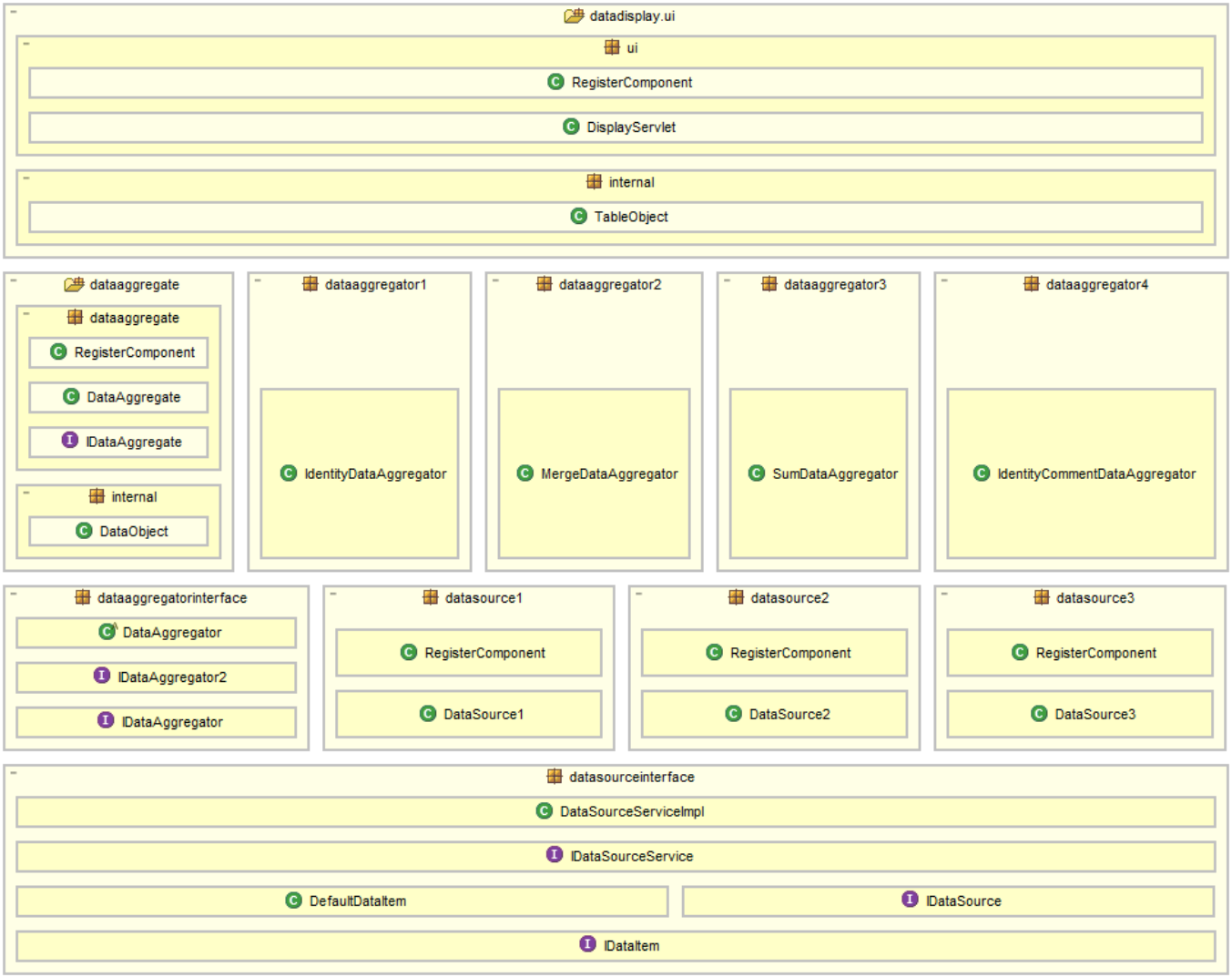
Layers

- In architecture, layers are used to structure the software
- Entities within a layer can only have dependencies to lower layers
- Bundles in OSGi belong to one layer only
- In reality more coarse grained frameworks are used
  - Feature (P2, Eclipse)
  - Application (Aries)
  - OSGi sub system specification
- For simplicity in these examples only bundles are used


Servlet
UI
Data Aggregate
Aggregate
Aggregator
Data
Aggregator 1
Data
Aggregator 2
Data Source 1
Data Source 2
Data Provider
Data Aggregator Interface
Data Aggregator
Interface
Data Source Interface
Data Source
Interface

Class Structure

- Classes fit to bundle structure



Agenda

- OSGi: basic structure and declaration
- Interface Usage
- Layers
- **Services and Layers**
- Versioning
- Testing
- Assemblies

Interface of a Layer

- Layers should have a clear interface
- API (Application Programming Interface)
  - is defining the publicly available functionality
  - is an interface
  - is implemented as service in OSGi
  - Example: IDataAggregate, IDataSourceService
- SPI (Service Provider Interface)
  - is defining the extension point
  - is implemented as service in OSGi in a separate bundle
  - Example: IDataSource, IDataAggregator

Agenda

- OSGi: basic structure and declaration
- Interface Usage
- Layers
- Services and Layers
- **Versioning**
- Testing
- Assemblies

Versioning in OSGi

- Bundles and exported packages are versioned
- Usage of versioning is different for implementation, API, SPI
  - Implementation: can be replaced easily, because interface is not changed
  - API: can be evolved in a compatible way, but implementation has to be adapted, too
  - SPI: evolution is difficult, because if interfaces are extended, then the old implementations are not valid any more
    - Java evolution: Ix → Ix2, e.g. IDataAggregator → IDataAggregator2 in the same package

Agenda

- OSGi: basic structure and declaration
- Interface Usage
- Layers
- Services and Layers
- Versioning
- **Testing**
- Assemblies

Testing

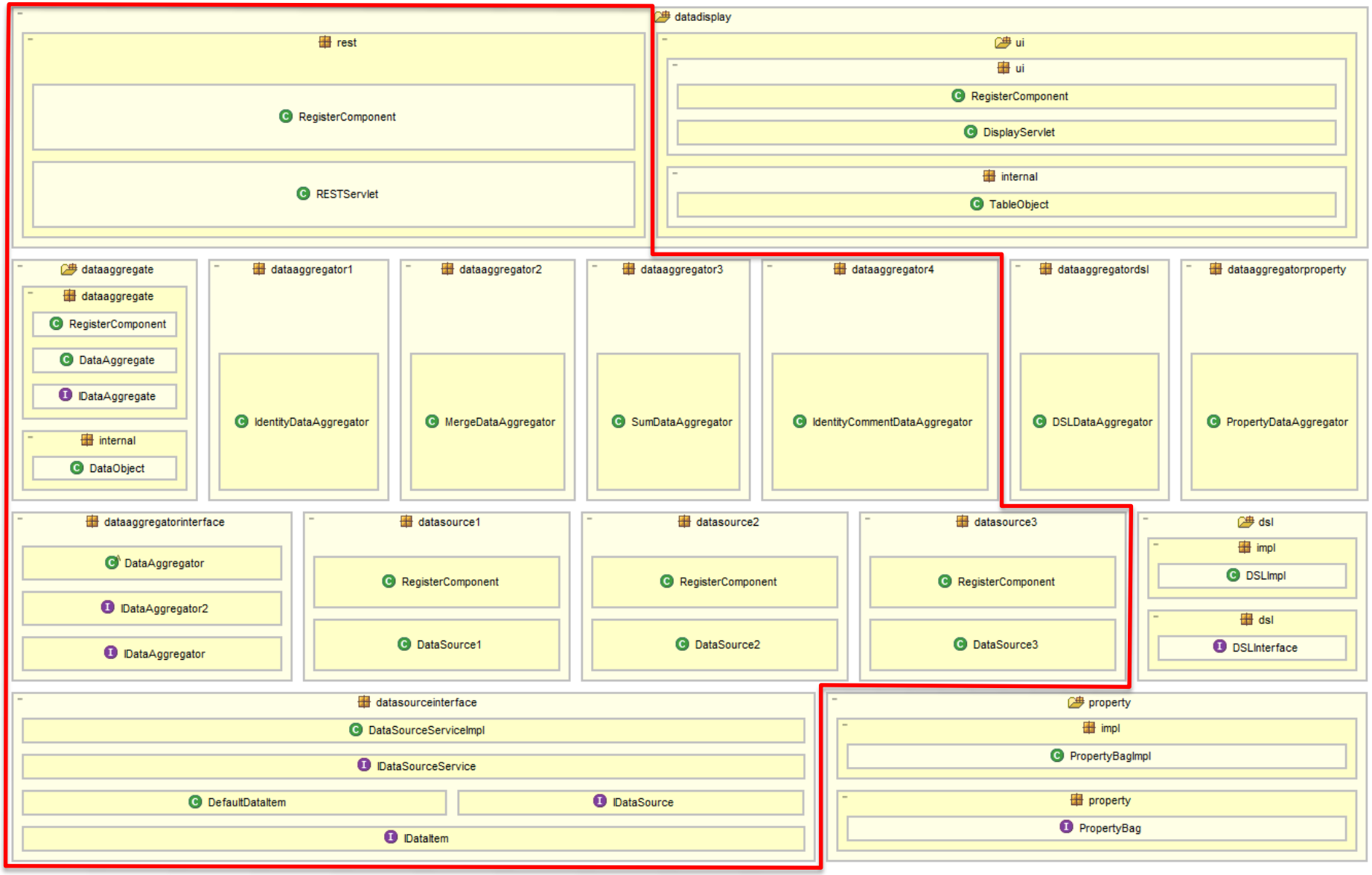
- Layers have to be separately testable
  - Mock objects (Easy Mock)
- Example: DataAggregate layer

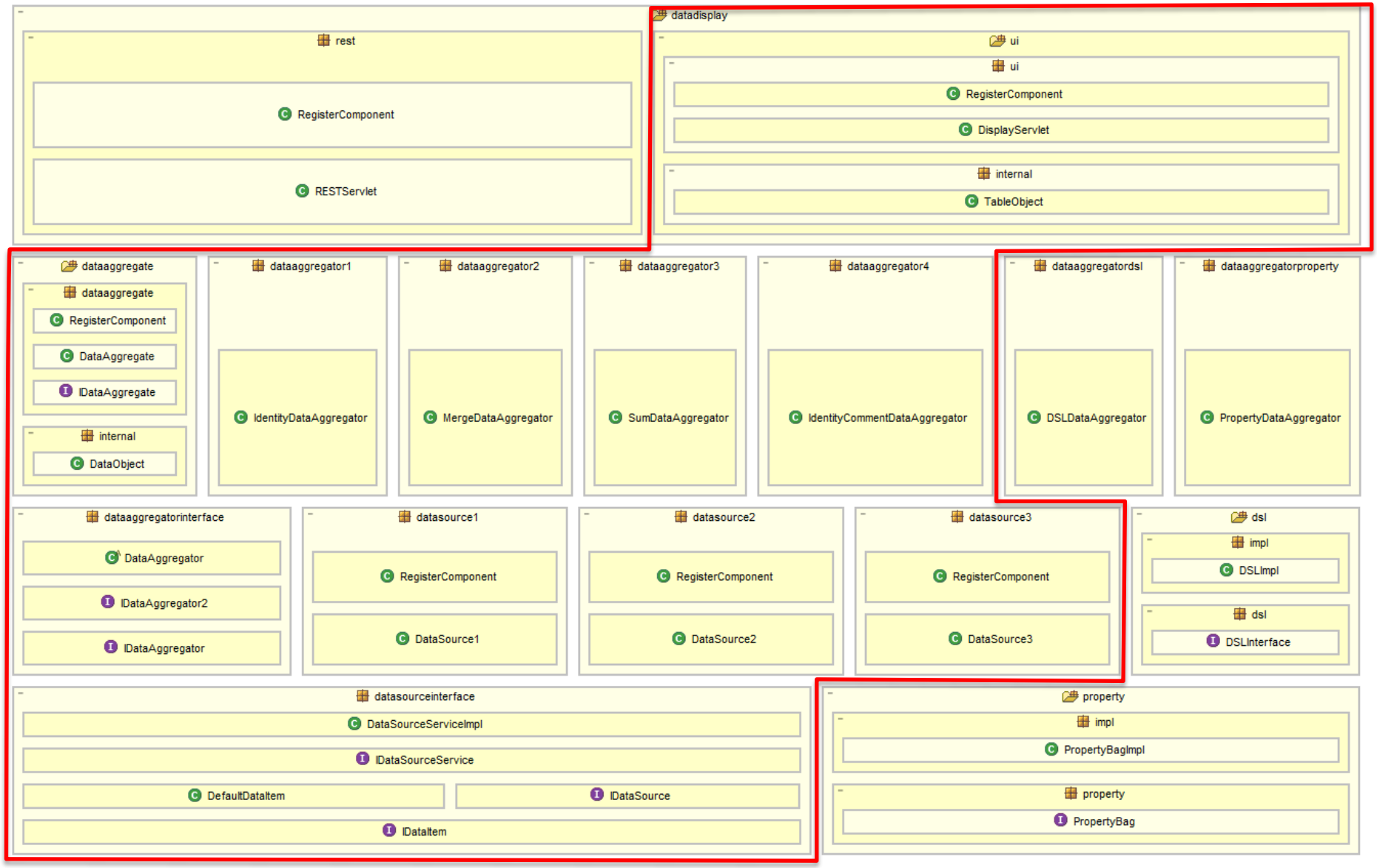
Agenda

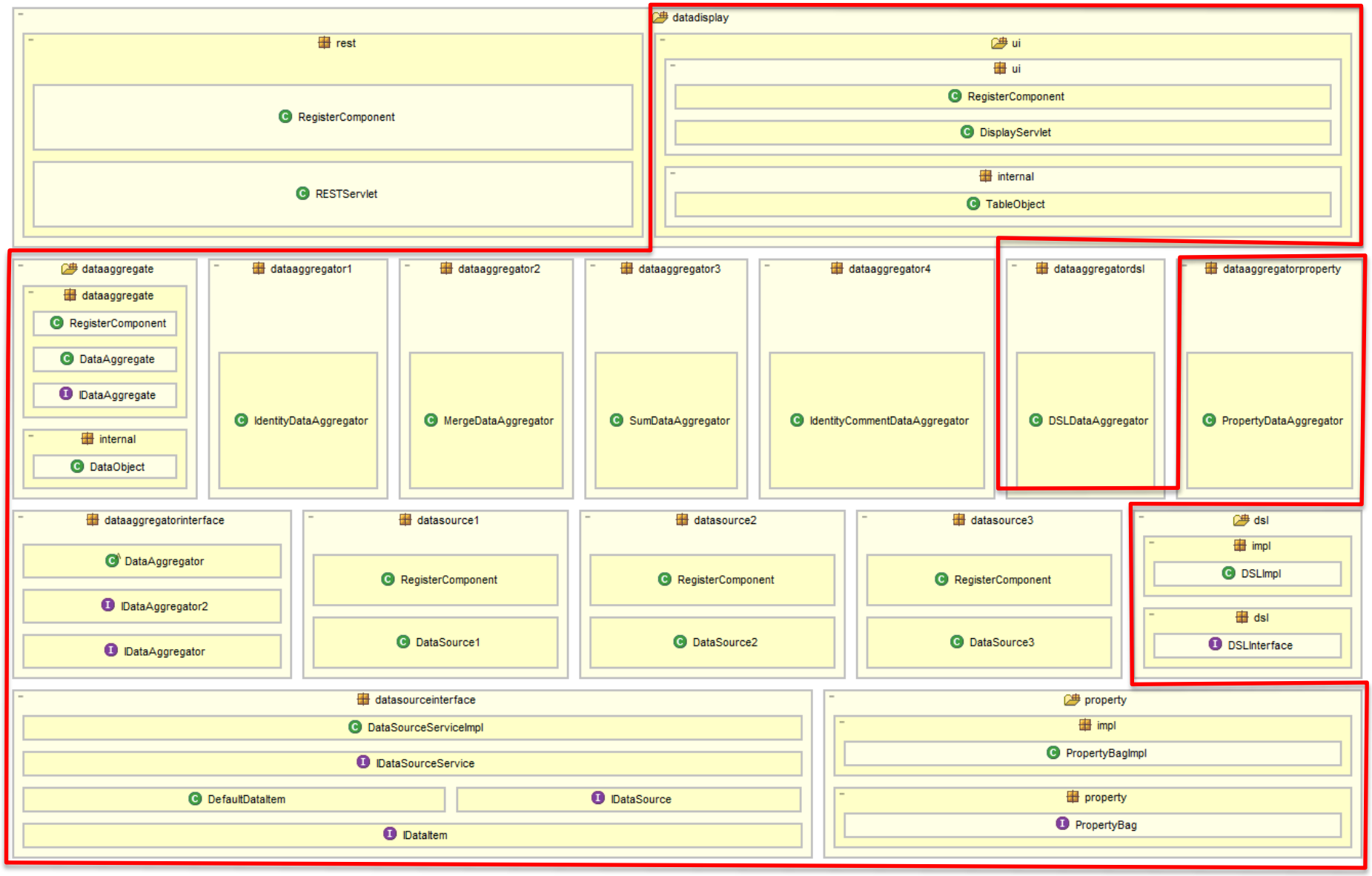
- OSGi: basic structure and declaration
- Interface Usage
- Layers
- Services and Layers
- Versioning
- Testing
- **Assemblies**

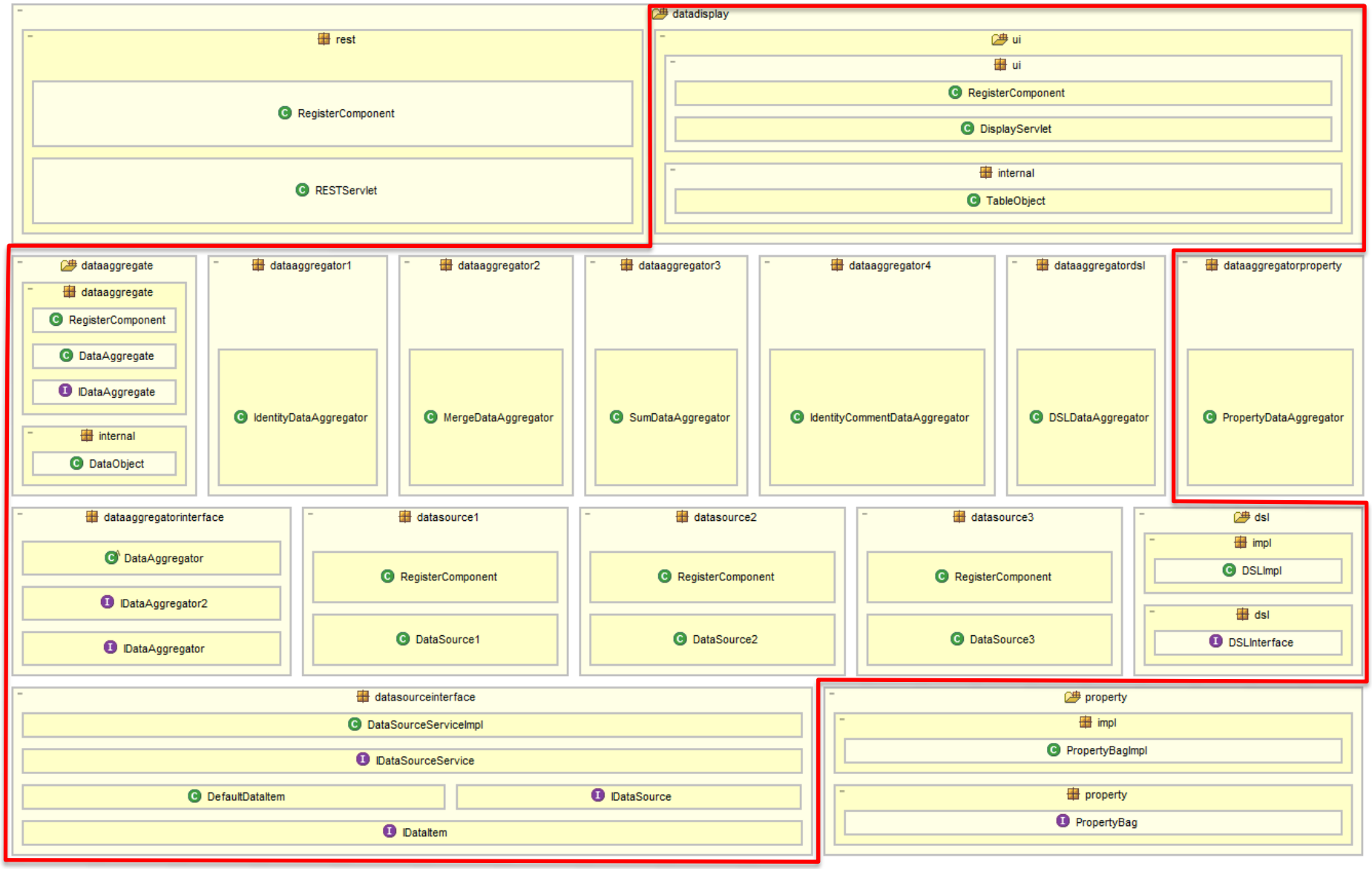
Multiple assemblies

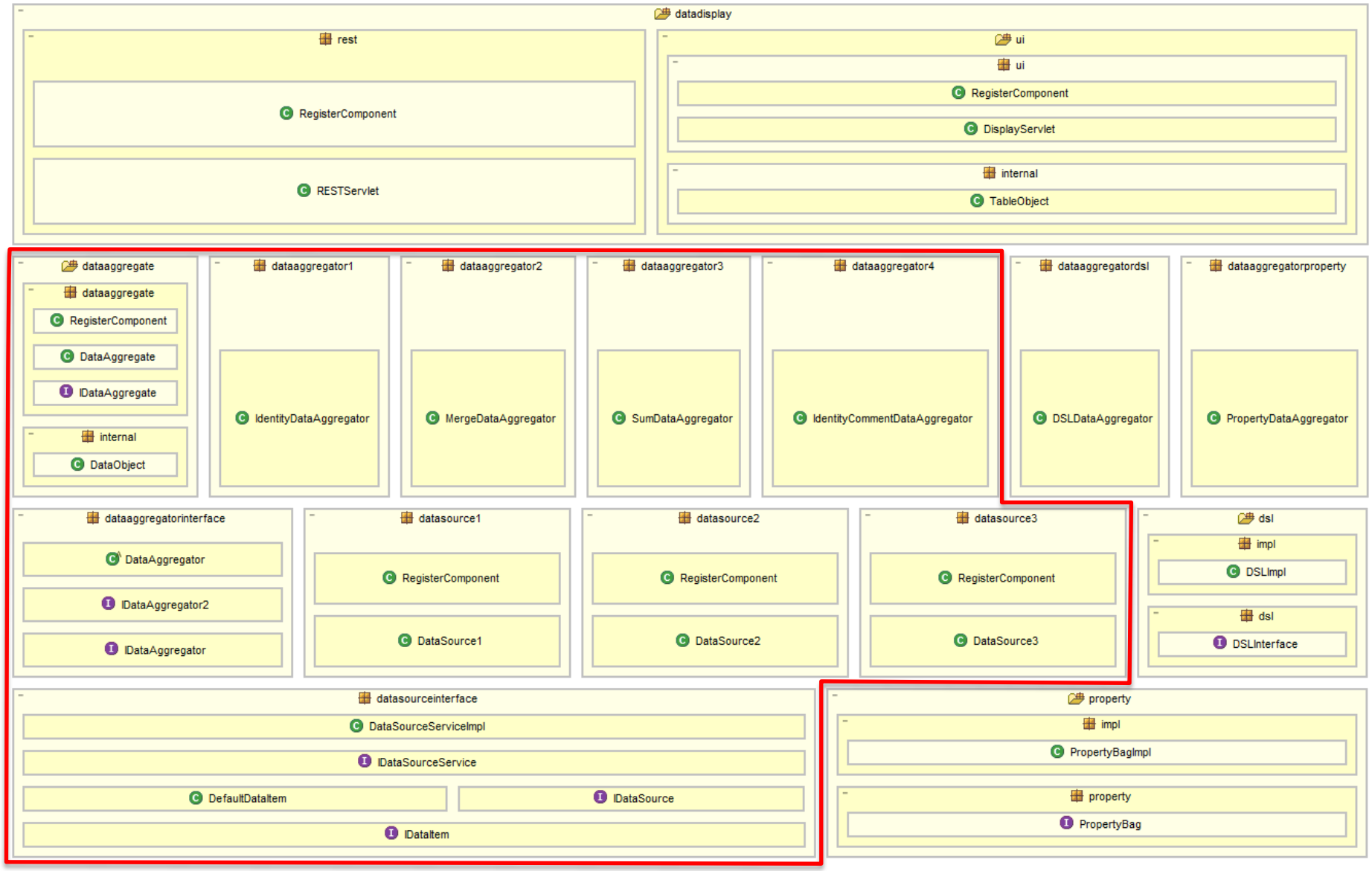
- Bundles assembled into multiple assemblies (e.g. products)
- An assembly contains bundles from different layers
  - A layer does not have to be used completely
- Examples:
  - Other access layer (REST)
  - Configuration data aggregator
  - DSL data aggregator


rest
RegisterComponent
RESTServlet
datadisplay
ui
ui
RegisterComponent
DisplayServlet
internal
TableObject
dataaggregate
dataaggregate
RegisterComponent
DataAggregate
IDataAggregate
internal
DataObject
dataaggregator1
IdentityDataAggregator
dataaggregator2
MergeDataAggregator
dataaggregator3
SumDataAggregator
dataaggregator4
IdentityCommentDataAggregator
dataaggregatordsl
DSLDataAggregator
dataaggregatorproperty
PropertyDataAggregator
dataaggregatorinterface
DataAggregator
IDataAggregator2
IDataAggregator
datasource1
RegisterComponent
DataSource1
datasource2
RegisterComponent
DataSource2
datasource3
RegisterComponent
DataSource3
dsl
impl
DSLImpl
dsl
DSLInterface
datasourceinterface
DataSourceServiceImpl
IDataSourceService
DefaultDataItem
IDataSource
IDataItem
property
impl
PropertyBagImpl
property
PropertyBag


rest
RegisterComponent
RESTServlet
datadisplay
ui
ui
RegisterComponent
DisplayServlet
internal
TableObject
dataaggregate
dataaggregate
RegisterComponent
DataAggregate
IDataAggregate
internal
DataObject
dataaggregator1
IdentityDataAggregator
dataaggregator2
MergeDataAggregator
dataaggregator3
SumDataAggregator
dataaggregator4
IdentityCommentDataAggregator
dataaggregatordsl
DSLDataAggregator
dataaggregatorproperty
PropertyDataAggregator
dataaggregatorinterface
DataAggregator
IDataAggregator2
IDataAggregator
datasource1
RegisterComponent
DataSource1
datasource2
RegisterComponent
DataSource2
datasource3
RegisterComponent
DataSource3
dsl
impl
DSLImpl
dsl
DSLInterface
datasourceinterface
DataSourceServiceImpl
IDataSourceService
DefaultDataItem
IDataSource
IDataItem
property
impl
PropertyBagImpl
property
PropertyBag


rest
RegisterComponent
RESTServlet
datadisplay
ui
ui
RegisterComponent
DisplayServlet
internal
TableObject
dataaggregate
dataaggregate
RegisterComponent
DataAggregate
IDataAggregate
internal
DataObject
dataaggregator1
IdentityDataAggregator
dataaggregator2
MergeDataAggregator
dataaggregator3
SumDataAggregator
dataaggregator4
IdentityCommentDataAggregator
dataaggregatordsl
DSLDataAggregator
dataaggregatorproperty
PropertyDataAggregator
dataaggregatorinterface
DataAggregator
IDataAggregator2
IDataAggregator
datasource1
RegisterComponent
DataSource1
datasource2
RegisterComponent
DataSource2
datasource3
RegisterComponent
DataSource3
dsl
impl
DSLImpl
dsl
DSLInterface
datasourceinterface
DataSourceServiceImpl
IDataSourceService
DefaultDataItem
IDataSource
IDataItem
property
impl
PropertyBagImpl
property
PropertyBag


rest
RegisterComponent
RESTServlet
datadisplay
ui
ui
RegisterComponent
DisplayServlet
internal
TableObject
dataaggregate
dataaggregate
RegisterComponent
DataAggregate
IDataAggregate
internal
DataObject
dataaggregator1
IdentityDataAggregator
dataaggregator2
MergeDataAggregator
dataaggregator3
SumDataAggregator
dataaggregator4
IdentityCommentDataAggregator
dataaggregatordsl
DSLDataAggregator
dataaggregatorproperty
PropertyDataAggregator
dataaggregatorinterface
DataAggregator
IDataAggregator2
IDataAggregator
datasource1
RegisterComponent
DataSource1
datasource2
RegisterComponent
DataSource2
datasource3
RegisterComponent
DataSource3
dsl
impl
DSLImpl
dsl
DSLInterface
datasourceinterface
DataSourceServiceImpl
IDataSourceService
DefaultDataItem
IDataSource
IDataItem
property
impl
PropertyBagImpl
property
PropertyBag


datadisplay
rest
RegisterComponent
RESTServlet
ui
ui
RegisterComponent
DisplayServlet
internal
TableObject
dataaggregate
dataaggregate
RegisterComponent
DataAggregate
IDataAggregate
internal
DataObject
dataaggregator1
IdentityDataAggregator
dataaggregator2
MergeDataAggregator
dataaggregator3
SumDataAggregator
dataaggregator4
IdentityCommentDataAggregator
dataaggregatordsl
DSLDataAggregator
dataaggregatorproperty
PropertyDataAggregator
dataaggregatorinterface
DataAggregator
IDataAggregator2
IDataAggregator
datasource1
RegisterComponent
DataSource1
datasource2
RegisterComponent
DataSource2
datasource3
RegisterComponent
DataSource3
dsl
impl
DSLImpl
dsl
DSLInterface
datasourceinterface
DataSourceServiceImpl
IDataSourceService
DefaultDataItem
IDataSource
IDataItem
property
impl
PropertyBagImpl
property
PropertyBag

Summary

- Module and service concepts should be used to structure software
- Bundles expose clear interfaces
- Communications works by using services (minimum requirement between layers)
- Layers are testable separately
- Different assemblies should be possible