Polish Japanese Institute of Information Technology

Design and Analysis of Information Systems (MAS)

Mariusz Trzaska, Ph. D.
mtrzaska@pjwstk.edu.pl

Lecture 01

Designing and modelling of the System's Architecture

http://www.mtrzaska.com

General Information

- Topics during the semester.
- Graduation
  - Classes:
    - 4 x mini-projects;
    - 2 x class tests;
    - Project (documentation and implementation).
  - Exam.

General Information (2)

- Literature

Mariusz Trzaska:
Modelowanie i implementacja systemów informatycznych. Rok 2008. Wydawnictwo PJWSTK. Stron 299. ISBN 978-83-89244-71-3.
http://www.mtrzaska.com/mas-ksiazka

Also as an eBook: PDF and ePub (at Virtualo.pl) (Polish only).

- Books about modelling, UML, object-orientation.
- Books about programming.

Outline

- Introduction
- Problems and solutions
- Architecture - Views
- Different kinds of architectures
  - *Mainframe,*
  - *Thin client,*
  - *Two-layers,*
  - *Three-layers*
- Three-layers architecture
- Model – View – Controller
- Summary

These slides make use of the following sources:

• Lectures *Software Architecture* by *Chandika Mendis (University of Moratuwa)*

• The paper *„Java SE Application Design With MVC”* by *Robert Eckstein*

Architecture

- The key allowing making complicated systems possible to build.
- Effective management of developing complicated systems
- Easier management of big employee groups
- Better reuse
- Improved integrity and integration inside of the system
- Implementation of business strategies

An Architecture – why?

You cannot use the same ad hoc approach to build a skyscraper that we use to build dog-houses.

Booch, SD'99

An Architecture – possible problems

- Decomposition of the system
  - How to divide the system into smaller pieces?
  - Do all items have been considered?
  - Do all items fit to each other?
- Control and management
  - Parallelization problems
  - Properties connected with many aspects
- We need to assure
  - Reuse
  - Reducing costs
  - Short time to market
  - Integrity of the system
  - Expandability
  - Maintenance
  - Conformity to our business goals

An architecture - definitions

- Architecture is the structure of the system, comprised of:
  - components or building blocks the externally visible properties of those components, and the relationships among them
  - Architecture is the structure of the system, comprised of.

Based on Bass, Clements, and Kazman. Software Architecture in Practice, Addison-Wesley 1997

System Decomposition

- Addresses the issue of Complexity
  - Intellectual Intractability
  - Management Intractability
- How do we break the system up into pieces?
  - Encapsulation
  - Loose coupling
  - Clean interfaces
- Do we have all the necessary pieces?
  - Functional requirements
  - Non functional requirements
- Do the pieces fit together?
  - Well defined interfaces
  - Stringent guidelines

Architecture Decisions - scope

- Architecture is the set of decisions that cannot be delegated without compromising the overall system objectives.
- Architecture decisions need to be made from a broad-scoped or system perspective.
- Any decision that could be made from a more narrowly-scoped perspective should be deferred to the team responsible (minimalist architecture)
- Architectural decisions should focus on high impact, high priority areas that are in strong alignment with the business strategy.
  - Strategic objectives
  - Important services
  - Systemic qualities and properties

Architecture - Views

- Conceptual
  - A diagram,
  - Identification of components and allocation of the responsibilities
- Logical
  - Updated conceptual diagram
    - With interfaces
    - Specifying components
  - Design of components, mechanism and connection protocols
- Execution
  - Processes (interaction diagrams)
  - Allocation running instances to processes,
  - Cooperation
  - Resource utilization

The Conceptual View

- Identifies high-level components taking into account a proper decomposition.
  - Dependencies,
  - Cooperation,
  - Responsibilities,

The Logical View

- Defines interfaces and specifies components:
  - Precisely,
  - Straightforward
- Describes different ways of:
  - Interaction,
  - Collaboration.
- Should provide the blueprint for the component developers to work relatively independently.
- Constitutes the detailed architecture diagrams.

The Execution View

- Takes into account:
  - Physical (implementation) model
  - Implementation
  - Reflection of the processes running in the real system
  - Scalability

Mainframe (legacy systems)

- This is not a modern client/server architecture, but there are some similarities
- A user connects to a mainframe using a simple terminal. The terminal intercepts key presses and sends them to the mainframe.
- The whole processing takes place in the mainframe.
- Is able to utilize different software/hardware platforms.

File-sharing Architectures

- This is not a modern client/server architecture
- The files are downloaded from a server to the workstation (desktop).
- Then, all processing (including business logic and data) took place on the workstation.
- Such an approach works quite well only if the amount of data is quite low.

Client - Server

- Approach introduced a database server to replace the file server.
- Using a relational database management system (DBMS), user queries could be answered directly.
- The client/server architecture reduced network traffic by providing a query response rather than total file transfer.
- It improves multi-user updating through a GUI front end to a shared database.
- In client/server architectures, Remote Procedure Calls (RPCs) or standard query language (SQL) statements are typically used to communicate between the client and server
- One of the most popular architectures.

A Thin Client Architecture

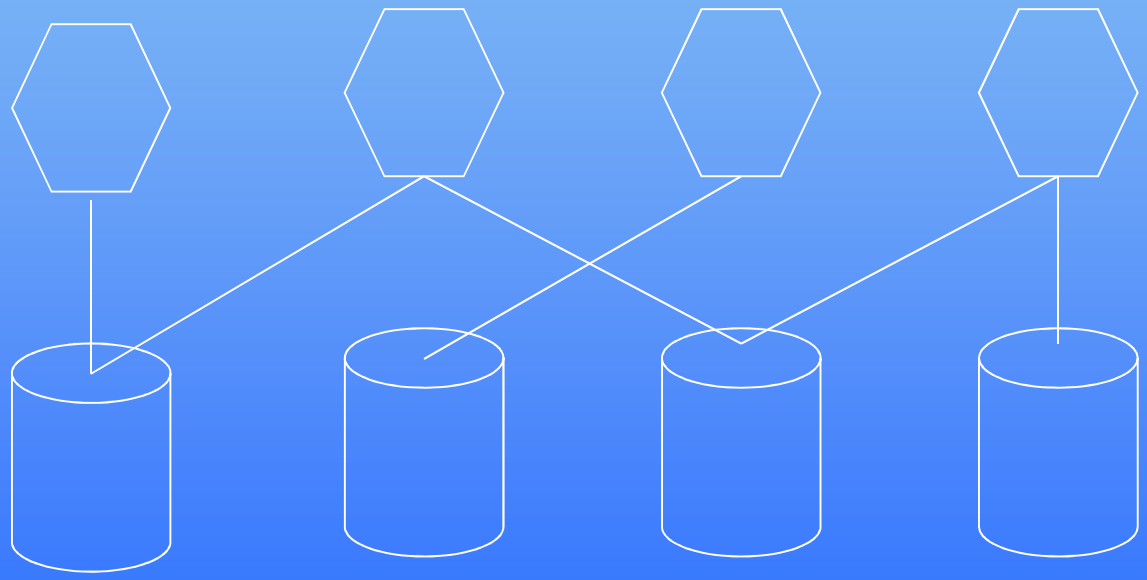
- Similar to the client - server.
- The client is a low-specification machine.
- **Entire** processing took place on the server
- Not so popular – why?

The Two-Tier Architecture

- Contains three components distributed in two layers:
  - Client (GUI),
  - Server (data, business logic)

GUI + some functions

DB Management + some functions (the bigger part)



- GUI is on the user's computer and DB functionality on the server (usually a high-spec computer).

The Two-Tier Architecture (2)

- Processing management is split between the user system interface environment and the database management server environment.
- The database management server provides:
  - Data,
  - stored procedures,
  - triggers etc.
- Implementation of processing management services using vendor proprietary database procedures restricts flexibility and choice of DBMS for applications
- limited flexibility in moving (repartitioning) program functionality from one server to another.

The Three Tier Architecture

- The most important change comparing to the two tier architecture:
  - Added a middleware between a GUI and DB
- Many ways of implementations and tasks for the middleware:
  - Transactions monitoring,
  - Exchanging messages between the other two layers,
  - Application server (business logic)
- Additional possibilities
- Queuing,
  - Running applications,
  - Special messages.

The Three Tier Architecture (2)

Transactions Monitoring

- Ability to updating many different DBMS during one logical transaction
- Transparent access to different data sources:
  - Files,
  - Relational DBMS,
  - Object DBMS,
- Different priorities for transactions
- Security

The Three Tier Architecture (3)

Messages Exchanging

- Asynchronous processing
  - Benefits,
  - Problems
- Synchronous processing
- Transparent communication between heterogenic applications
- „Intelligent” traffic management:
  - Priorities,
  - Queueing,
  - Automatic events notifications
- Separation from communication problems

An Application Server

- An application is run on a shared server rather then a desktop client
- GUI processing is performed out of the server
- Services:
  - Business logic,
  - Data access
- Improved security (a user does not have direct access to the application)

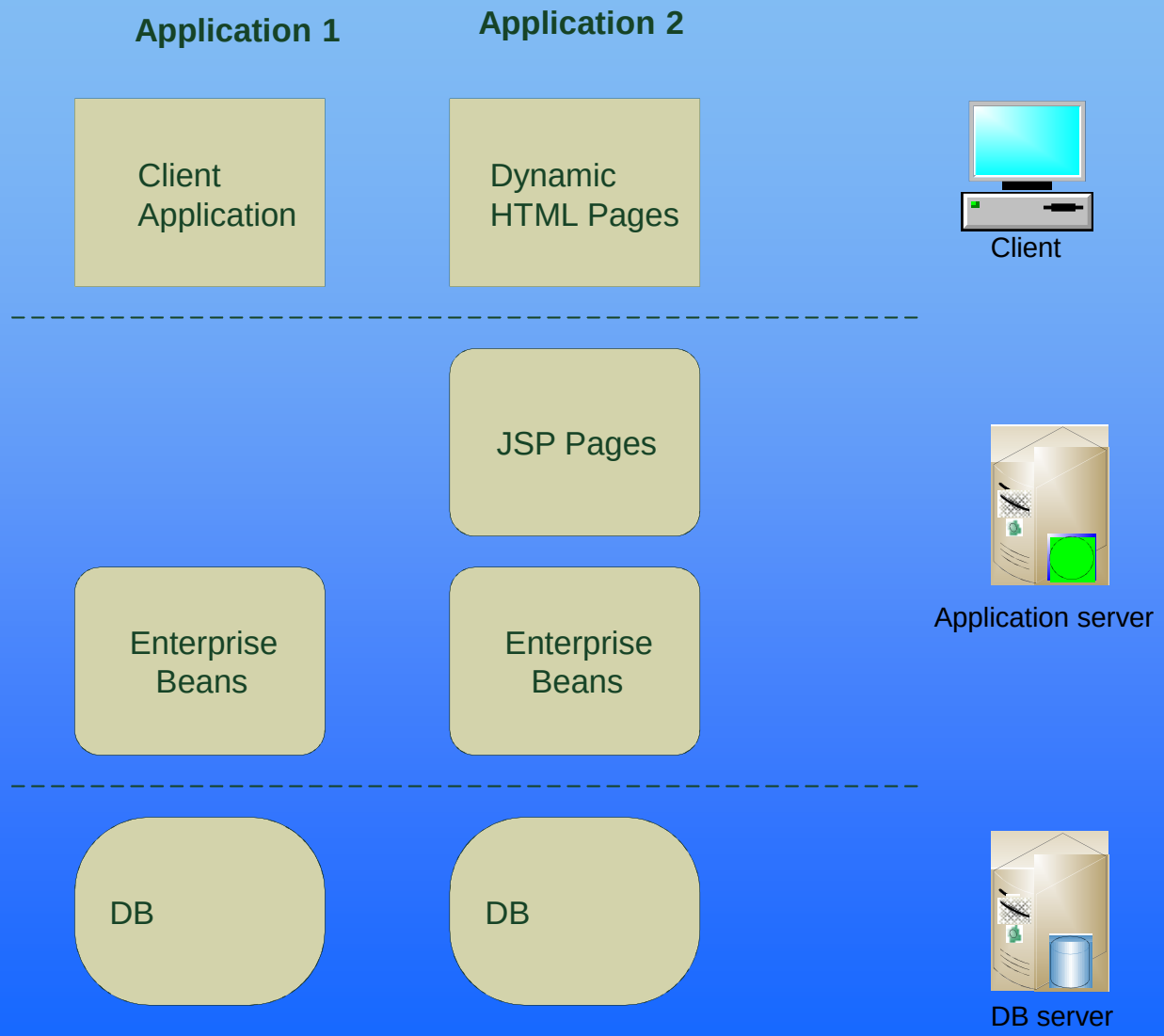
An Application Server (2)

- Better scalability
- Lower costs of maintenance on one server then many user computers.
- Examples:
  - Apache Tomcat
  - JBoss
  - Application Server J2EE
  - Plone,
  - Zope
  - Websphere Application Server

An Application Server (3)

- Common components:
  - Security
  - DB connection
  - Resources access
  - Interfaces to external components
  - Transaction management
  - Events controlled components (or listening to events)

Distributed Multi Tier Application

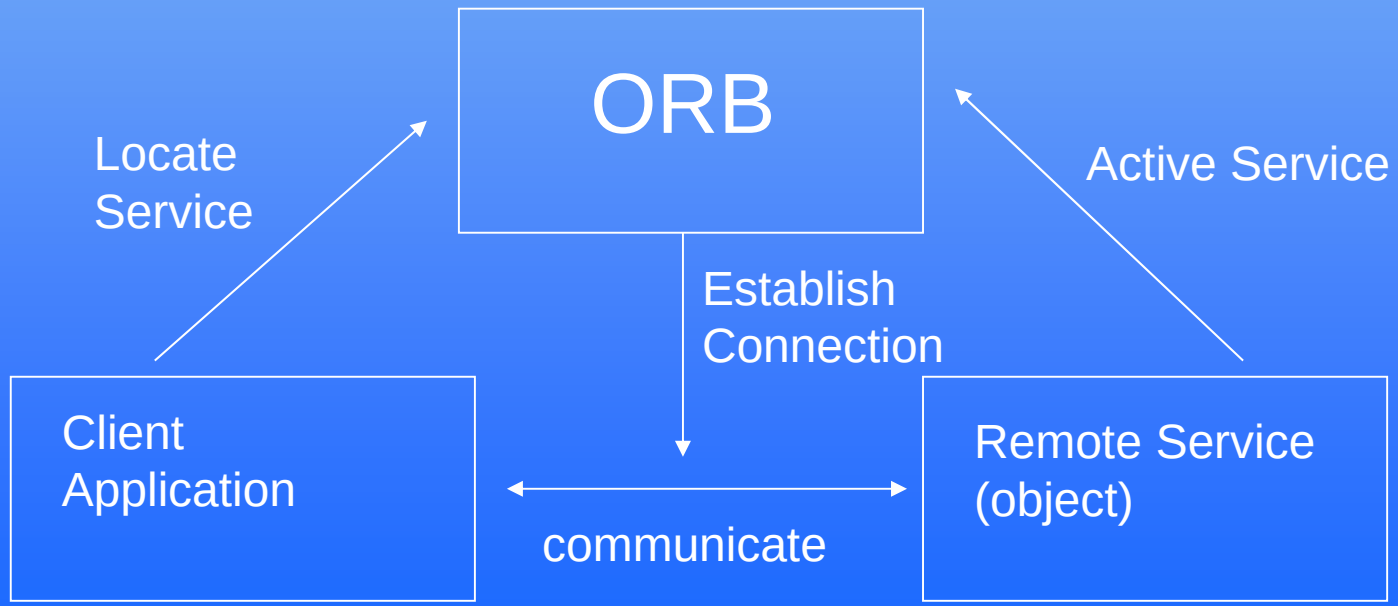


Three tier with an ORB architecture (*Object Request Broker*)

- Client/Server system using technologies that support distributed objects: CORBA, RMI, DCOM, etc
- Promote interoperability of distributed object systems because they enable users to build systems by piecing together objects- from different vendors- that communicate with each other via the ORB
- The implementation details of the ORB are generally not important to developers building distributed systems. The developers are only concerned with the interface details.
- This form of information hiding enhances system maintainability since the object communication details are hidden from the developers and isolated in the ORB
- Despites many advantages they are not so popular (?)

ORB

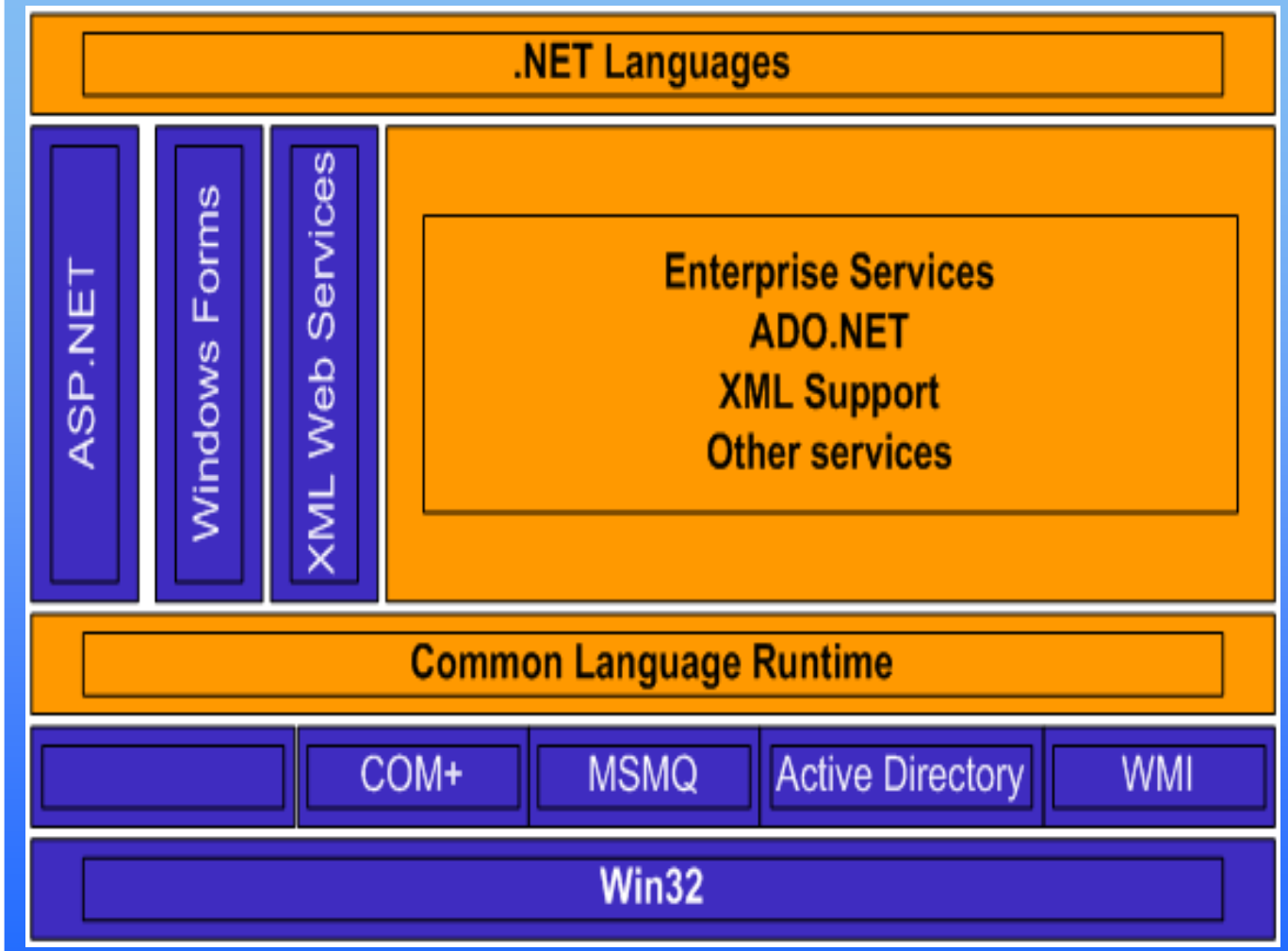
- The most important functionalities (services) of the ORB technology:
  - Interface definition,
  - Location and possible activation of remote objects
  - Communication between clients and object



ORB (2)

- Other functionalities:
  - Security,
  - Transactions,
  - Searching,
  - …
- Many different ways of implementing the ORB concept:
  - Client implementation,
  - Separated processes,
  - Part of the OS or Application Server

.NET Framework - Windows 2003 Server as an Application Server



- Enterprise Services
- Connection Pooling
- Distributed Transactions
- Security
- Messaging (MSMQ)
- Management/ Monitoring

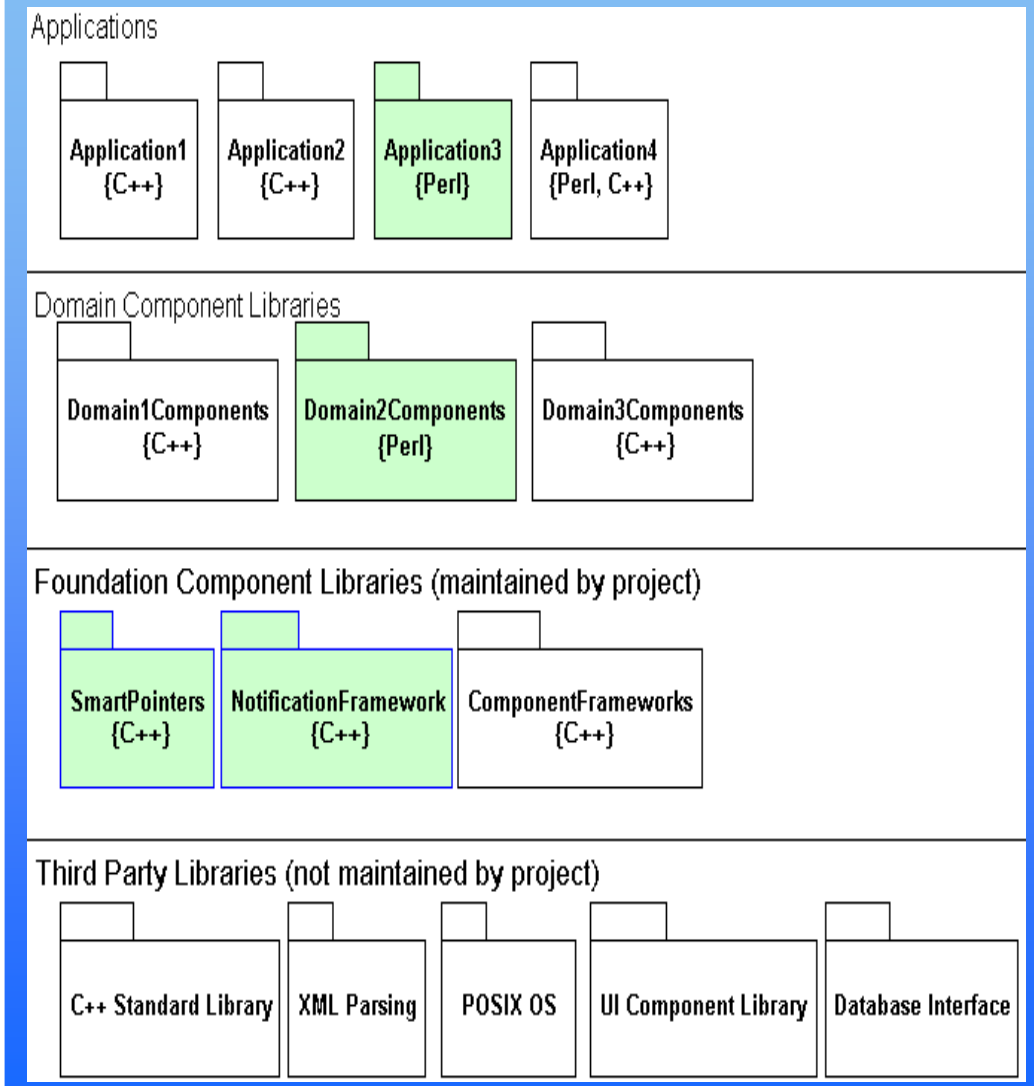
Źródło: MS Windows 2003 Technical Overview of Application Services, MS Corporation

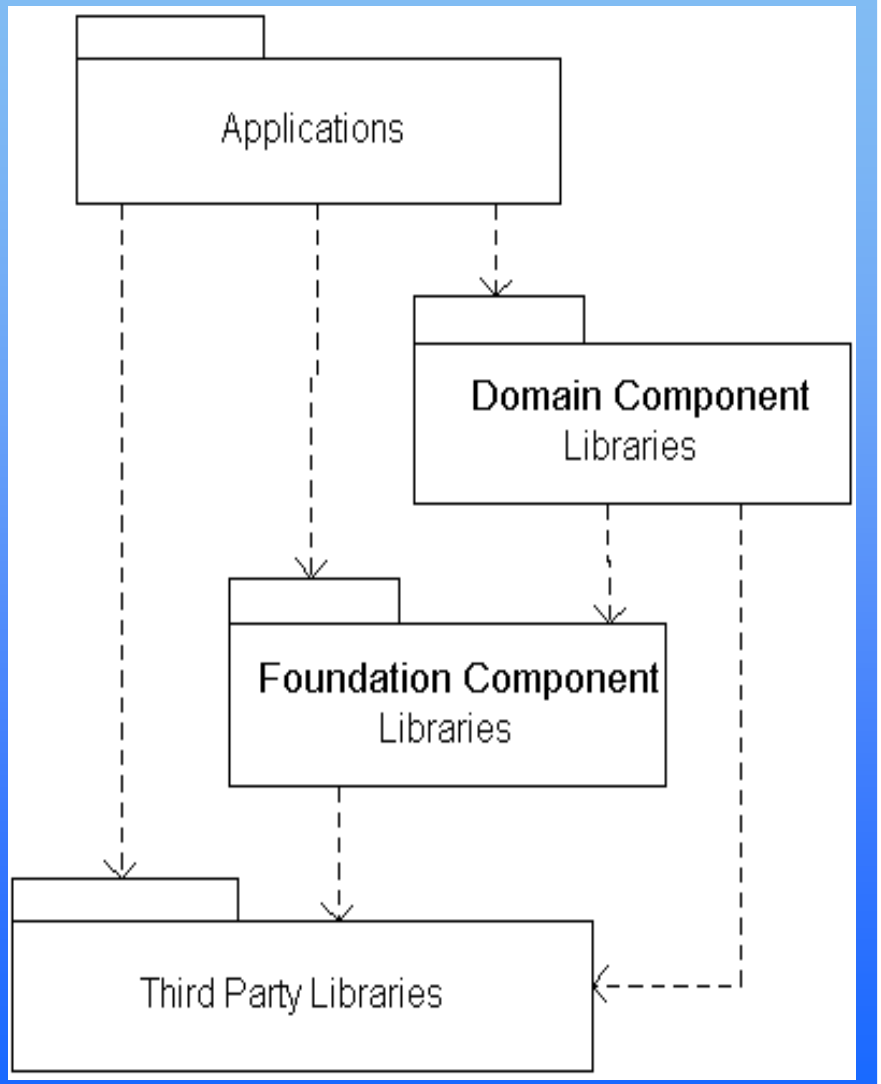
Light-weight Containers – Spring

Layered Architecture

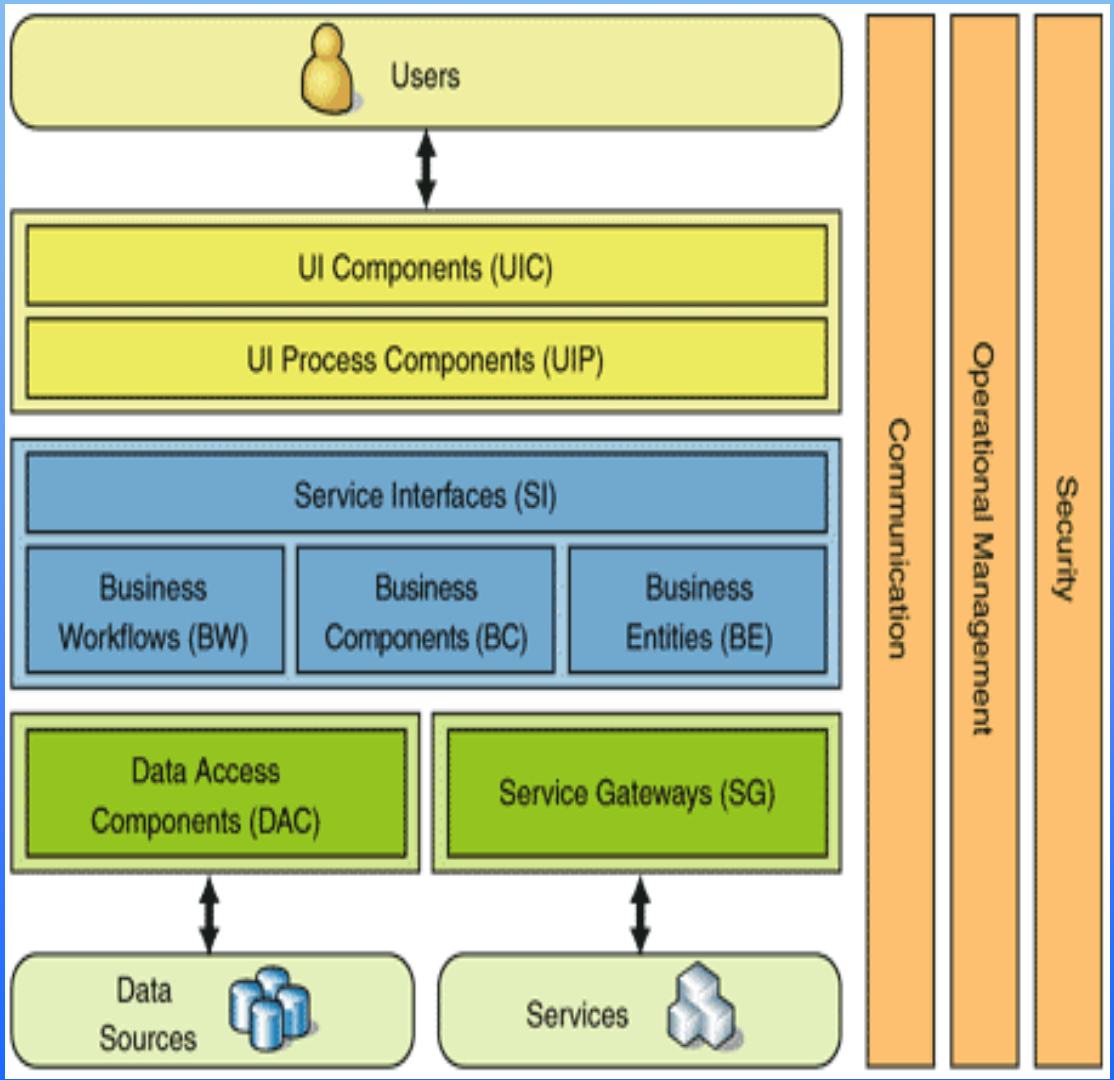
- Challenges with large scale systems
  - large amounts of source code (typically millions of lines)
  - high complexity of interaction between components
  - extensive use of off-the-shelf components
  - use of multiple languages
  - large numbers of developers (often hundreds, often geographically distributed)
  - multiple persistence mechanism (files, relational databases, object databases)
  - distribution of components over several hardware platforms
  - high amounts of concurrency
- Layered Architecture Diagrams are a way of abstracting large systems into manageable concerns.

Layered Architecture (2)





Layered Architecture Diagram - .NET (an Example)



- **Presentation**. The presentation layer provides the application's user interface (UI). Typically, this involves the use of Windows Forms or ASP.NET.
- **Business**. The business layer implements the business functionality of the application. The domain layer is typically composed of a number of components implemented using one or more .NET - enabled programming languages.
- **Data.** The data layer provides access to external systems such as databases. The primary .NET technology involved at this layer is ADO.NET. However, it is not uncommon to use some .NET XML capabilities here as well.

Źródło: MSDN – Patterns and Practices

Model – View – Controller

- A design pattern (?) developed in 1979 by Trygve Reenskaug in Xerox Palo Alto Research Center.
- Commonly utilized in programming GUI on the Java platform.
- Some kind of modification of the three layers architecture.

Model – View – Controller (2)

- Consists of three items (layers):
  - Model - data
  - View - visualization
  - Controller - management

Model

- Represents data (including metadata),
- Defines access rules,
- Responsible for a right modification of the data (including i.e. changes messaging)
- Usually is some reflection of the real world processes.

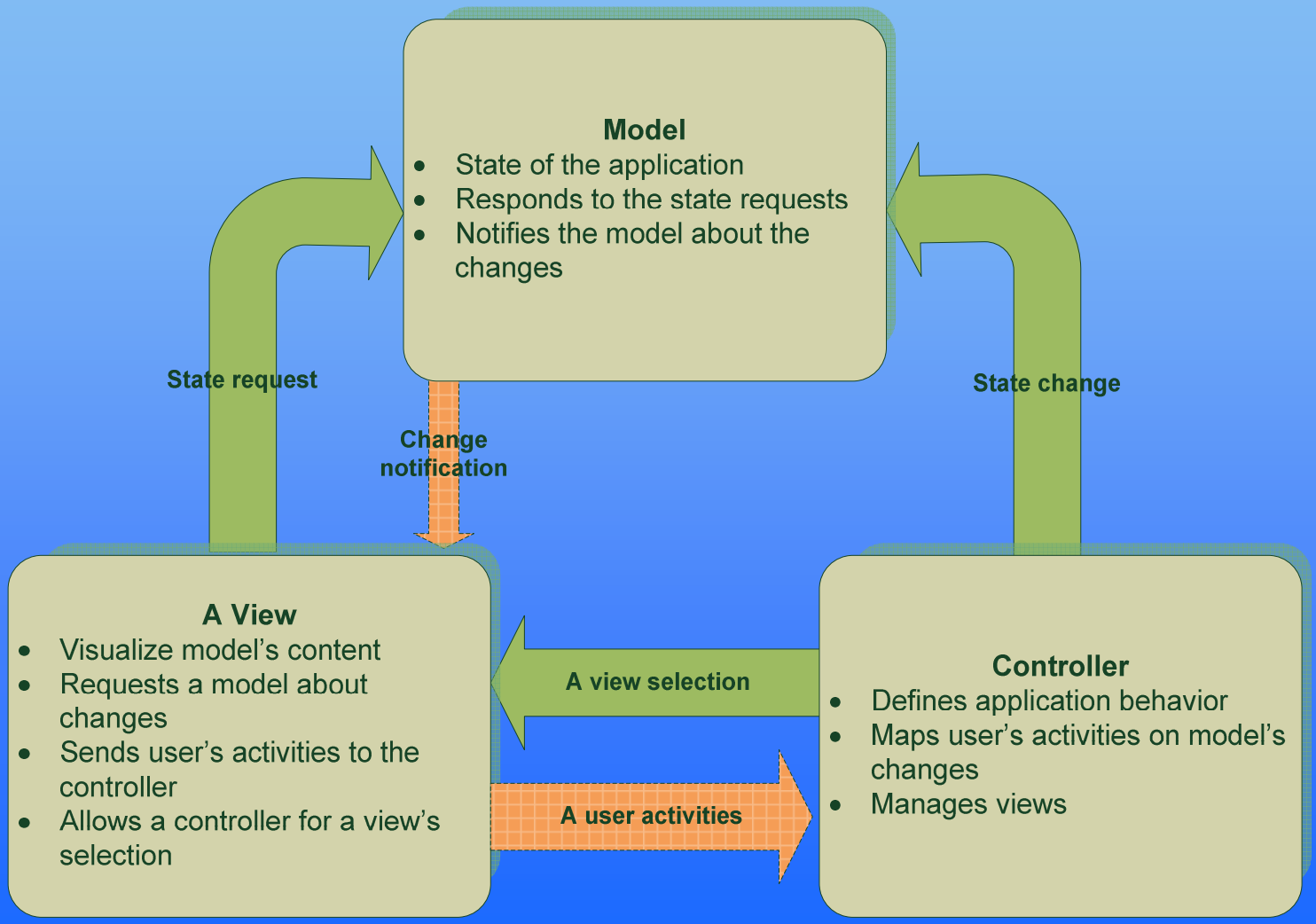
View

- Visualize content of the model (data).
- Precisely specifies how the data looks like.
- Assures a proper reactions on model (data) changes.
  - Push technology – a view registers itself in the model to got changes notifications,
  - Pull technology – a view is self-responsible for querying the model.

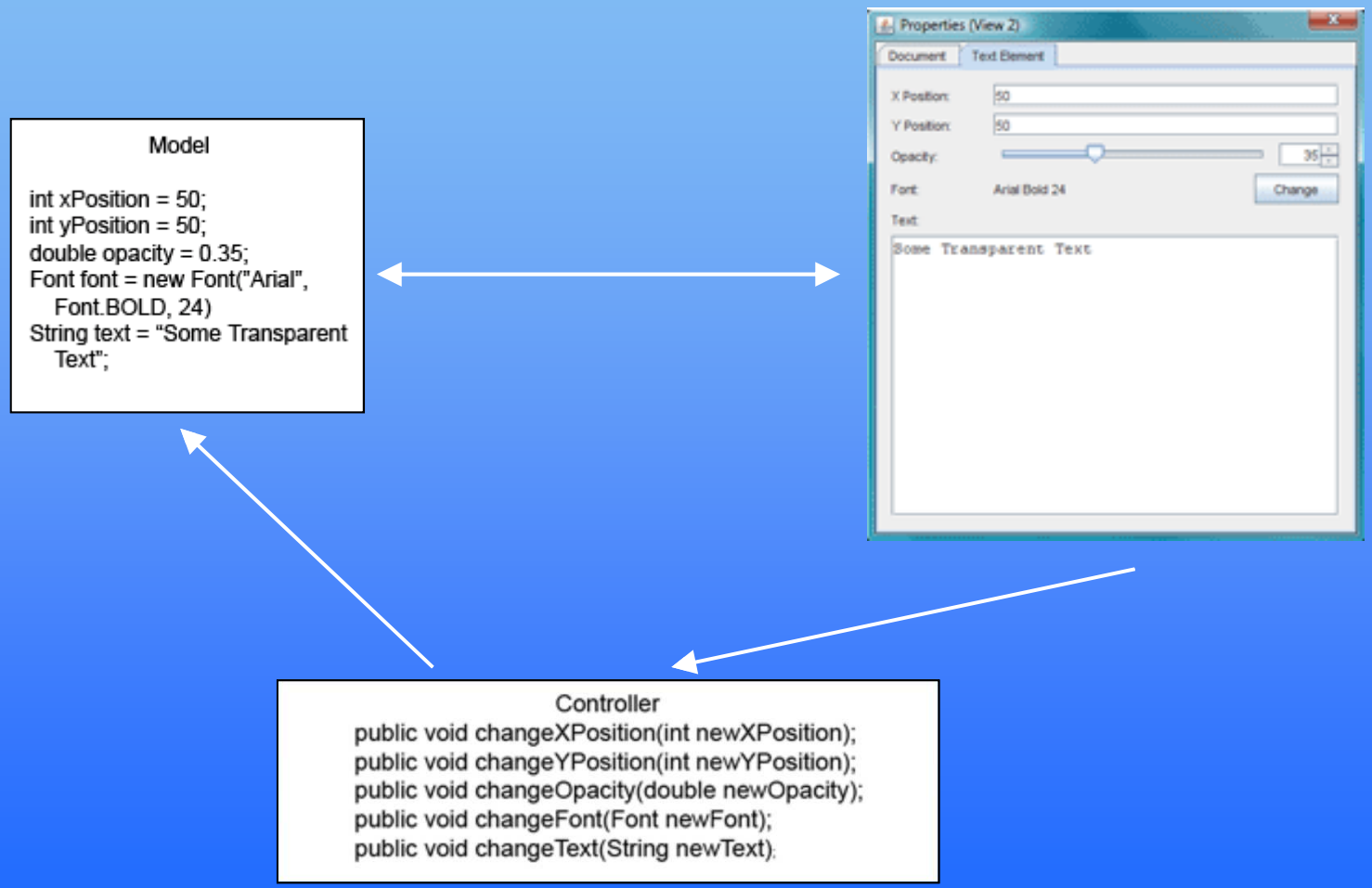
Controller

- Translates user's activities to actions which will be reflected in the model.
- Different kinds of applications:
  - Desktop – events like mouse click or menu selection,
  - Enterprise (web) application - HTTP GET or POST requests
- A controller is also capable of creating a view.

MVC - diagram


Model
State of the application
Responds to the state requests
Notifies the model about the changes
State request
Change notification
State change
A View
Visualize model's content
Requests a model about changes
Sends user's activities to the controller
Allows a controller for a view's selection
A view selection
A user activities
Controller
Defines application behavior
Maps user's activities on model's changes
Manages views

MVC – Java SE 6 sample



MVC – Java SE 6 sample (2)

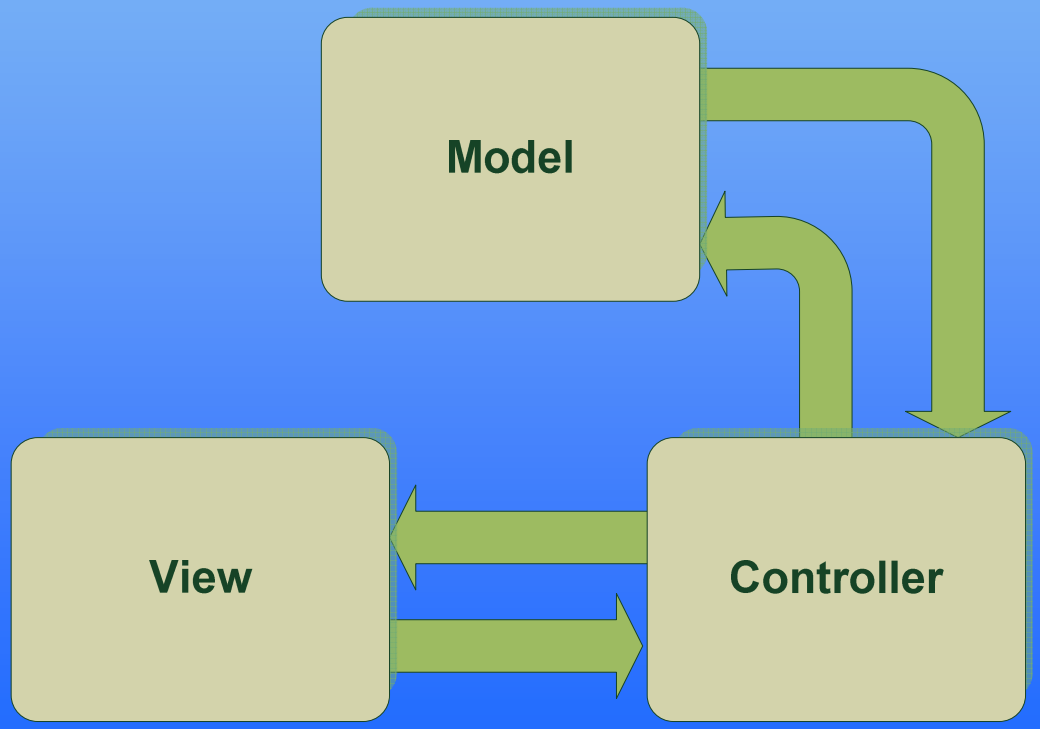
- After creating a model, view and controller:
  - A view registers itself in the model to receiving changes notifications (*push*). The model is not aware of the controller.
  - The controller is connected to the view (using a listener).
  - The controller got an reference to the model.

MVC – Java SE 6 sample (3)

- If a user performs an activity:
  - The view recognizes the GUI activity, i.e. button click.
  - The view calls appropriate method from the controller.
  - The controller optionally modifies state of the model.
  - If the model has been modified then the view(s) is/are notified (using listeners).

MVC – a new approach

- Differences according to the previous version
  - Notifications about model changes are passed through the controller.
  - Hence, the controller controls data flow between the view and the model.
- Such an approach guarantees a better separation of the model and the view.
- Typical three layer architecture?



http://java.sun.com/developer/technicalArticles/javase/mvc/index.html

Summary

- Carefully designed architecture significantly makes easier development of big computer systems.
- Currently the most popular is three-layers architecture
  - GUI,
  - Business logic,
  - Data source.
- There is a lot of existing solutions supporting creating systems using such an approach.