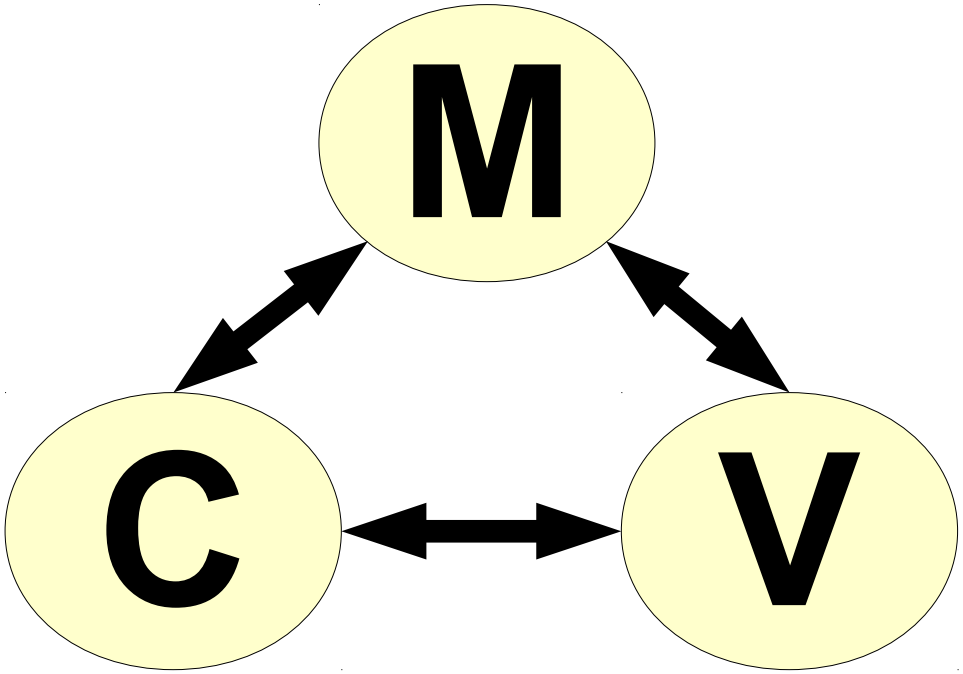
MVC in XPages
Model – View – Controller



A presentation for DanNotes

by John Dalsgaard

Notesnet.dk

DALSGAARD DATA

Me

- Worked with Notes since 1995 version 4.5
- Java since Notes 5.0.7 (awfull Java implementation – version 1.1!!!)
- Large web-apps. (40.000+ users)
- Object Oriented approach since 1999 (yes, in LotusScript...)
- Now: XPages & mobile apps....
- Certified Principal/advanced administrator and developer – all versions 4.6 → 8.5

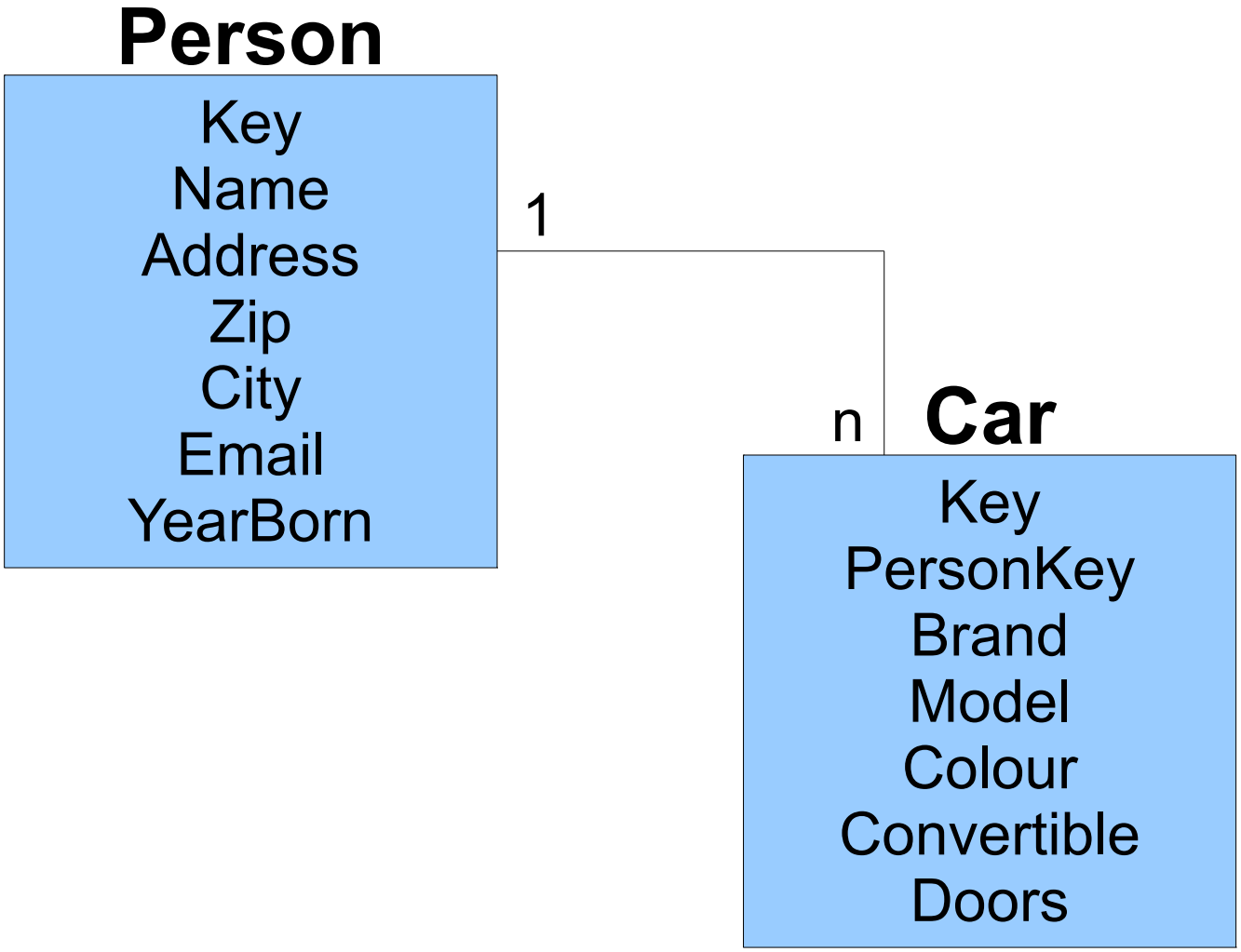
Agenda

- Demo: What is this all about?
- What is MVC
  - Data model: Data bean & DAO
  - Service layer: CRUD/Facade
  - View layer: View bean
- Why would you want to use MVC
- What to do in Domino
  - Data structure
  - Challenges...
- A word about the Demo app.

Demo

- Simple example
- Isolate data access (DAO + interface)
- Validation in model (service layer)
- Binding to Java beans through Expression Language
- Separate Model from View → Easy to add new ”view” component
  - XPage
  - JSON WebService

Demo – Data model




DEMO TIME...

Agenda

- Demo: What is this all about?
- What is MVC
  - Data model: Data bean & DAO
  - Service layer: CRUD/Facade
  - View layer: View bean
- Why would you want to use MVC
- What to do in Domino
  - Data structure
  - Challenges...
- A word about the Demo app.

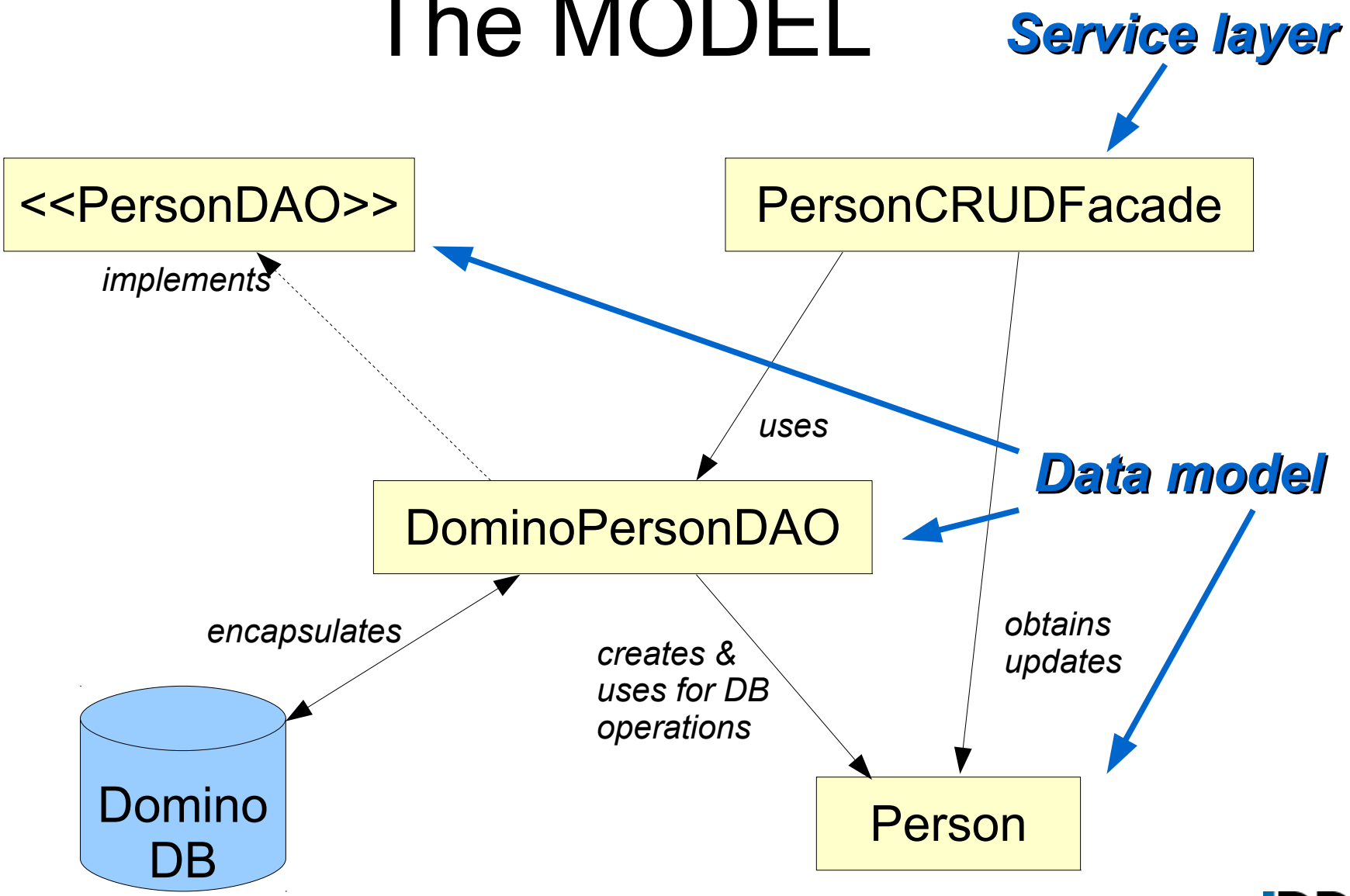
What is MVC

- A ”design pattern”
  - Very good practice for the task at hand
- Separation of the tasks:
  - **Model**: Business rules, data model and data manipulation
  - **Viewer**: Presentation of data
  - **Controller**: Flow in application

MVC - Model

- Java objects that implement:
  - Data model
    - Contain data
    - Manipulate data from the database
      - Specific database implementation
      - CRUD (Create, Read, Update, Delete)
  - Service layer
    - Implement business rules
      - Validation
      - Combine data (e.g. number of years employed based on hire date)

The MODEL


Service layer
<<PersonDAO>>
PersonCRUDFacade
implements
uses
Data model
DominoPersonDAO
encapsulates
creates &
uses for DB
operations
obtains
updates
Domino
DB
Person

Data model

- Is the only layer that knows the physical details of data access (documents, views, etc.)
- Consists of:
  - Data bean
    - all fields – with getter/setters – no logic
  - DAO (Data Access Objects)
    - Interface used by service layer
      - Defines all methods/operations
    - Implementation of interface
      - All handling of underlying database (e.g. recycle...)
      - Implements necessary base functions for CRUD

Service layer

- This is where the business logic of the application lives
  - Workflows
  - Validations
  - Searches
  - Etc.
- Connects to the DAO implementation through its interface
  - Not relying on the implementation and the underlying database ”dialect” (e.g. Domino)

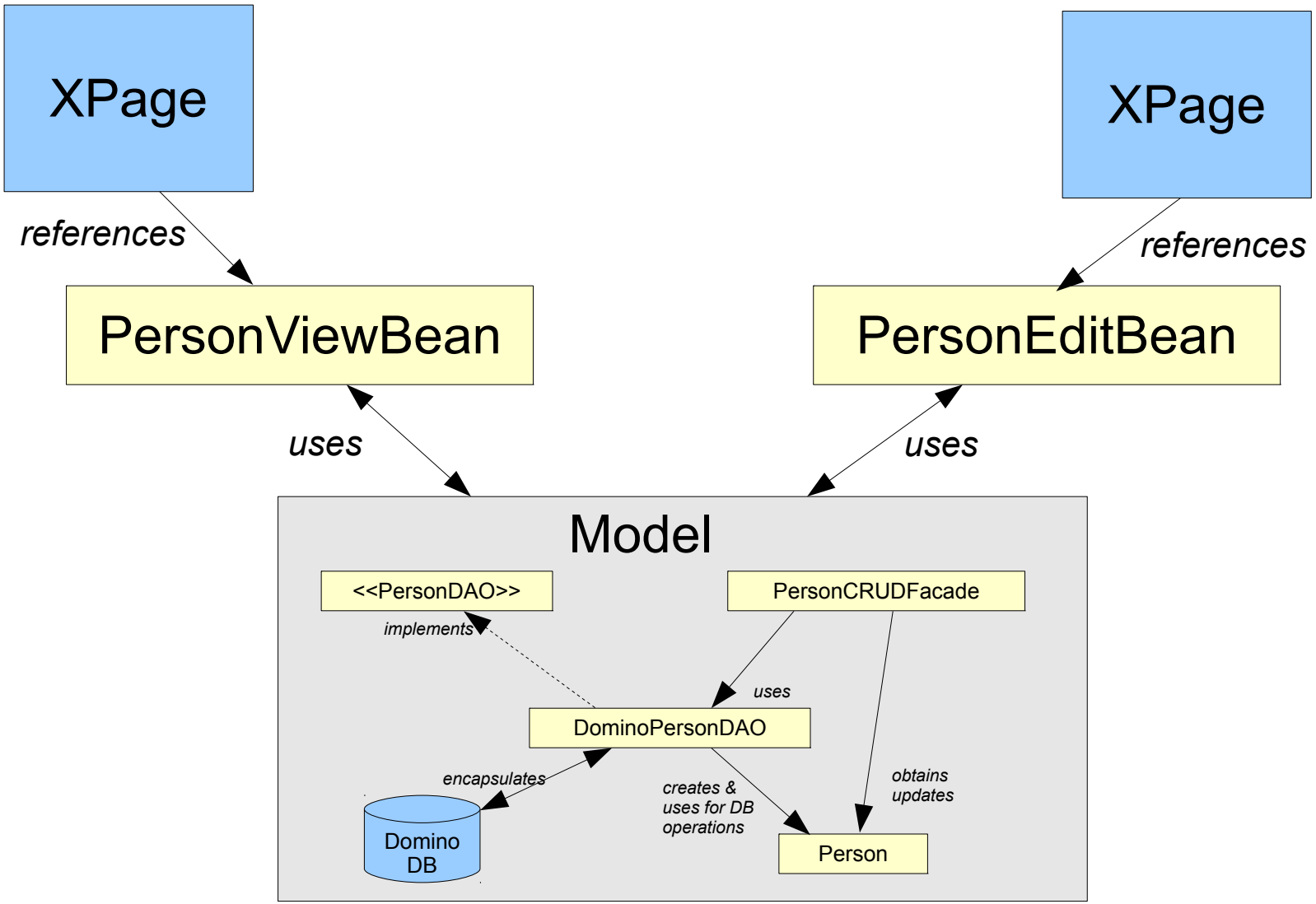
Service layer

- Do **not** duplicate Domino ”techniques” from the old days
  - Eg. Invoice with invoice lines
  - We have seen the pattern: line_1, line_2, line_3 in traditional Notes applicatctions
    - Fixed number of ”relations”
  - Create two data objects and (DAO's etc.)
    - Invoice
    - InvoiceLine
  - Hide implementation details from the service layer

MVC - View

- Java objects for presentation that
  - Are used by the XPage
  - Often one-to-one between XPages and view Beans
  - XPage talk to the view Bean – and never to the underlying database (it doesn't know it exists!)
  - Typically defined to the XPages via faces-config.xml

The VIEW


XPage
XPage
references
references
PersonViewBean
PersonEditBean
uses
uses
Model
<<PersonDAO>>
PersonCRUDFacade
implements
uses
DominoPersonDAO
encapsulates
creates &
uses for DB
operations
obtains
updates
Domino
DB
Person

View Layer

- View Bean
  - Implements all the methods that the ”client” sees
  - Talks to the service layer
    - E.g. through a saveUser() method that does validation and throws a validation error to the bean if data is not valid
  - Handles UI ”things”, e.g. showing validation errors reported by the service layer
    - E.g. by connecting the validation errors from the service layer to the corresponding components on the XPage

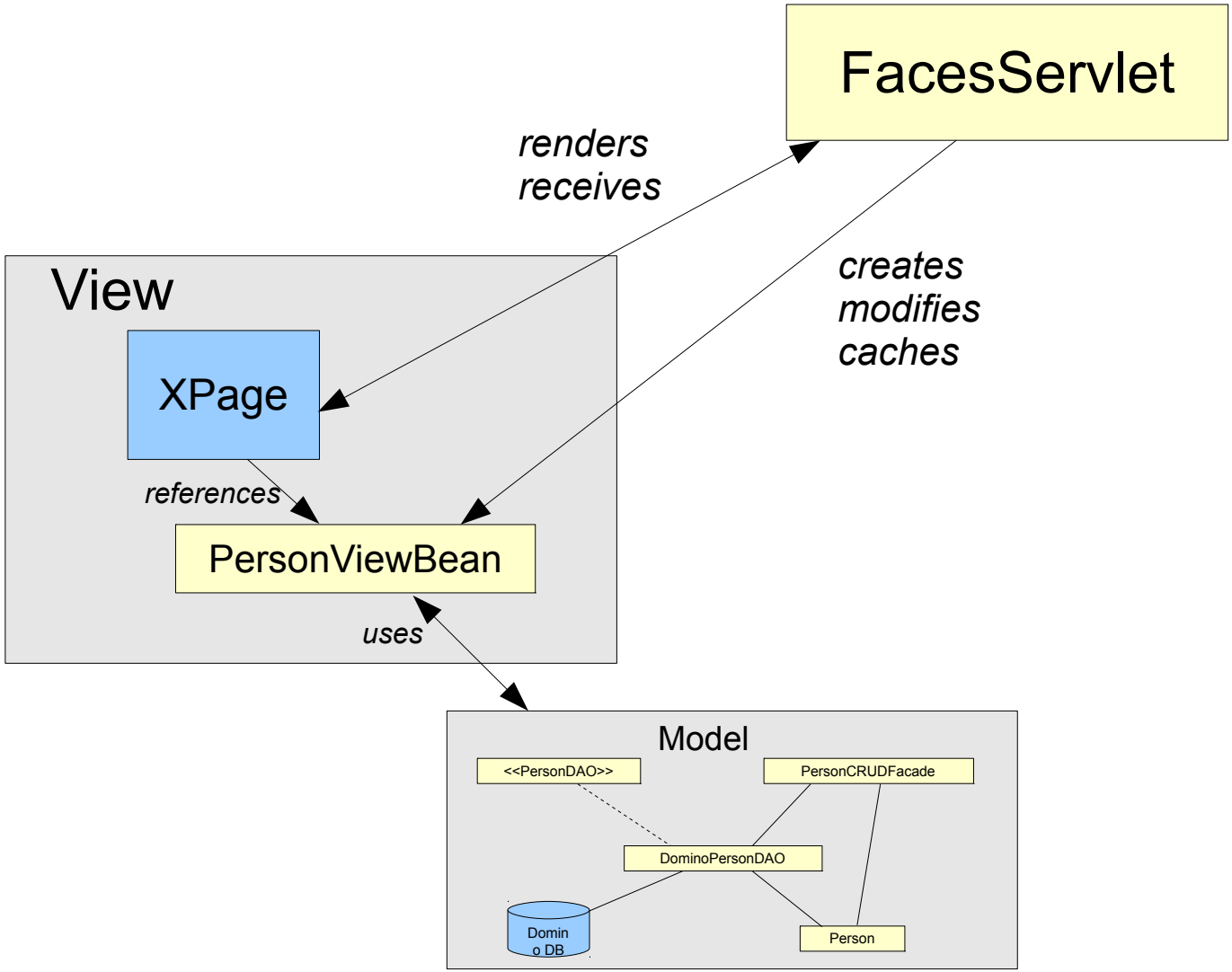
View Layer

- XPage
  - Access the View Bean through the scopes
    - Define the bean in faces-config.xml
    - … or create instance on load of page
  - Access properties and methods through
    - EL: e.g. UserBean.user.name
    - SSJS: e.g. UserBean.user.getName()
    - Call a ”save” method on submit, e.g. a button with SSJS: UserBean.user.saveUser()

MV**C** - Controller

- This is the FacesServlet in XPages
  - You do **NOT** code the controller...!!!
  - It handles:
    - Flow of incoming requests
    - Data streams
    - Interacts with the view layer for us
- If you **do** want to change behaviour in the controller?
  - You need to understand the JSF LifeCycle
  - Use a PhaseListener

The CONTROLLER



Agenda

- Demo: What is this all about?
- What is MVC
  - Data model: Data bean & DAO
  - Service layer: CRUD/Facade
  - View layer: View bean
- Why would you want to use MVC
- What to do in Domino
  - Data structure
  - Challenges...
- A word about the Demo app.

Why use MVC?

- Avoid data-duplication (in responses)...!!!
- Avoid successive lookups
  - Use Java objects in memory
- Enable several ”clients” using same model, e.g.
  - Browsers (XPages)
  - Mobile browsers (jQuery Mobile)
  - Mobile apps (JSON webservices)
- Independent of database architecture
  - Possible to change to another DBMS
  - Enable isolated optimizations (e.g. change to use org.openntf.domino API)

Why use MVC?

- Unit test...
  - The structure allows us to test our code at all ”Model” levels:
    - Data bean
    - DAO implementation
    - CRUD/Facade bean
  - This allows easy re-test of all code after we have modified it

    → BETTER code...!!!!

Agenda

- Demo: What is this all about?
- What is MVC
  - Data model: Data bean & DAO
  - Service layer: CRUD/Facade
  - View layer: View bean
- Why would you want to use MVC
- What to do in Domino
  - Data structure
  - Challenges...
- A word about the Demo app.

MVC in Domino

- First important fact:
  - An NSF is **NOT** a relational database
    - However, we can emulate that in the service layer and treat it as such
    - In the DAO we can use some of the characteristics of the NSF (e.g. response documents) – especially in relation to existing applications
    - You should create Java objects to emulate the parent/child relations no matter what the underlying ”technique” is. Let the DAO handle the implementation...

MVC in Domino

- Data duplication (from parent to child doc.) can be avoided...!!
  - ...by using your parent/child structure in the Java objects
- You will (most likely) not use any datasources in your XPages – but refer to beans instead

MVC in Domino

- Validation and error messages
  - Choose to disable client side validation
    - Will show error messages next to the fields
  - Let the service layer bean validate data on submission
    - Capture all error messages with a field reference (e.g. in a HashMap)
    - Throw a validation error if not valid
  - If view bean gets a validation error on save...
    - Read all errors (through a getter)
    - Iterate and ”bind” errors to all fields (through a FacesMessage)

MVC in Domino

- Security
  - Readers/authors fields
    - Very strong feature (compared to RDBMS's)
    - Should **reflect** and **enforce** chosen security model – **not** be used for data retrieval etc. – due to performance impact
      - But this is not really different from what you should have been doing always...
  - These mechanisms should be implemented in the DAO
    - Also includes other system fields like $PublicAccess etc.

MVC in Domino

- Beware:
  - Using Scoped beans you may need to ”Clean” your project for changes to be effective....
  - Use ”LogReader” (by Majkilde – download from openntf.org) to see messages in error-log-0.xml (!!)
  - Sometimes ”System.out.println” ***can*** be your friend in locating errors... ;-)

Agenda

- Demo: What is this all about?
- What is MVC
  - Data model: Data bean & DAO
  - Service layer: CRUD/Facade
  - View layer: View bean
- Why would you want to use MVC
- What to do in Domino
  - Data structure
  - Challenges...
- A word about the Demo app.

Demo database

- Demo database available:
  - Bitbucket: https://bitbucket.org/john_dalsgaard/demo-apps.git
- To set up:
  - Clone this project to your local source control
  - Open the cloned project as a new Java Project in Domino Designer
  - From the Package Explorer view
    - Select Team Development and then Associate with a new NSF...
  - Create the database on a server (9.0+) and you are ready to explore the database

Demo database - EXTRAS

- Caching....
  - In the service layer we have implemented caching of ALL dataobjects in memory
    - Runs FAST
    - Need to remember to update memory object when updating the DB – but it is not difficult!
- Neat way of entering several rows
  - … and handling cursor position (!!)
- Debug info written to console in lots of places
  - Phase listener shows in what phase these messages are generated

Demo database - EXTRAS

- Uses org.openntf.domino API
  - You need to install the org.openntf.domino.jar in your jvm/lib/ext directory.
    - Download from openntf.org
- Validation:
  - Implemented in the service layer. Note method to place errors against right fields using FacesMessage

Questions?

?

- **…. or contact one of ”Gang of Four”:**
  - Jakob Majkilde: majkilde.dk
  - Per Henrik Lausten: phl-consult.dk
  - John Dalsgaard: www.dalsgaard-data.dk
  - John Foldager: izone.dk

Contact

You are always welcome to contact me:

John Dalsgaard
Dalsgaard Data A/S
Solbjergvej 42
Solbjerg
DK-4270 Høng

Phone: +45 4914-1271
Email: john@dalsgaard-data.dk
www.dalsgaard-data.eu

Sources

- Great articles from Pipalia:
  - Using JSF Framework Development Standards for your XPages Project
    - Article 1 - The concept and motivation...
    - Article 2 - Designing for MVC
    - Article 3 - The DAO
    - Article 4 - The CRUD (Facade)
    - Article 5 - The Bean...
    - Demo database

Sources

- Validators....
  - Pipalia: Using beans as validators
  - Stack overflow: Answer by Sven Hasselbach
  - Stack overflow: getComponent()
- XPages & DB2
  - Example of alternate DAO using DB2