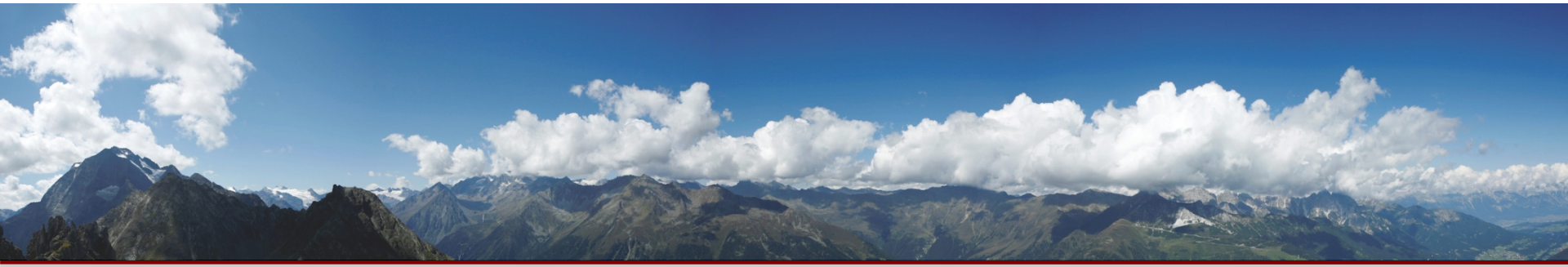
Web Engineering

Web Application Architectures



© Copyright 2013 Ioan Toma, Srdjan Komazec, Nelia Lasierra

Where we are?

STI · INNSBRUCK

#	Date	Title
1	5th March	Web Engineering Introduction and Overview
2	12th March	Requirements Engineering for Web Applications
3	19th March	Web Application Modeling
4	**26th March**	**Web Application Architectures**
5	16th April	Developing Applications with WebML
6	23rd April	Testing and Usability
7	30th April	Web Technologies I
8	7th May	Web Technologies II
9	21th May	Web Application Development Process
10	28th May	Project Management for Web Applications
11	11th June	Web Application Security
12	18th June	Mobile Application Development
13	25th June	Final Exam

Where we are?

STI · INNSBRUCK

#	Date	Title
1	5th March	Web Engineering Introduction and Overview
2	12th March	Requirements Engineering for Web Applications
3	19th March	Web Application Modeling
4	26th March	Web Application Architectures
5	16th April	------
6	**23rd April**	**Developing Applications with WebML** **Testing and Usability**
7	30th April	Web Technologies I
8	7th May	Web Technologies II
9	21th May	Web Application Development Process
10	28th May	Project Management for Web Applications
11	11th June	Web Application Security
12	18th June	Mobile Application Development
13	25th June	Final Exam

Overview

STI · INNSBRUCK

- Web Application Architectures
  - Introduction
    - Software architectures
    - Architectures development
    - Patterns and frameworks
    - Specifics of Web applications
  - Web Application Architectures
    - Architecture types
    - MVC, Client-Server, N-Layer, JSP Model 1 and 2, Apache Struts
- Summary

What is an architecture?

INTRODUCTION

Software Architectures

- “Architecture is defined [...] as the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution.”(IEEE Architecture Working Group, P1471, 1999)

- Architectures describe structure
  - Components of software systems, their interfaces and relationships
  - Static as well as dynamic aspects
  - Blueprint of software system

- Architectures connect software development phases
  - Requirements mapped iteratively to components and their relationships

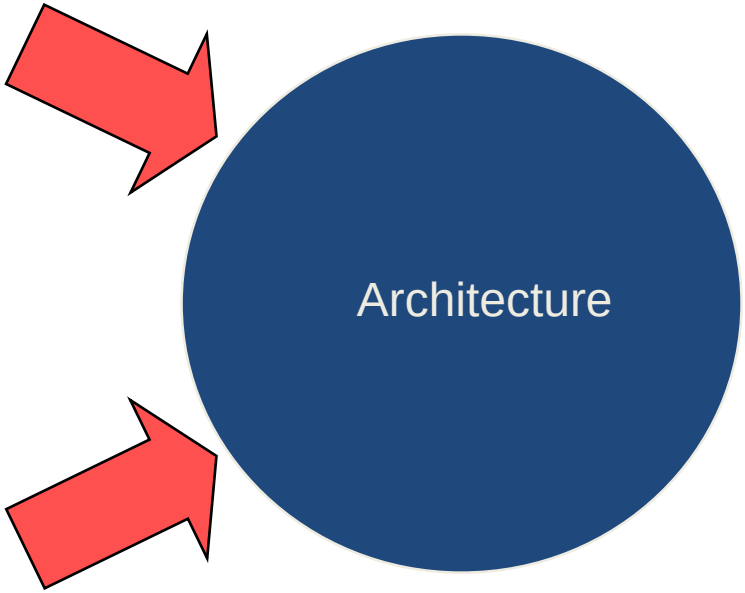
- “Architecture is the set of design decisions [...] that keeps its implementers and maintainers from exercising needless creativity.”(Desmond F. D’Souza and Alan C. Wills, 1999)

- Architectures describe different viewpoints
  - conceptual view: entities of application domain and their relationships
  - process view: system runs, concurrency, synchronization
  - implementation view: software artifacts (subsystems, components, source code)
  - runtime view: components at runtime and their communication

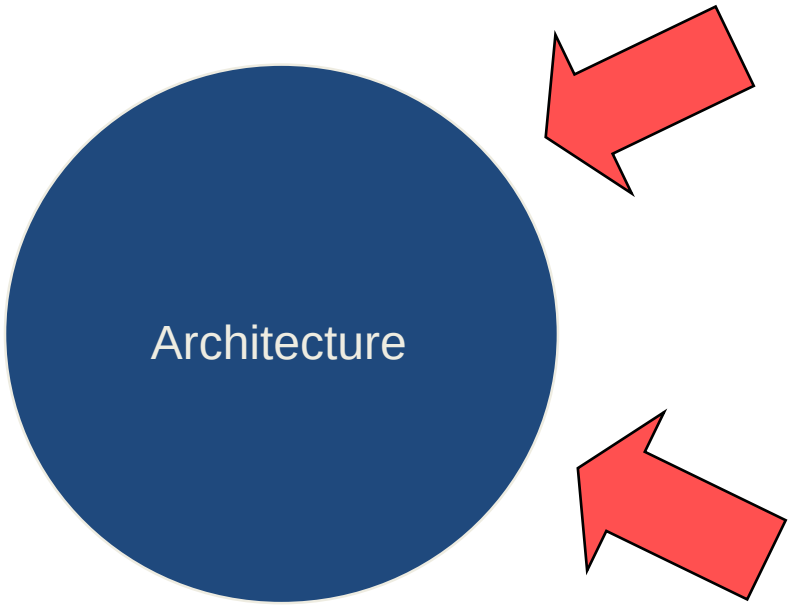
- Architectures make systems comprehensible and controllable
  - structuring according to different viewpoints
  - enables communication between different stakeholders

STI · INNSBRUCK

Functional Requirements
- Clients
- Users
- Other Stakeholders



Non-Functional Requirements
- Performance
- Scalability
- Reusability
- Other?



Technical Aspects
- Operating System
- Middleware
- Legacy Systems
- Other?

Experience with
- Existing Architecture
- Patterns
- Project Management
- Other?

Developing Architectures

STI · INNSBRUCK

- Remember, requirements are always subject to change
  - Organizational & environment changes
  - Ambiguous requirements initially
- Thus, iterative approaches are the suggested means of development
  - Pro: Helps to mitigate design risks
  - Caution: Doesn’t guarantee a good architecture

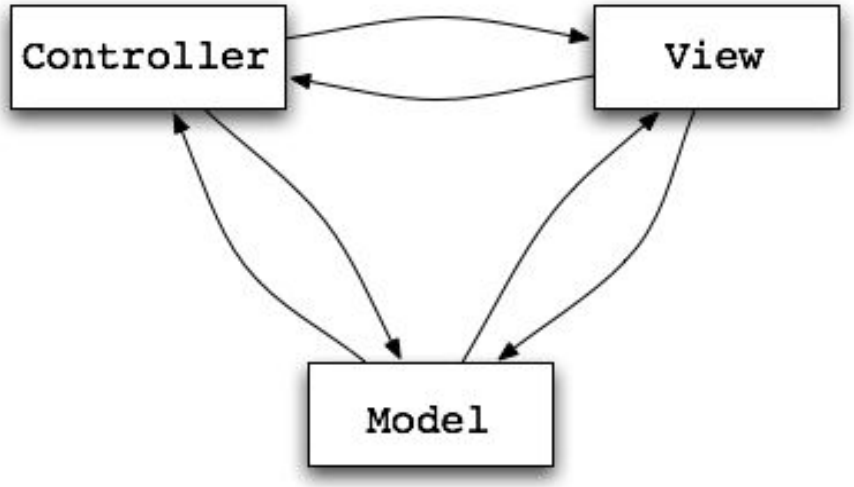
- Patterns describe recurring design problems
- 3 types of patterns
  - Architecture patterns (e.g. MVC)
  - Design patterns (e.g. Publisher-Subscriber)
  - Idioms (e.g. Counted-Pointer in C++)
- They are a guideline, implementation must be grounded to the specific problem
- Patterns need to be “integrated” amongst each other!

- Frameworks: another option to reuse existing architecture
  - Something that provides you a frame to be filled!
- Reuse of existing software objects that just need to be properly configured
- Bound to a specific technology
  - Require training
  - High cost of switch
  - Level of customization not always acceptable
- Together with design patterns
  - help to improve software quality
  - reduce development time

Model View Controller

- Architectural Pattern from Smalltalk (1979)
  - Architectural pattern for user interfaces

- Decouples data and presentation
- Divides the application into 3 interconnected parts
- Express a solution to be adopted in each system
- Eases the development
- Independent of any technology



Model View Controller

STI · INNSBRUCK

- This design pattern separate the concerns of the application: modelling, presentation and actions:

- Model
  - Deals with the data itself of the application and the captures the behaviour
  - Directly manages the data and the rules (logic to process the data)

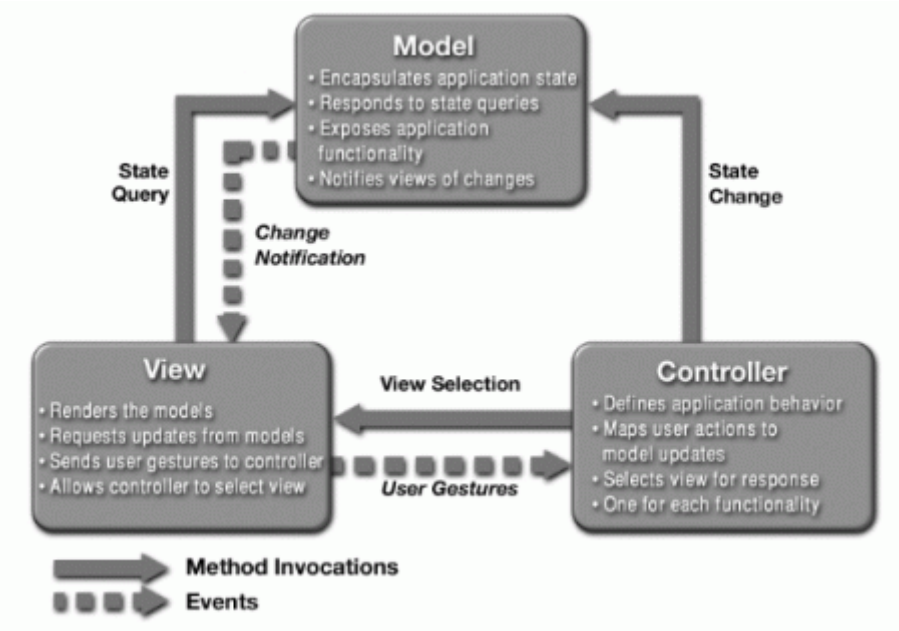
- View
  - Deals with the visualization of the data
  - Gets the data from the model and present to the user

- Controller
  - Handles the actions/input from the user
  - Acts on both the model and the view
  - Control the data flow → takes the input from the view and passes it to the model through the adequate action

Model View Controller

STI · INNSBRUCK

- Model
  - encapsulate application state
  - responds to state queries
  - exposes application functionality
  - notifies views of changes
- View
  - renders the models
  - requests updates from models
  - sends user interaction to controller
  - allows controller to select view
- Controller
  - defines application behavior
  - maps user actions to model updates
  - selects view for response
  - one for each functionality



Generic MVC Structure, courtesy of Sun

Specifics of Web apps

STI · INNSBRUCK

- High quality demands
  - Security
  - Extensibility
  - Adaptability
  - Stability
  - Performance
- Broad range of technical solutions that can be integrated.
  - Hard to evaluate quality demands (various components)
  - Hard to solve (where is the problem?)

Specifics of Web apps

STI · INNSBRUCK

- Inhomogeneity and immaturity of technical solutions
  - Fast product lifecycles
  - Many components: open source – quality?
  - Lack of standards
- Global requirements
  - Multi linguality
  - Cultural adaptation?
    - E.g. Google for Korea

WEB APPLICATION ARCHITECTURES

Architecture Types

STI · INNSBRUCK

- Layering Aspect
  - “Separation of concerns”
  - How many concurrent users are you serving?
  - Shared needs among multiple applications? (e.g., security)

- Data Aspect
  - What kind(s) of data are you delivering?
    - Structured vs. non-structured
    - On-demand vs. real-time
  - What are the bandwidth requirements?
    - Size & nature of data
    - Again, audience concerns

Architecture Types

STI · INNSBRUCK

- Web Platform Architecture (WPA)
  - Platform = Infrastructure
    - Hardware
    - Software modules & configurations
    - Choice of software platform (e.g., J2EE, .NET)
- Web Application Architecture (WAA)
  - Conceptual view of how key business processes and needs are separated & implemented
  - Often domain-specific
  - Greater complexity requires greater modularity

Generic Web (Platform) Architecture

STI · INNSBRUCK

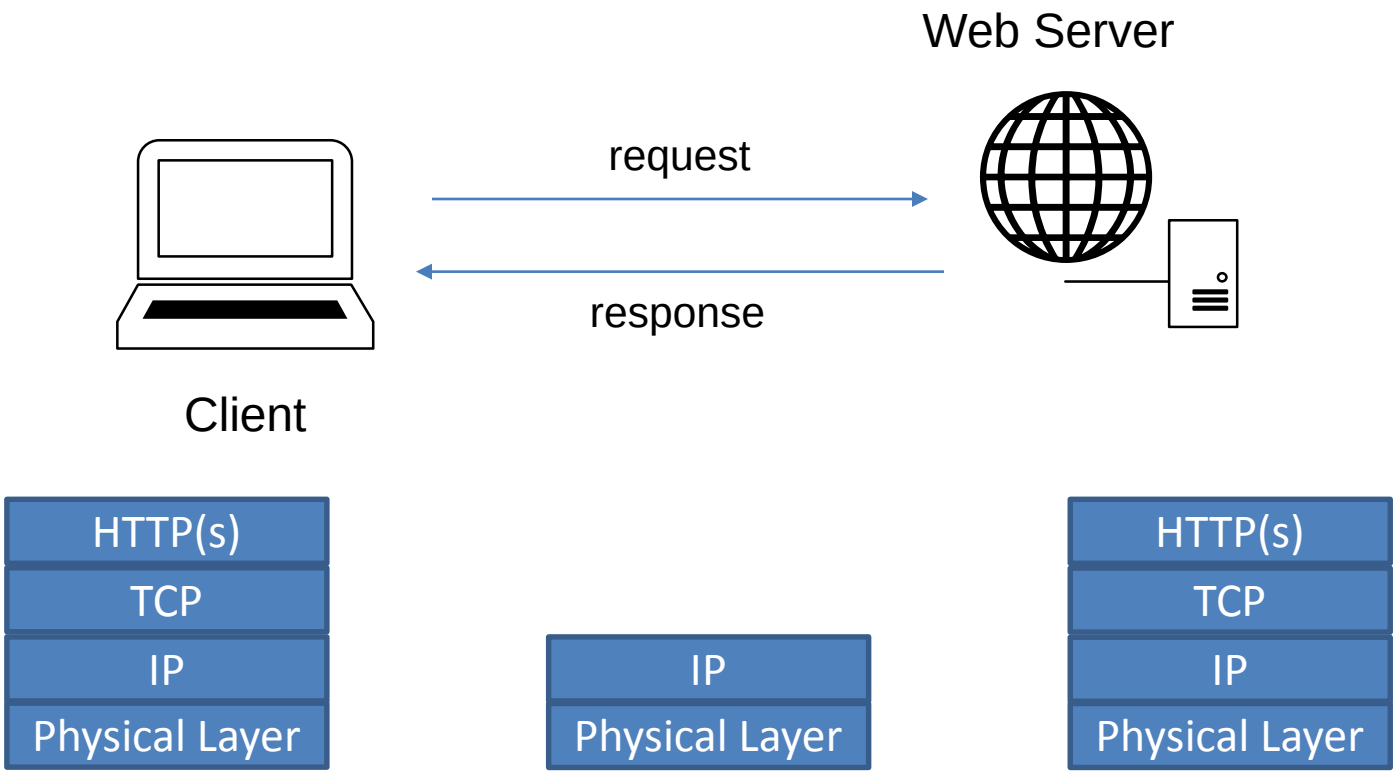
- A web platform includes the software & hardware modules on top of which the web application is going to be built.

- These modules offer a *set of services* (or “capabilities”) that are going to be used by the web application.
    - Network/Communication capabilities → TC/IP (Internet) and HTTP
    - Data store capabilities → Database
    - Visualization capabilities → e.g. show graphically the information
    - Logic execution capabilities

- A web “platform” is based on:
  - TCP/IP
  - HTTP
  - HTML

Generic Web (Platform) Architecture

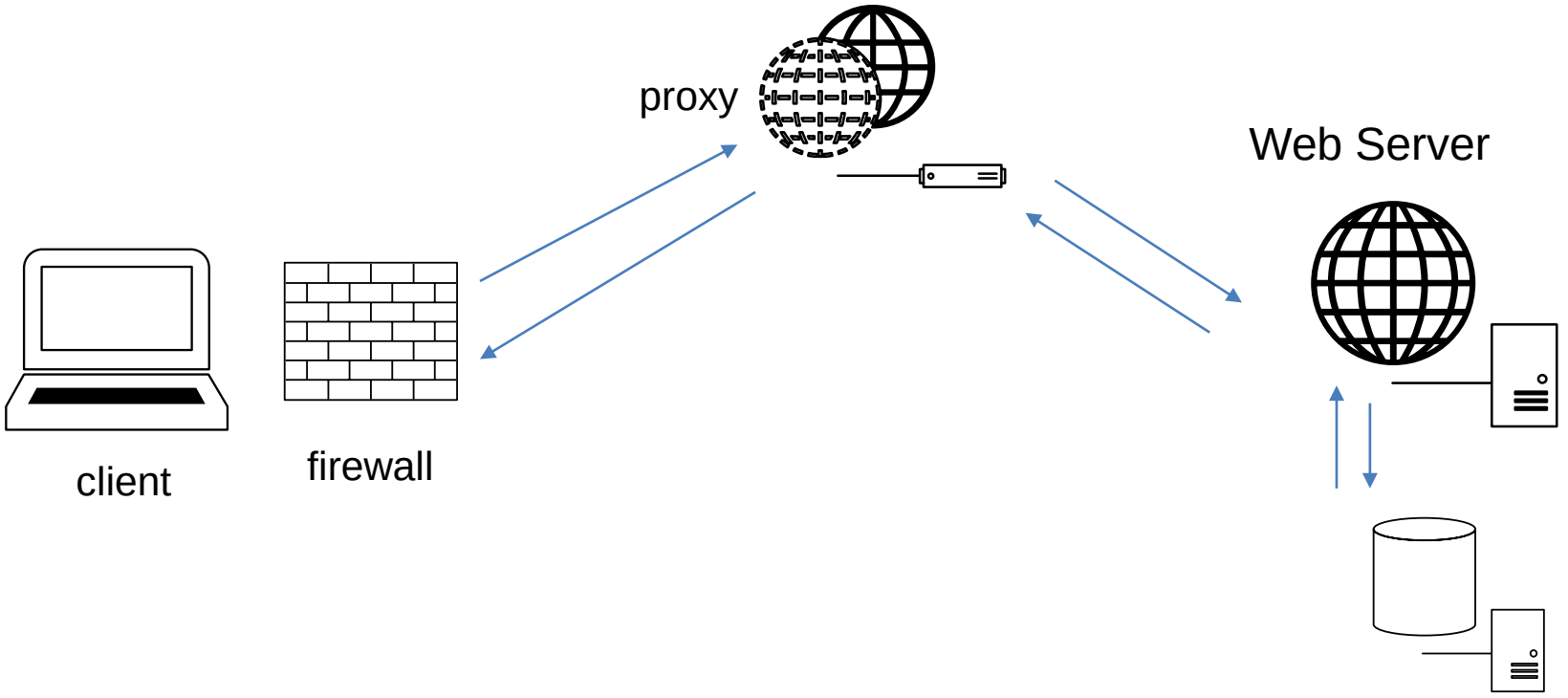
- It's essentially a Client/Server architecture!
  - In term of patterns one of the simplest ones



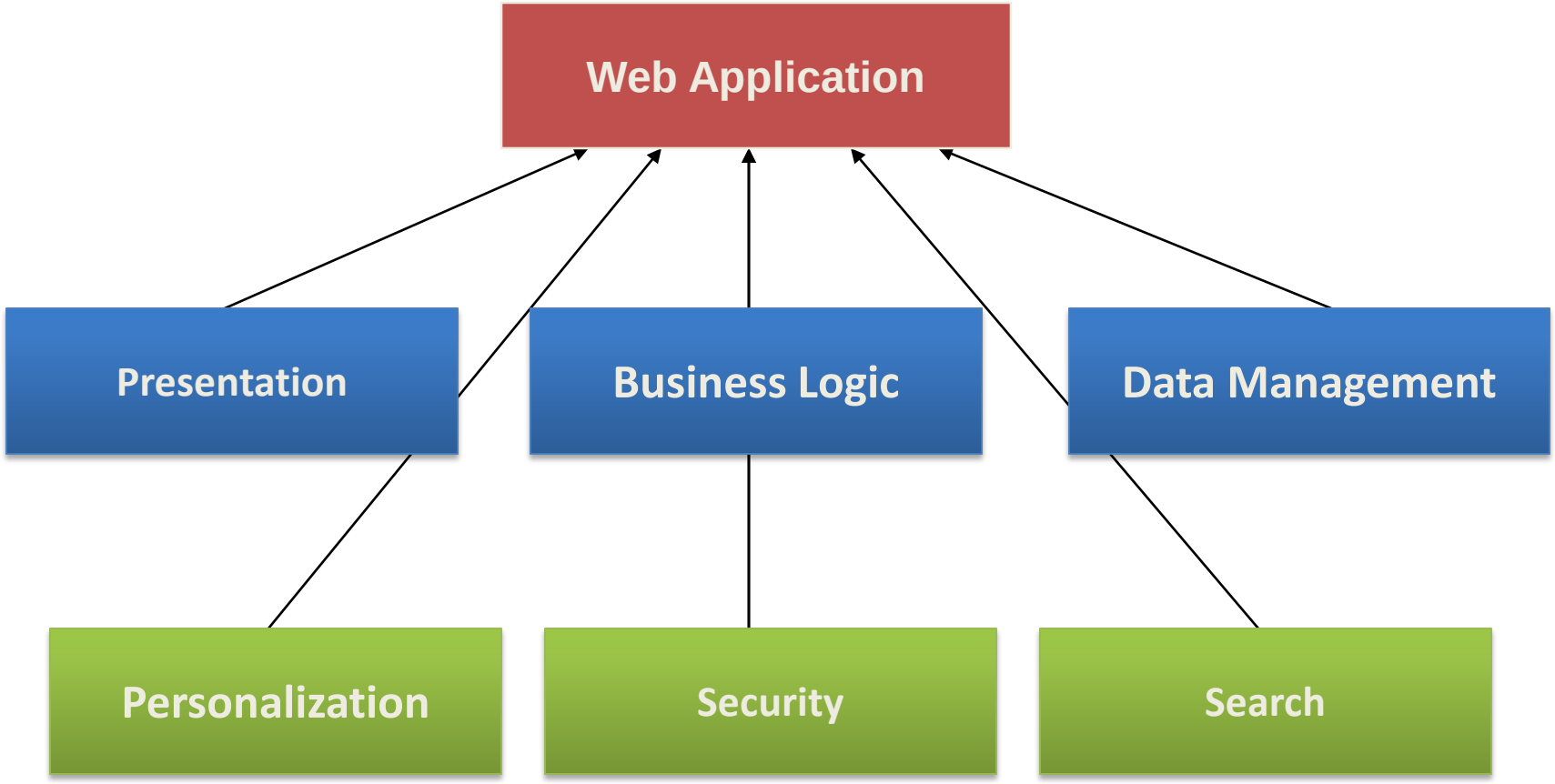
Generic Web (Platform) Architecture

STI · INNSBRUCK

- But still things can get complex…
  - Components on the network (firewall, proxy, load balancer)
  - Components in the intranet (Web server, application server, data base, legacy systems, web services)



Example of a WAA



Platform vs application architecture

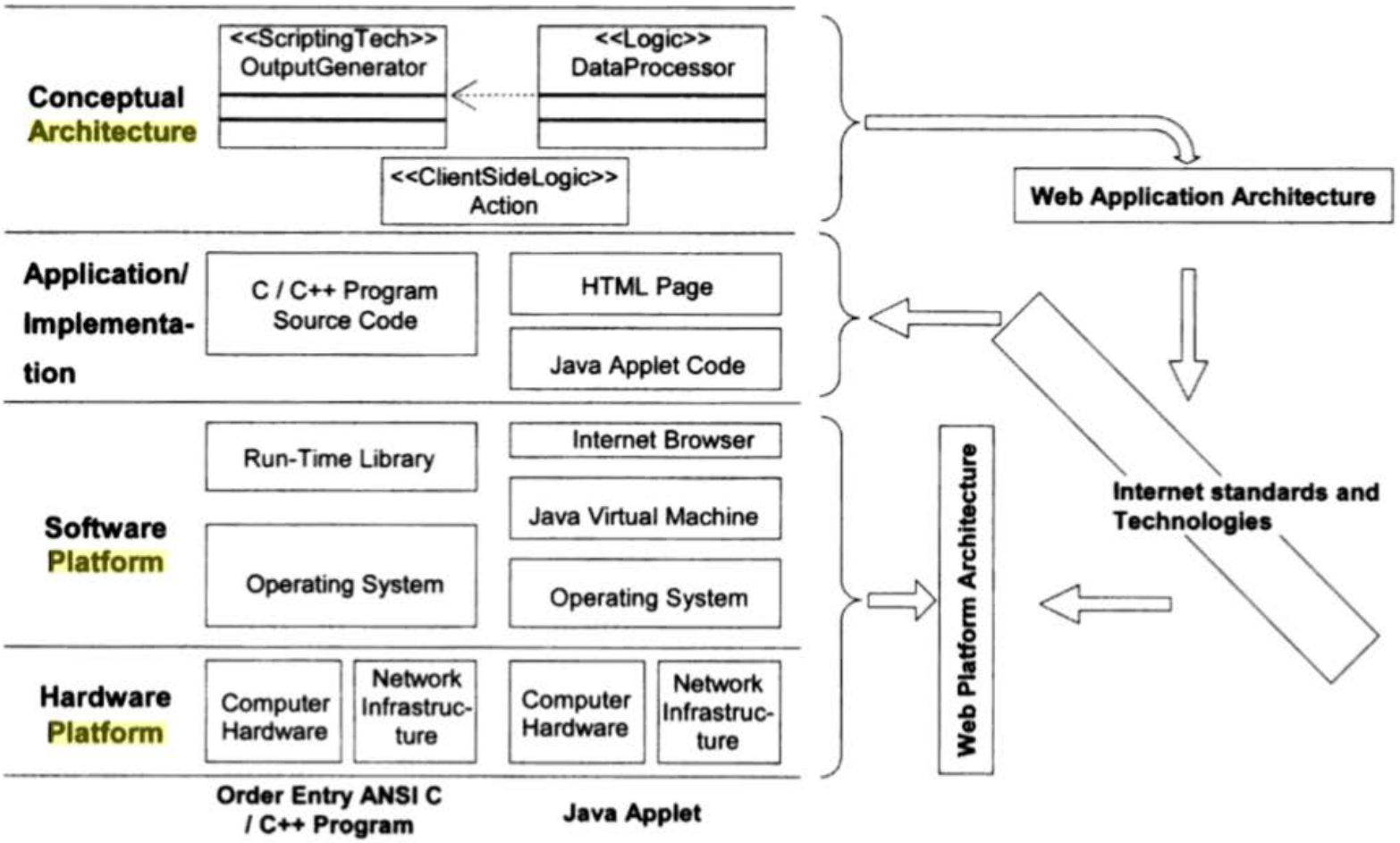


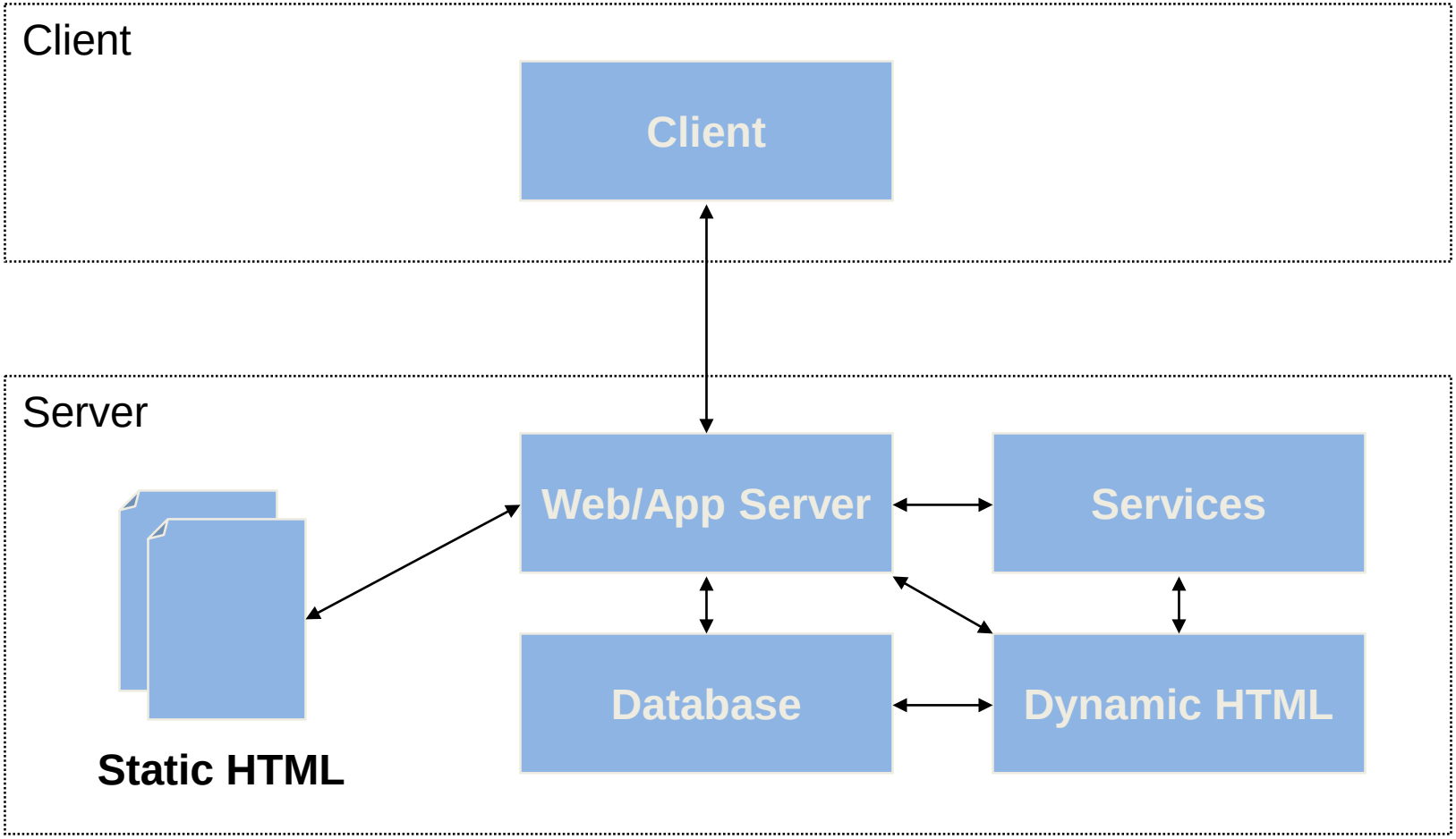
Image taken from: Guide to Web Application and Platform Architectures
By Ilia Petrov, Christian Meiler, Udo Mayer
https://books.google.at/books?id=Ajn-nP7IdRQC&pg=PA28&lpg=PA28&dq=web+platform+architecture&source=bl&ots=GmSoxC_oyY&sig=AzVXag6gnUiMRk7ZRfRAP6PHv2c&hl=en&sa=X&ei=Zs8SVdXeCsbqUufNgrAM&ved=0CEUQ6AEwBg#v=onepage&q=web%20platform%20architecture&f=false

N-tier/layer Architectures

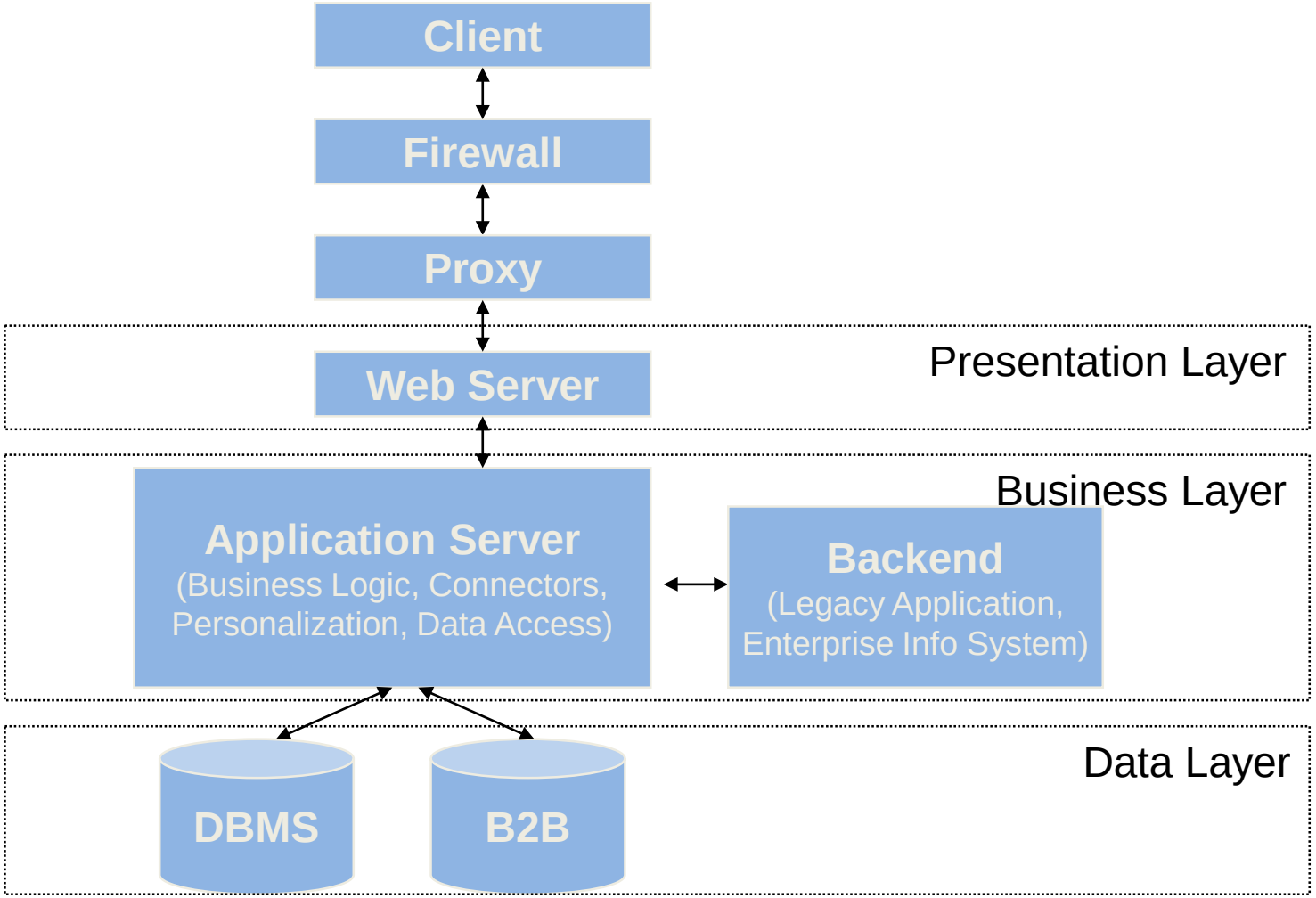
STI · INNSBRUCK

- N-tier/layer architectures
  - Separate components according to different layers/tiers:
    - Presentation: → user interaction with the system
    - Logic: → core functionality of the system
    - Data: → access to the data to be used in the system
  - This separation can be:
    - **Tiers :** physical separation
    - N-tier architectures:
      - 2-tier architecture: client/server
      - 3-tier architecture: client-server-dataserver
    - **Layers :** logical separation
      - Layers group the software components that make up the application or service.
      - These layers may be located on the same physical tier, or may be located on separate tiers.
      - 3 – layer architecture:
        - » Presentation
        - » Logic
        - » Data

Client/Server (2-Layer)



N-Layer Architectures



Why an N-Layer Architecture?

STI · INNSBRUCK

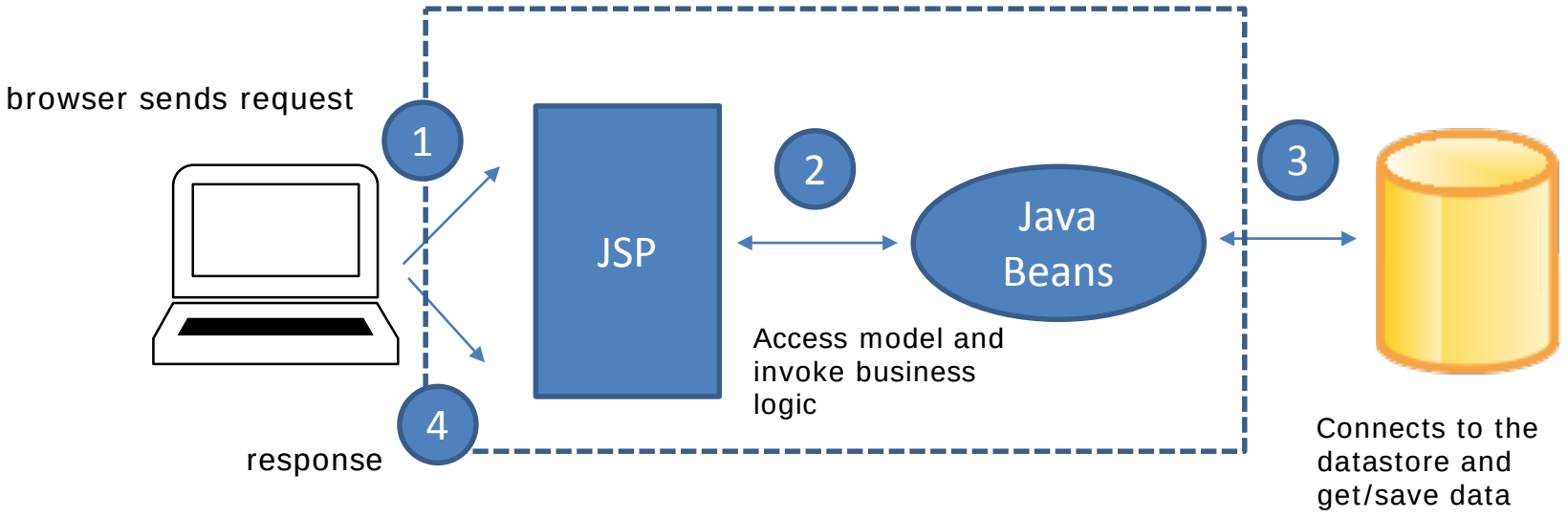
- Separating services in business layer promotes re-use among applications
  - Loose-coupling – changes reduce impact on overall system.
  - More maintainable (in terms of code)
  - More extensible (modular)

- Trade-offs
  - Needless complexity
  - More points of failure

JSP-Model-1 Architecture

STI · INNSBRUCK

- Easier way to deploy a JSP based web site.
- Some separation between content (Model JavaBeans) and presentation (JSP).
- This architecture is good enough for small applications.
- Larger applications have a lot of presentation logic → increase of code in the JSP



http://www.java-samples.com/showtutorial.php?tutorialid=349

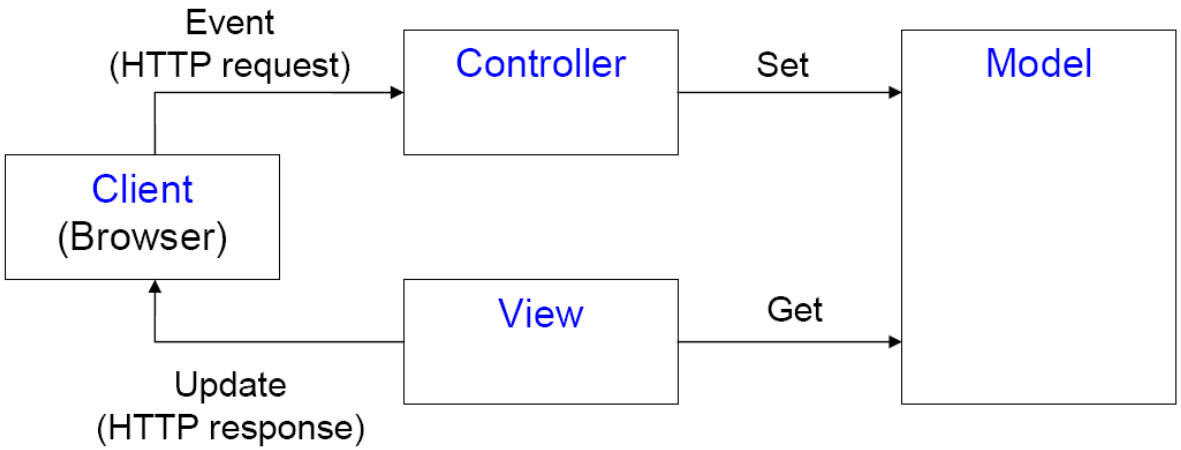
JSP-Model-1 Architecture

STI · INNSBRUCK

- Advantages:
  - Easy and quick development
- Disadvantages:
  - Navigation control is decentralized
  - Time consuming
  - Hard to extend

http://www.javatpoint.com/model-1-and-model-2-mvc-architecture

- **Adaptation of MVC for the Web**
  - Model 2 (Sun) → MVC architecture for Web-based applications
  - stateless connection between the client and the server
  - notification of view changes
  - re-querying the server to discover modification of application's state



Model-View-Controller 2 (MVC 2)

STI · INNSBRUCK

1) HTTP requests are passed from the client to the Controller
2) The Controller updates the Model (for the selection of the appropriate action)
3) Then invokes the controller decides on the appropriate view according to the last action and the output.
4) The appropriate view is shown to the users

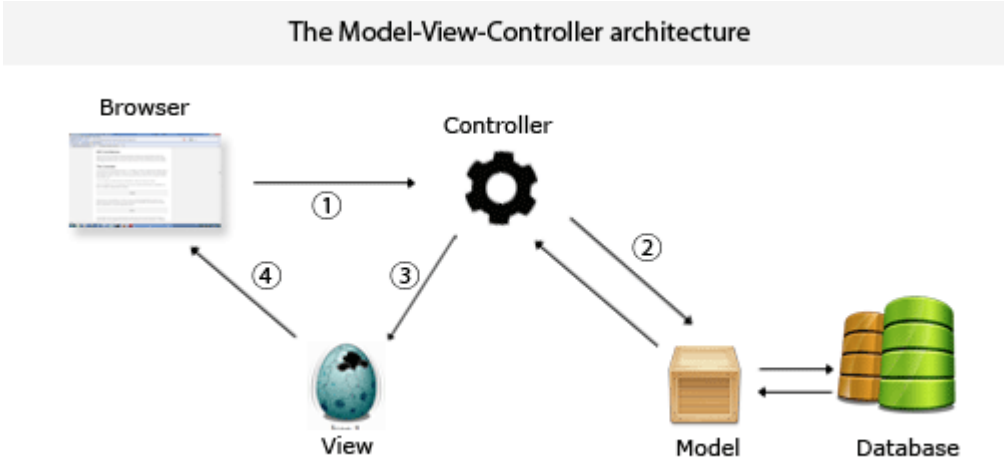
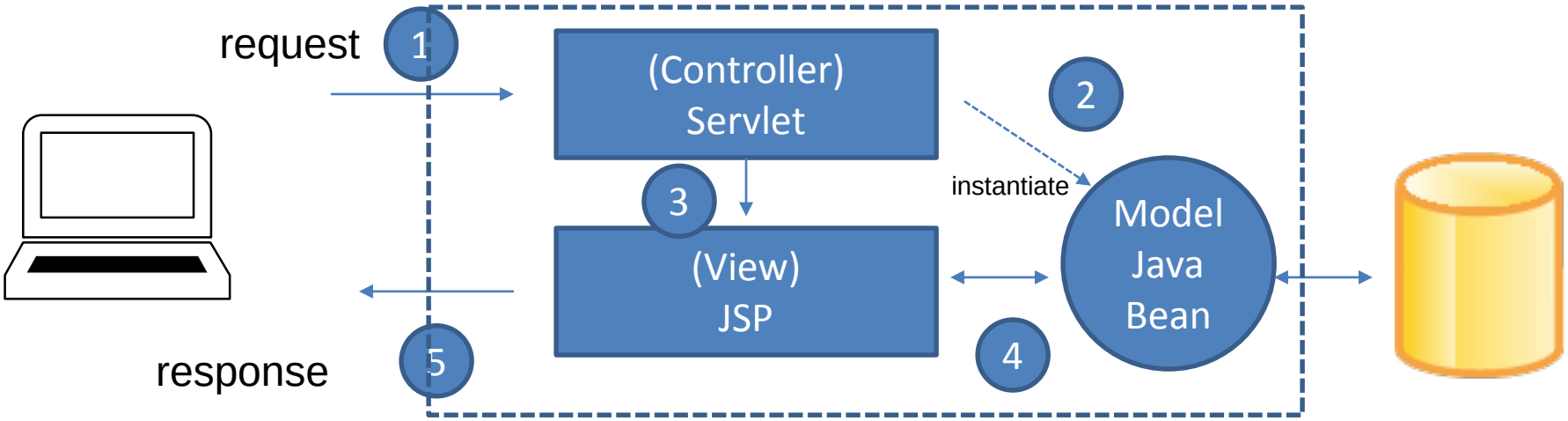


Image from : http://www.9lessons.info/2011/02/getting-start-with-ruby-on-rails.html

JSP-Model-2 Architecture



1. user request
2. create/change model
3. create/change view
4. generate output
5. server response

http://kb.deister.net/index.php/JSP_Acces_Models

- Web application frameworks based on MVC
  - Struts 1 and Struts 2
    - Strut 1:
      - Action servlet, action, action form, jsp
        - » Ex: http://www.javaguru.biz/struts1-framework-architecture/
      - Struts.xml, web.xml
    - Strut 2:
      - Actions, Interceptors …
        - » Ex: http://www.tutorialspoint.com/struts_2/struts_architecture.htm
  - Ruby on Rails
    - » Ex: http://www.9lessons.info/2011/02/getting-start-with-ruby-on-rails.html

(*) Worth to take a look to …

http://hibernate.org/ :

Provides a framework for mapping an object oriented domain model to a traditional relational database.

Spring: https://spring.io/guides

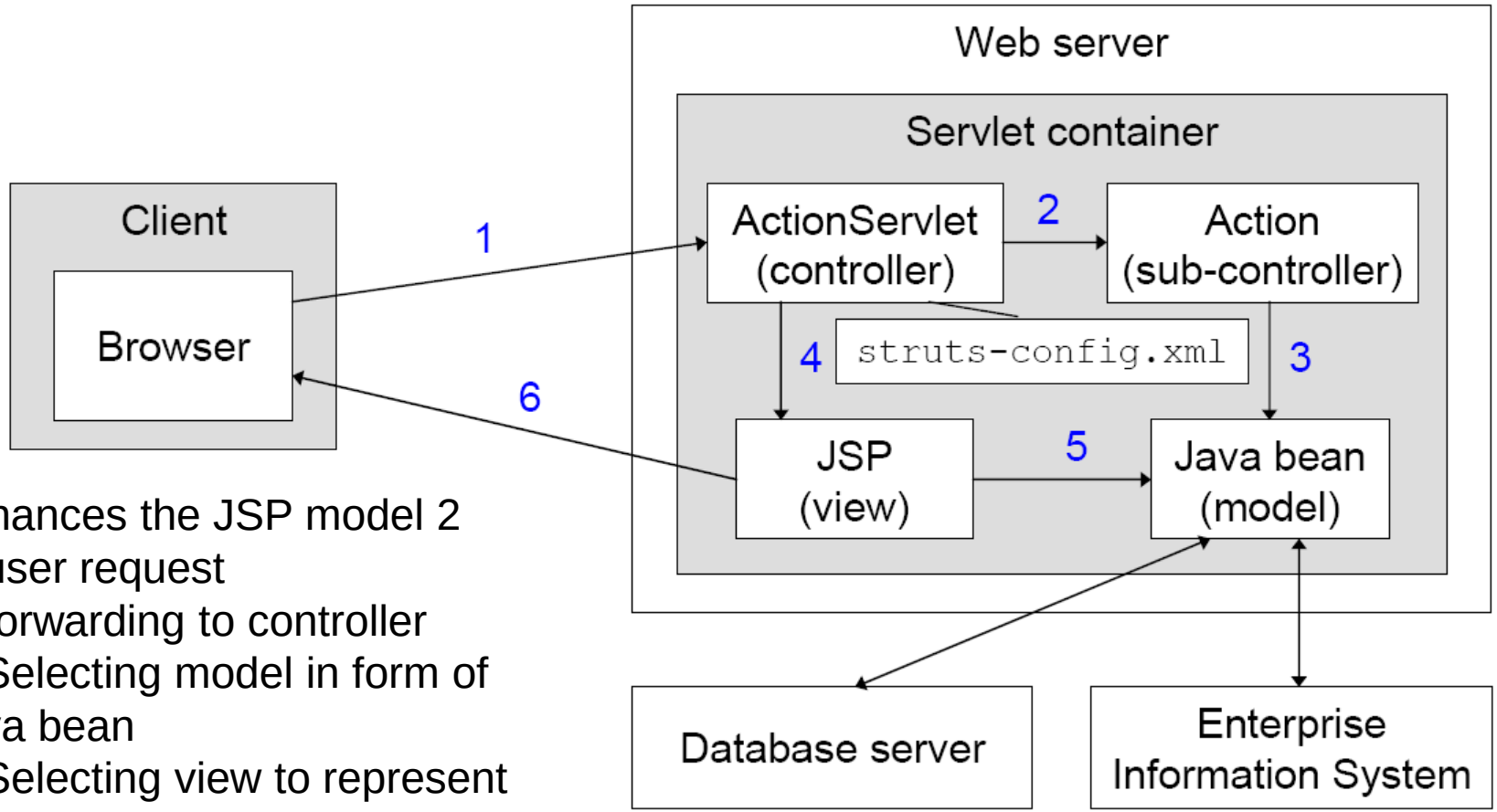
JSP-Model-2 Architecture

STI · INNSBRUCK

- Advantages:
  - Navigation control is centralized.
  - Easy to maintain
  - Easy to extend
  - Easy to test
  - Better separation of concerns
- Disadvantages:
  - You need to write the controller code

http://www.javatpoint.com/model-1-and-model-2-mvc-architecture

Apache Struts Architecture



Enhances the JSP model 2
1. user request
2. forwarding to controller
3. Selecting model in form of Java bean
4. Selecting view to represent contents
5. generate output
6. server response

http://struts.apache.org/

WRAP-UP

Things to keep in mind (or summary)

- Good design of architecture is crucial
- You can leverage on patterns and frameworks
  - Both have advantages and disadvantages
- Design is constrained on Web “infrastructure”
- MVC is the most commonly used pattern

Bibliography

STI · INNSBRUCK

- Mandatory reading
  - Kappel, G., Proll, B. Reich, S. & Retschitzegger, W. (2006). *Web Engineering*, Wiley & Sons. **4th Chapter**
- Web links
  - Model-View-Controller pattern http://en.wikipedia.org/wiki/Model-View-Controller

Next Lecture

STI · INNSBRUCK

#	Date	Title
1	5th March	Web Engineering Introduction and Overview
2	12th March	Requirements Engineering for Web Applications
3	19th March	Web Application Modeling
4	26th March	Web Application Architectures
5	16th April	------
6	**23rd April**	**Developing Applications with WebML** **Testing and Usability**
7	30th April	Web Technologies I
8	7th May	Web Technologies II
9	21th May	Web Application Development Process
10	28th May	Project Management for Web Applications
11	11th June	Web Application Security
12	18th June	Mobile Application Development
13	25th June	Final Exam

Questions?