Architecture of the hybris Commerce Suite

- The hybris Multichannel Suite is a highly flexible and modular piece of software. The flexibility comes from the several layers of abstraction and functionality.

- It contains a hybris Platform consisting of core functionality and all other additional functionality built on it.

- To familiarizeyourself with the whole architecture, first of all you need to understand its basic concept and layer approach to architecture.

- Then you can dig deeper into the topic of ServiceLayer and how it works together with the Spring framework.

- Then you should learn about more technical topics like caching, filtering solution,
multitenancy, and possibly how to keep track of attributes changes.

Basic Architecture

From a business point of view, the hybris Commerce Suite is divided into individual packages, such as Commerce, Content, Channel, and Orders.

These packages are bundles of functionality assembled for a certain range of business functionality. All of these packages rely on more basic functionality provided by the hybris Platform.
While the hybris Platform can run without any package, no package can run without the hybris Platform.

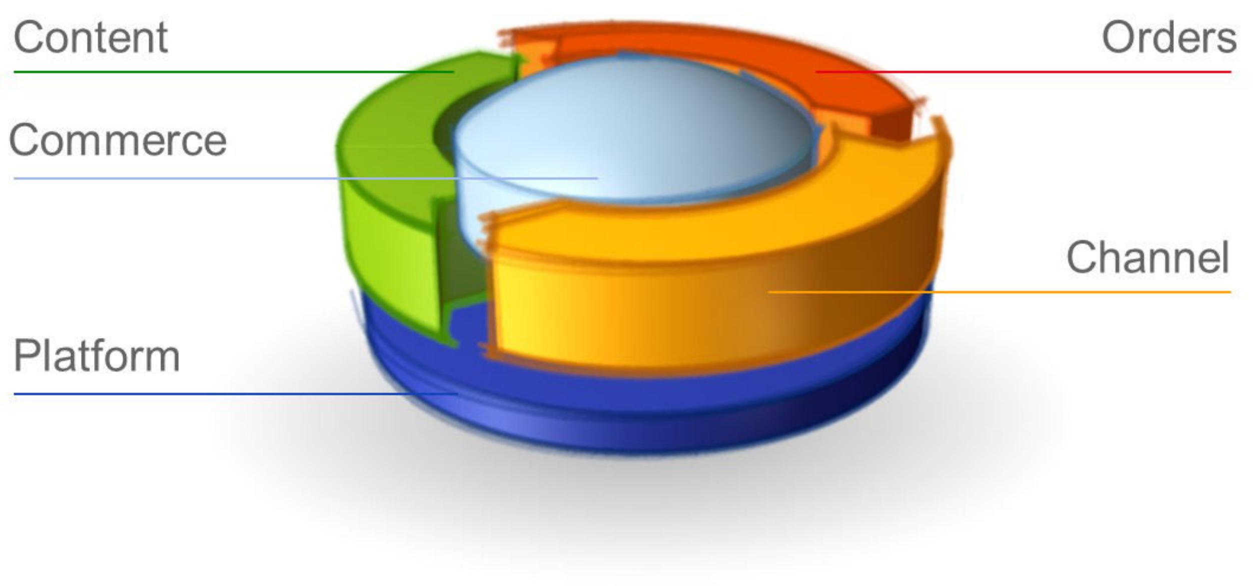


Figure: The hybris Commerce Suite diagram

- From a more technical point of view, packages consist of individual modules . For example, the hybris Print technically consists of two extensions: **Print** (the technical foundation) and **Print Cockpit** (the graphical user interface).
- Extensions are written by hybris or the implementation partner of your project. Extensions written by hybris provide standardized functionality and are supported and maintained by hybris.
- If you write an extension, you need to maintain them by yourself, but you are free to implement any business functionality you need. A full hybris Commerce Suite installation therefore consists of the hybris Platform plus any hybris packages plus any additional extensions that you have implemented.
- Extensions that are part of hybris Platform proper are also referred to as the core extensions. On top of these core extensions, hybris Platform contains several pieces of hybris software, such as the BuildFrameWork, and third-party software, such as the **pre-bundled Apache Tomcat.**
- Hybris Commerce Suite is run in a Java Virtual Machine on a Servlet Container or a J2EE-compliant application server (such as IBM Websphere or Oracle WebLogic) and connects to an external database (MySQL, Oracle DB, Microsoft SQL Server).
- Internal caching and persistence mechanisms allow the hybris Commerce Suite to run on a Servlet Container. A full-fledged J2EE-compliant application server can be used but is not necessary.

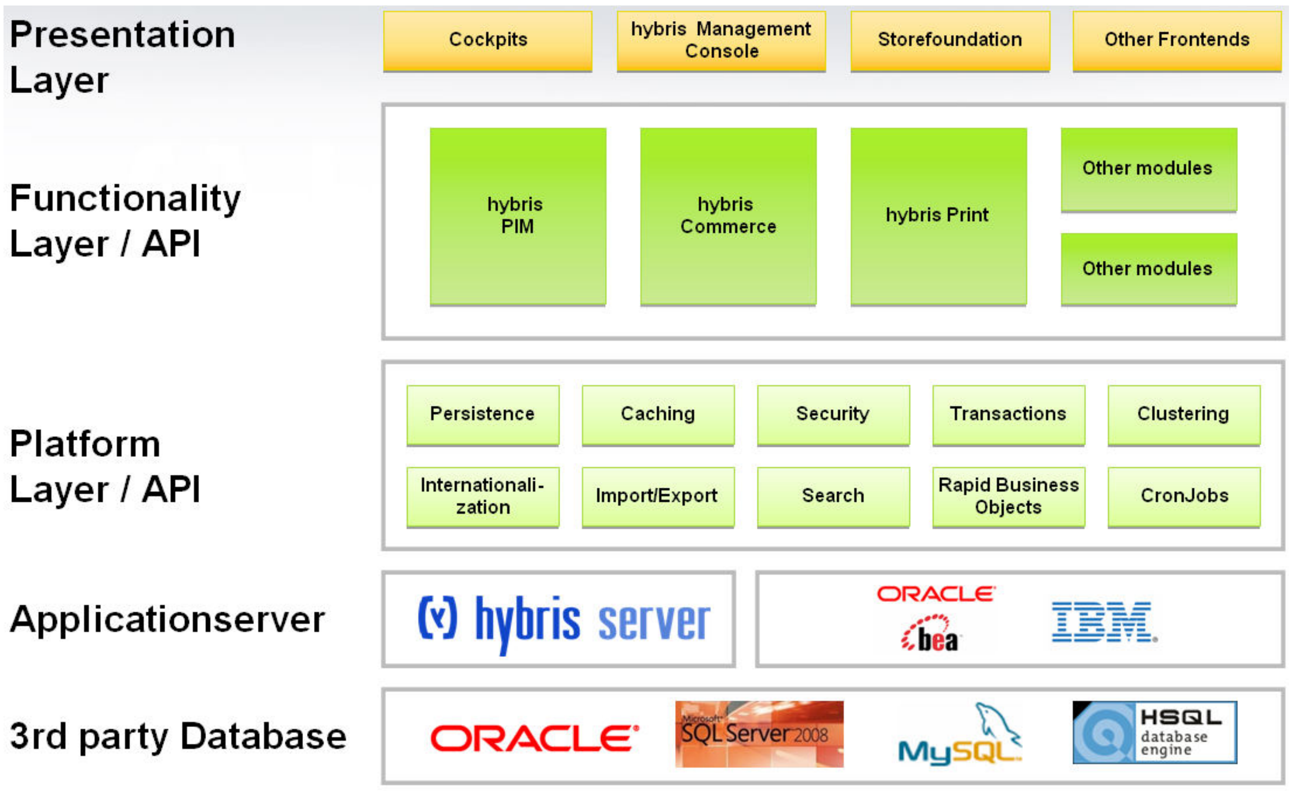


Figure: A technical view on the basic architecture of a hybris Commerce Suite installation.

- The hybris Platform layer abstracts data from the storage structure on the database using the persistence framework and provides functionality such as Clustering and the hybris Platform Cache.
- Relying on the persistence framework, the other functional components of the Platform Layer provide basic business functionality: **Transactions, CronJobs, Personalization, Internationalization, and more.**
- The packages on the Functional Layer (hybris Commerce, hybris PIM, hybris Print) use the hybris Platform to implement the functions they deliver. Actually, hybris Platform is part of any hybris Package.

Layer Architecture

The hybris Commerce Suite contains several layers, each of which has a different function and data abstraction level.

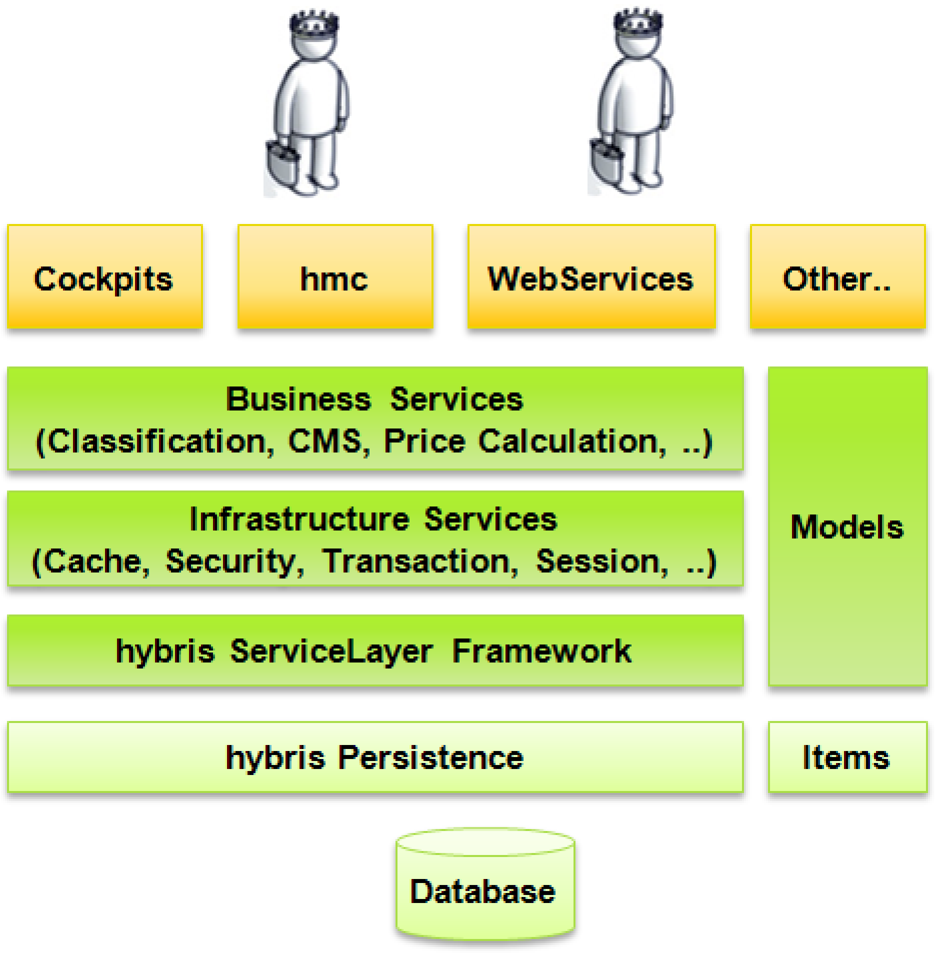

Cockpits
hmc
WebServices
Other..
Business Services
(Classification, CMS, Price Calculation, ..)
Infrastructure Services
(Cache, Security, Transaction, Session, ..)
hybris ServiceLayer Framework
Models
hybris Persistence
Items
Database

Figure: Overview of the architectural layers of the hybris Commerce Suite.

Cockpits, hmc, WebServices

This is where objects are represented in a way that an end-user can interact with them: add products to a cart, edit a product description, or set a password for a user account, for example. On this layer it is possible to let a user do something with an object in the hybris Platform via a graphical user interface.

Functionality on this level (such as the JSF-based hybris StoreFoundation or the ZK-based hybris Print Cockpit) uses the ServiceLayer for functionality and the Type Layer for storage of objects.

Depending on the complexity of your implementation, this layer can itself consist of several individual layers and even use an individual data model

ServiceLayer Framework
(including the actual ServiceLayer, the Infrastructure Services, and the Business Services)

Provides the Java Application Programmer's Interface (API) for objects in the hybris Commerce Suite, the hybris API. The hybris ServiceLayer relies on so-called models, which are POJO objects. Attributes on models have automatically generated getter and setter methods. Models are generated based on types.

Product look like on this Layer metioned bellow

ProductModel.java

```
public class ProductModel extends ItemModel
{
   /** <i>Generated constant</i> - Attribute key of
<code>Product.catalogVersion</code>
   * attribute defined at extension <code>catalog</code>. */
   private static final String CATALOGVERSION = "catalogVersion";

   /** <i>Generated constant</i> - Attribute key of <code>Product.code</code>
   * attribute defined at extension <code>core</code>. */
   private static final String CODE = "code";

    * <i>Generated method</i> - Getter of the
<code>Product.catalogVersion</code>
   * attribute defined at extension <code>catalog</core>.
    * @return the catalogVersion
    */
   public CatalogVersionModel getCatalogVersion()
   {
      if( !isAttributeLoaded(CATALOGVERSION))
      {
         this._catalogVersion = (CatalogVersionModel)
            loadAttribute(CATALOGVERSION);
      }
      return this._catalogVersion;
   }

   /**
    * <i>Generated method</i> - Getter of the <code>Product.code</code>
   * attribute defined at extension <code>core</core>.
    * @return the code
    */
   public String getCode()
   {https://wiki.hybris.com/display/release5/OCC+Architecture+Overview
```

```
        return _code;
    }
```

Database

Although not a layer of the hybris Commerce Suite, the database is also an important component in this overview: the database makes the data held in the hybris Commerce Suite persistent.
Database-specific representation of numbers and strings.

TypeLayer

Describes business object models. It is on this layer that definitions of business objects (types) and their fields (attributes) are made via the `items.xml` items.xml file. Models are generated based on types.
Product look like on this Layer metioned bellow

items.xml

```
<itemtype code="Product" extends="LocalizableItem">
   <attributes>
      <attribute qualifier="code" type="java.lang.String"/>
   </attributes>
</itemtype>
```

Persistence Layer

Deals with abstraction from the database, caching, and clustering. You are not likely to get into contact with the Persistence Layer at all as it is completely transparent and does not require any explicit interaction from your side.

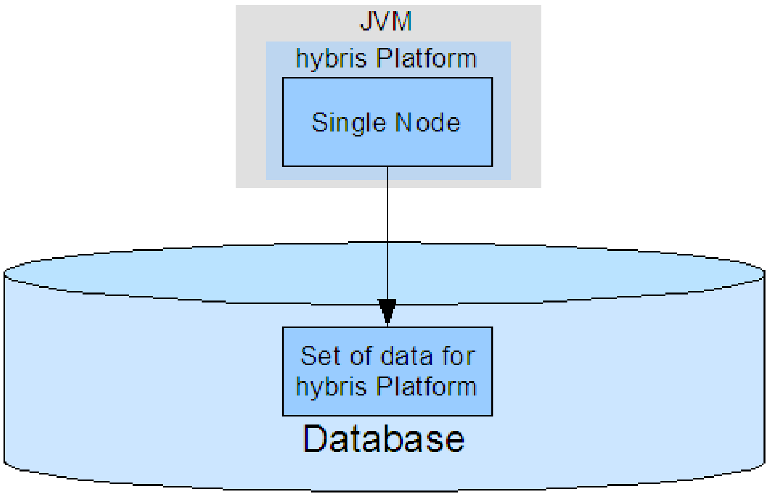
SQL-compliant representation of numbers and strings in a database table, such as VARCHAR, CLOB

Modes of Operation

You can run the hybris Commerce Suite in three different modes of operation:

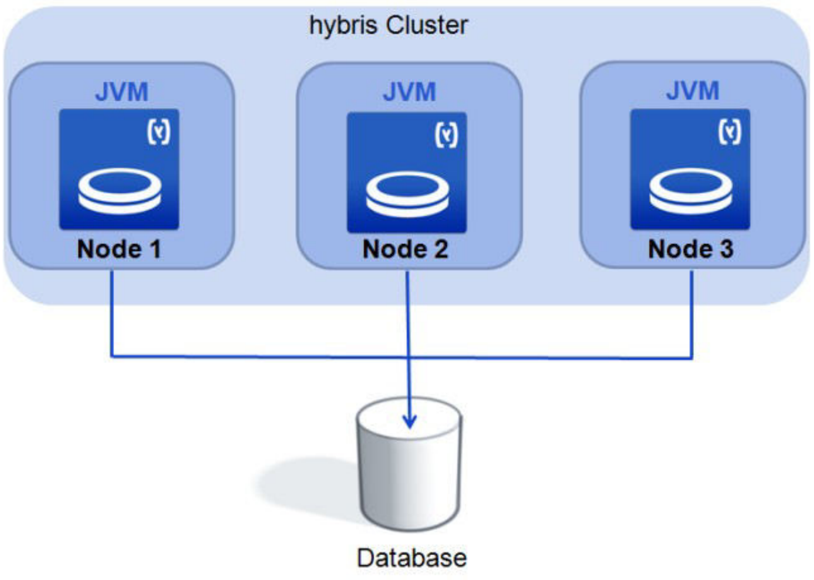
1) Single Node
2) Cluster Mode
3)Multi-Tenant Mode

1) Single Node:



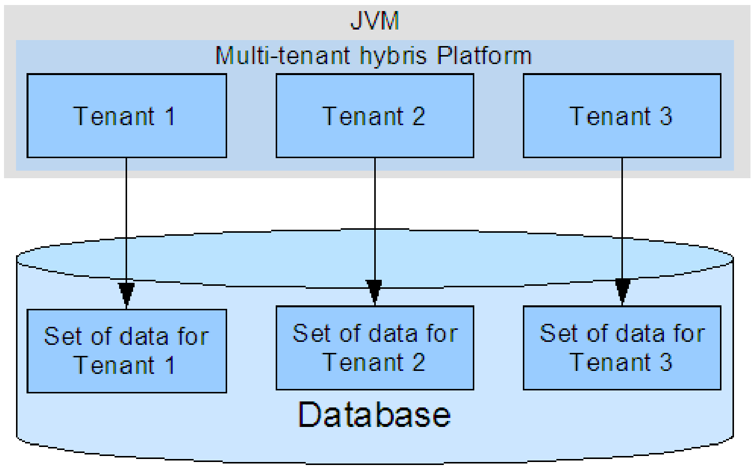
The most basic mode of operation. A single machine running one instance of a hybris Commerce Suite installation. Does not have several nodes (unlike in Cluster Mode) and only has one set of data (unlike Multi-Tenant mode).

2)Cluster Mode:



The hybris Cluster consists of several individual nodes. These nodes access a common database and communicate among each other via the TCP or UDP protocol. Summing up, this is a multi-node, cross-linked version of Single Node.

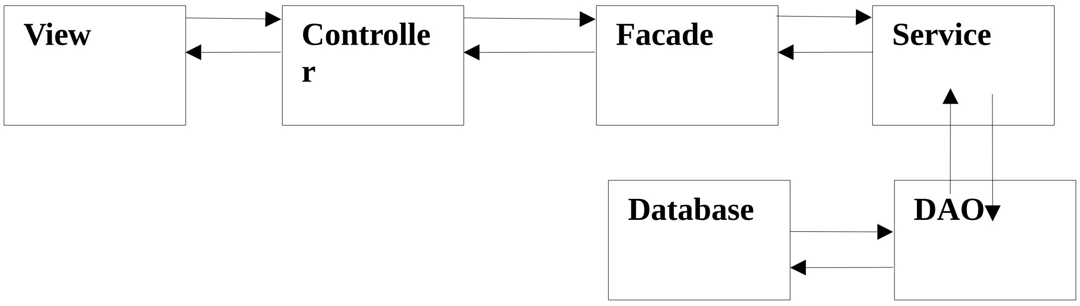
3)Multi-Tenant Mode



A hybris Commerce Suite in Multi-Tenant Mode allows using several individual, distinguished sets of data separated by database table prefixes. Multi-Tenant Mode can be used for Single Node and for Cluster Mode operation.

Hybris Followed Design Pattern

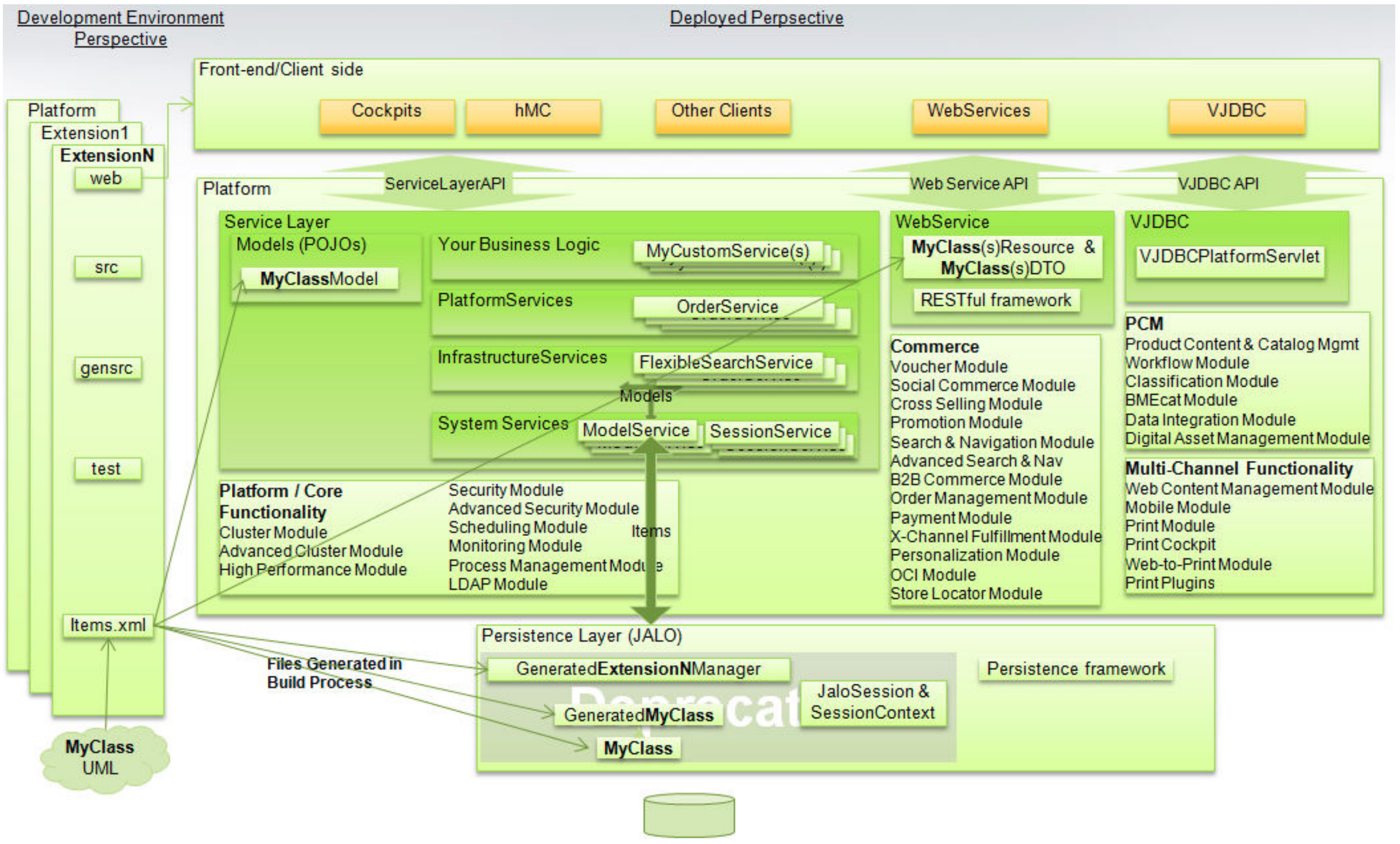
Hybris is Fallows **Façade Layer Design** pattern



Architectural Overview for Tutorial Trail

Note:-Before starting the tutorial trail, you should have a broad understanding of the hybris Suite's architectural framework..

The diagram below illustrates key concepts/features of the hybris Architecture that will be covered in this tutorial trail. Key points to take away from this diagram include:



- From a development perspective, the hybris Suite comprises of a platform holding the central hybris logic, multple extensions and a persistence layer.
    - the platform holds the central functionality of the hybris Suite: security, caching, jobs, JMX etc.

    - the extensions written both by hybris and by partners contain functionality that builds upon the hybris Platform and focuses on business use cases and user stories
    - the persistence layer contains low-level persistence classes. While we used to encourage partners to implement business logic in in the persistence classes, we no longer do so. Partners should implement their business logic in the services in the ServiceLayer.
- There are 3 main APIs that clients can use: the Service Layer API, Web Service API and VJDBC API.
    - the Service layer follows a service oriented architecture, consisting of numerous services to which partners can add their own: see ServiceLayer
        - Partners should code as much as possible in the service layer - creating new models and services as required.
        - the services liaise with each other and with clients using Models,
        - there are about 30 default services available ranging from high level plafom services, to infrastructure and system services.
        - the ModelService service converts from items (used in the persistence layer) to Models (used in the service layer).
        - the Session Service is used to maintain Session-scoped data
    - the web service API is a RESTful webservice framework: see Web Services in the hybris Multichannel Suite
        - the plumbing code is automatically generated such that all items specified in all extensions are automatically available via REST
    - VJDBC offers secure remote access to the database: see VirtualJDBC Extension
        - using both plain SQL and hybris' FlexibleSearch
        - as communication takes place over HTTP(S), there are no firewall issues
- to introduce a new Data Model and associated logic, partners should..
    - create their own new extension
    - specify their data model in the new extension's items.xml file.
        - the hybris build process generates several files to support the back-end persistence of, and webservice access to the new items specified in the data model - see diagram below. In the diagram we assume **MyClass** is a new Data Model defined in items.xml, from which is generated a number of files:

            - **MyClassModel.java -** a representation of the new Data Model in the service layer
            - **MyClass(s)Resource.java** and **MyClass(s)DTO.java** - support for CRUD via the hybris RESTful webservice
            - And two low-level persistence-tier classes which you should not have to touch:
                - **GeneratedMyClass.java -** low level persistence class for the model , which is regenerated on each build (so do not place any custom logic here)
                - **MyClass.jav**a - inherits from GeneratedMyClass.java and is not regenerated if it already exists, so you can place custome logic here
        - write business logic in new services in the service layer (TODO: and by overriding/replacing existing ones?)
- in this trail we suggest that partners use TDD from the front-layers to the back to iteratively build well tested logic that best fits their required use cases.