Overview of Web Application Development
Web Technologies I.

Zsolt Tóth
University of Miskolc

2018
Overview

1. Architecture
2. Component–based Development
3. Model View Controller

Back–end

4. Programming Languages
5. Server Side Components

Front–end

6. Programming Languages
7. Single Page Applications

Tutorials

Table of Contents
Client–Server Model

- **Simple Model**
  - Clients request services
  - Server waits for and servers requests

- **Widely used**
  - FTP, SSH
  - WWW, SMTP

- **Centralized**

- **Web Applications**
  - Information Systems
  - Search Engines
  - Social Media
  - Web Shops
  - e–Government
  - e–Banking
  - Monitoring Systems
N–tier Architecture

- Detailed than Client–Server Model
- Tiers are not Layers
- Tiers have specific functions, purpose
- Typical tiers
  - Presentation
    - Client–side
    - Web sites
    - Mobile / Desktop applications
  - Business
    - Server–side
    - Business logic
  - Database
    - Server–side
    - Not available directly.
Presentation Tier

- Front-end
- Send requests, use services
- Visualize response
- Run on client side
- User Interface
  - text, tables, lists
  - images, charts
  - audio, video
- Controls
  - button, text field,
  - checkbox radio–button
  - drop–down list

Web Applications
- Thin client
- Only visualization
- Client–side checking!

Desktop Applications
- Thick client
- Installing, updating?

Mobile
- Thin or thick client
- Browser vs native application
- Different display
Business Tier

- Back–end
- Server–side
- Provides services
- Handle internal exceptions
- Seems to be a huge monolithic application but it may be made of sub–systems
- sub–systems are built from components

Components

- **model** – Domain classes, POJOs
- **persist** – Storage, JDBC, myBatis, Hibernate, etc.
- **service** – High level functions, Use–Case
- **controller** – Handle HTTP requests, sanitize input data
Database Tier

- Storage of business objects
- Changeability of the storage implementation
- Separated from business tier
- Well-known trusted technologies are used
- Data is accessed via 3rd-party APIs

DMBS
- Oracle, DB2, MS SQL Server
- MySQL, PostgreSQL
- Database expert is required!

NoSQL
- MongoDB
- Neo4j

File System
- Custom directory hierarchy
- Security issues
Terms

**sub-system**  A part of the entire system that provide a well-defined functionality.

**module**  A development unit that has a well-defined purpose. Modules are identified by their name.

**component**  A module used by another module.

**artifact**  A specific version of a module. A module with a version number.

These terms are slightly different. During the course, we will stick to these definitions. Do not mix them.
Modules, Components

- Tests
  - Unit
  - Component
  - Integration
- Dependencies
- Build process
- Deployment

Version number
- major.minor.build.revision
- alpha, beta, release
- candidate, commercial
- distribution
- Never use multiple version of the same module!
Interface–based Programming

- Decouple components
- Loose coupling
- Exchangeable components
- Ease design and development

- Separates definition and implementation
- Defines expected behavior
  - return type
  - parameters
  - exceptions
  - documentation
Benefits

- Complex tasks can be broken down
- Increase re-usability
- Simplify tasks

- Separate different
  - programming language
  - tools
  - technologies

Divide and Conquer!
Tools – Maven

- **Application**
  - `mvn <goal>`
  - Eclipse plugin

- **Packaging**
  - `pom`
  - `jar, war`

- **Project Structure**
  - `src`
    - `main`
    - `test`
  - `target`
  - `pom.xml`
    - `groupId, artifactId, version`

**Build tool**

1. validate
2. compile
3. test
4. package
5. integration-test
6. verify
7. install
8. deploy

**Dependency manager**

- **Repository**
  - 1. local ~/.m2
  - 2. company (Neuxs)
  - 3. global (www.maven.org)
Model View Controller

- Architectural Pattern (1970)
- User Interface Implementation
- Desktop Applications
  - traditionally
- Web Applications
  - popular
- Variants
  - **MVP** Model–View–Presenter
  - **MVA** Model–View–Adapter
  - **MVVM** Model–View–ViewModel
Model

Domain Objects
- POJOs
- Beans
  - public getter / setter methods
  - public default constructor
- Describe the Structure of the Entities

Business Logic
- Processes
- Tasks
- Calculations
- Rules

Persistence
- Database
- File System
- Text or binary files
View

- Visualize model objects
- Shown to the user
- Different representations
  - Forms
  - Tables
  - Charts
- Capture user interaction

User Interface

- Buttons
- Text fields
- Labels
- Images
- Check box
- Radio button
- Drop-down lists
Controller

- Connects Model and View
- Invoke Business Services
- Transform user interactions
- Update View
Table of Contents

1 Overview
   - Architecture
   - Component–based Development
   - Model View Controller

2 Back–end
   - Programming Languages
   - Server Side Components

3 Front–end
   - Programming Languages
   - Single Page Applications

4 Tutorials
Common Gateway Interface

- CGI Program / Script
- PHP, C
- Generate HTML

- Started by the Web Server
- Create process per HTTP Requests
- Costly
**Script Languages**

**PHP**
- Most popular
- Local companies use it.
- Frameworks
  - Yii
- Content Management Systems
  - Joomla
  - WordPress
- Easy–to–start
- Supported by web server hosting companies
- Wikipedia

**Python**
- High–level, general purpose language
- Popular but not in Miskolc
- Easy–to–Learn
- Google, Youtube

**Ruby**
- Ruby on Rails Framework
- Not popular in Miskolc
- Twitter
Object Oriented Languages

- Web Applications
  - Complex,
  - Reliable,
  - Stable
- Expensive Infrastructure
- Components can be separated easily
- Good, well-tested code (hopefully 😊)

Development
- Slow
- Standardized
  - Fixed roles
  - Methodologies
- Difficult
  - If you do it well 😊
  - Lots of technology
- Both Java and C# are used by local companies

You will learn about it in details at the "Web Application Development in Java" course.

In this course you will use an existing Java based web application.
model & util

util
- General, common functions
- Logging

model
- Domain objects
- POJOs
- No beans
- Exceptions related to the Domain
**persist & persist-mysql**

**persist**
- Data Access Objects
- Interfaces
- Exceptions related to Persistence
  - Existing object
  - Not found
  - Permission Denied
  - ...
- Do NOT related database!
- Defines
  - function
  - task
  - NOT implementation

**persist-mysql**
- May be merged into **persist**
- MySQL specific implementation
- Arbitrary technologies
  - JDBC
  - Hibernate
  - MyBatis
- Alternatives
  - persist-oracle
  - persist-postgre
  - persist-xml
  - persist-filesystem

Zsolt Tóth (University of Miskolc)
service & service-impl

service
- Interfaces for High-Level Functions
- Exception
- Processes, Tasks
- Use-Case

service-impl
- Can be merged into service
- Use DAOs via Interface
  - persist
controller

- Connect services and view
- Map services to URL
- Convert HTTP Requests / Responses
  - Parameters
  - Status codes
- Conversion between business and value objects.
  - Server uses Java Objects
  - HTTP is text based
- Sanitize input data

- Java EE
- Servlets
- Spring
- Dispatcher Servlet
- Static content ???
- Endpoint reference 😊
Data Transfer Object

- Model object cannot be published.
  - may contain sensitive data
  - not beans
- DTOs are beans
  - JSON
- Have no business logic
- Simplify the requests
  - Defines complex, composite structures
  - One parameter can be enough

Assembler

- Mapping between business objects and DTOs
- Capture the rules and logic

```json
{"employees" : [
  {
    "firstName" : "John",
    "lastName" : "Doe"
  },
  {
    "firstName" : "Anna",
    "lastName" : "Smith"
  }
]}
```
# Table of Contents

1. Overview
   - Architecture
   - Component–based Development
   - Model View Controller

2. Back–end
   - Programming Languages
   - Server Side Components

3. Front–end
   - Programming Languages
   - Single Page Applications

4. Tutorials
HTML

- Plain text
- Markup Language
- Interpreted and visualized by the Browser
- Structural elements
  - divisions
  - paragraphs
  - images
  - tables
  - lists
  - forms

```html
<html>
<head>
<title>Page Title</title>
</head>
<body>
<h1>This is a Heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```
CSS

- Cascaded Style Sheets
  - Browser
  - External
  - Internal
  - In-line
- Defines the appearance
- Text based
- CSS pre-processors
  - less.js
  - SASS
  - Stylus
- Templates
  - Bootstrap

```css
body {
  background: blue;
}

h1 {
  color: white;
  text-align: center;
}

p {
  font-family: verdana;
  font-size: 20px;
}
```
JavaScript

- Client side programming
- Dynamic behavior
- Script Language
- Interpreted by the Browser
- Event handlers
  - Buttons
  - onLoad
- AJAX
- Frameworks
  - AngularJS
  - TypeScript

```javascript
document.getElementById("demo").innerHTML = "Hello JavaScript";
```
Single Page Applications

Separate pages
- Independent HTML pages
  + Easy–to–create
  - Multiple pages
  - Redundancy
  - Huge network traffic

Modular Site
- AJAX
- Page Fragments
  + No redundancy
  + Moderate network traffic
  - HTML fragments
  - Re-usability

Single Page Applications
- AJAX
- JSON / XML Messages
  + Low Internet Traffic
  + Re–usable components
  + Loose coupling
  - Client–side computation
  - Not so easy–to–learn
Table of Contents

1 Overview
   - Architecture
   - Component–based Development
   - Model View Controller

2 Back–end
   - Programming Languages
   - Server Side Components

3 Front–end
   - Programming Languages
   - Single Page Applications

4 Tutorials
Goal

- Develop front-end for an existing web applications!
  - HTML
  - CSS
  - JavaScript
  - AJAX
- "Real" work environment
- Practice current technologies
  - Basics
  - Advanced technologies in "Web Technologies II" course

Prerequisites

- Linux
- bash
- WebStorm
- git, gitHub
Q&A

Can I practice at home?
Yes. You will learn it in the 1st tutorial.

I don’t know a technology. What can I do?
You should have learned about each technology. Required technologies will be summarized in tutorials.

I have got a problem with something. How can I solve it?
Troubleshooting checklist

1. Read the error message
2. Google it
3. Ask your
   1. friends
   2. classmates
   3. tutor