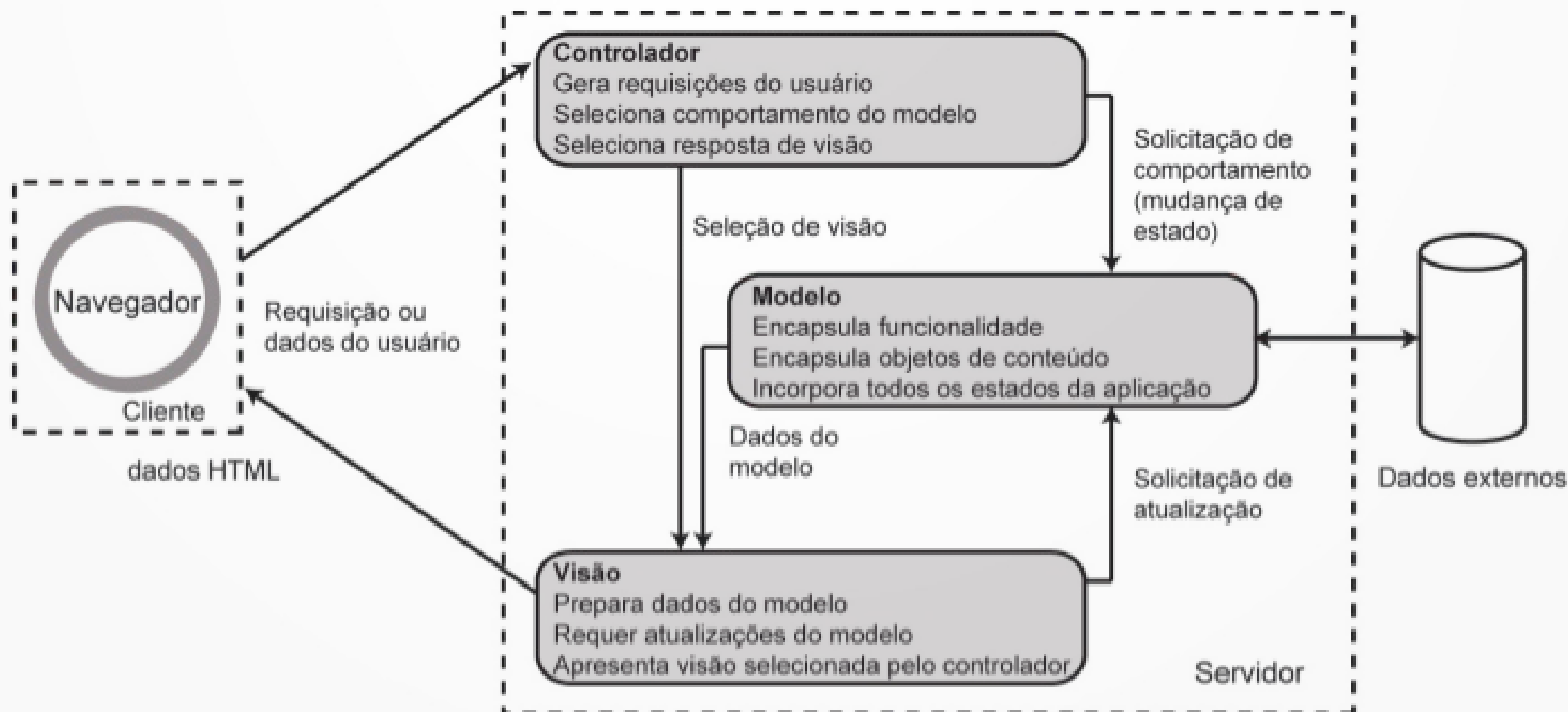


# DESENVOLVIMENTO DE SOFTWARE PARA A WEB 1

Módulo 5  
MVC: Servlet, JSP, JSTL & JDBC

**Delano Medeiros Beder**  
**delano@dc.ufscar.br**

# Modelo – Visão – Controlador



# Modelo – Visão – Controlador

- Model 2 (MVC simples para web)
  - Modelo: *Value Objects (POJOs)* e *DAOs*. Apenas se preocupam com as informações e seus relacionamentos
  - Visão: páginas JSP e formulários. Apenas se preocupam com aspectos de interface, como renderização de informações e validação
  - Controle: *Servlets* e classes Java que fazem a escolha de quais visões exibir, quais classes executar, implementando o fluxo da aplicação conforme as interações do usuário
- Importante: nenhum conteúdo é gerado nos *Servlets*!

# Modelo: Padrão DAO

- *Data Access Object* (ou simplesmente DAO) é um padrão de projeto para persistência de dados que permite separar regras de negócio das regras de acesso a banco de dados.
- Em uma aplicação que utilize a arquitetura MVC, todas as funcionalidades de banco de dados, tais como obter as conexões, mapear objetos Java para tipos de dados SQL ou executar comandos SQL, devem ser feitas por classes DAO.

# Modelo: Padrão DAO

```
abstract public class GenericDAO {  
  
    public GenericDAO() {}  
  
    protected Connection getConnection() throws SQLException {  
        String url = "jdbc:mysql://localhost:3306/Livraria";  
        return DriverManager.getConnection(url, "root", "root");  
    }  
}
```

Classe Abstrata: raiz hierarquia *DAOs*

Classe Concreta LivroDAO

Entidade Livro

```
public class LivroDAO extends GenericDAO {  
  
    public void insert(Livro livro) {  
        String sql = "INSERT INTO Livro (titulo, autor, ano, preco, editora_id) VALUES (?, ?, ?, ?, ?)";  
  
        try {  
            Connection conn = this.getConnection();  
            PreparedStatement statement = conn.prepareStatement(sql);  
            statement.setString(1, livro.getTitulo());  
            statement.setString(2, livro.getAutor());  
            statement.setInt(3, livro.getAno());  
            statement.setFloat(4, livro.getPreco());  
            statement.setLong(5, livro.getEditora().getId());  
            statement.executeUpdate();  
  
            statement.close();  
            conn.close();  
        } catch (SQLException e) {  
            throw new RuntimeException(e);  
        }  
    }  
  
    public List<Livro> getAll() {}  
    public void delete(Livro livro) {}  
    public void update(Livro livro) {}  
    public Livro get(Long id) {}  
}
```



# Modelo: Entidades (POJOs)

```
public class Livro {  
  
    private Long id;  
    private String titulo;  
    private String autor;  
    private Integer ano;  
    private Float preco;  
    private Editora editora;  
  
    public Livro(Long id) {}  
    public Livro(String titulo, String autor, Integer ano, Float preco, {}  
    public Livro(Long id, String titulo, String autor, Integer ano, {}  
  
    public Long getId() {}  
    public void setId(Long id) {}  
    public String getTitulo() {}  
    public void setTitulo(String titulo) {}  
    public String getAutor() {}  
    public void setAutor(String autor) {}  
    public Integer getAno() {}  
    public void setAno(Integer ano) {}  
    public Float getPreco() {}  
    public void setPreco(Float preco) {}  
    public Editora getEditora() {}  
    public void setEditora(Editora editora) {}  
}
```

Entidade Livro

Plain Old Java Objects (POJO)

Construtores

Métodos *getters/setters*

# Uma aplicação completa (MVC)

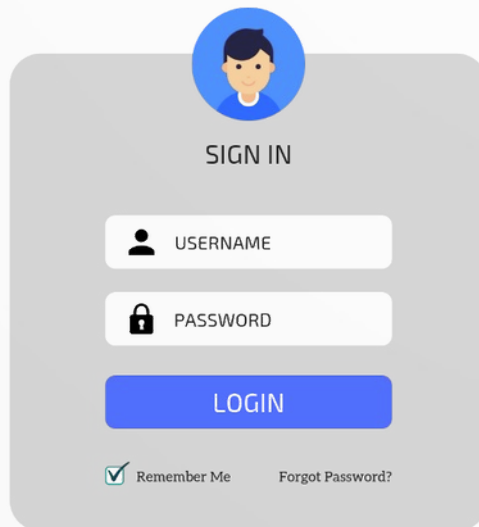
## Demonstração 1

Livraria: CRUD Livros


CRUD: **C**reate, **R**ead, **U**ppdate & **D**elelete

# Autenticação x Autorização

- A autenticação verifica a identidade digital do usuário, ou seja, processo de verificação de uma identidade. Em termos mais simples, é quando o usuário prova de fato quem ele é.
- Por sua vez, a autorização é o processo que ocorre após ser validada a autenticação. Diz respeito aos privilégios que são concedidos a determinado usuário ao utilizar uma aplicação.



The illustration shows a login interface. At the top is a circular profile icon of a person. Below it is the text 'SIGN IN'. There are two input fields: the first is labeled 'USERNAME' with a person icon, and the second is labeled 'PASSWORD' with a lock icon. Below these is a blue button labeled 'LOGIN'. At the bottom, there is a 'Remember Me' checkbox which is checked, and a link labeled 'Forgot Password?'.



The illustration shows an access control dashboard. At the top is a circular profile icon of a person. Below it is the text 'CONTROLE DE ACESSO'. There is a section for user information: a card showing '123.456.789-10', 'Maria Antônia Silva Santos', and 'Financeiro'. Below this is a list of permissions with checkboxes: 'Contas a Pagar' (checked), 'Contas a Receber' (checked), 'Controle de Estoque' (locked), 'Cadastro de Usuários' (locked), 'Cadastro de Clientes' (locked), 'Vendas' (locked), and 'Pedidos' (locked).



# Uma aplicação completa (MVC)

## Demonstração 2

Autenticação/Autorização de Usuários

# O que é AJAX

**Asynchronous JavaScript and XML.**

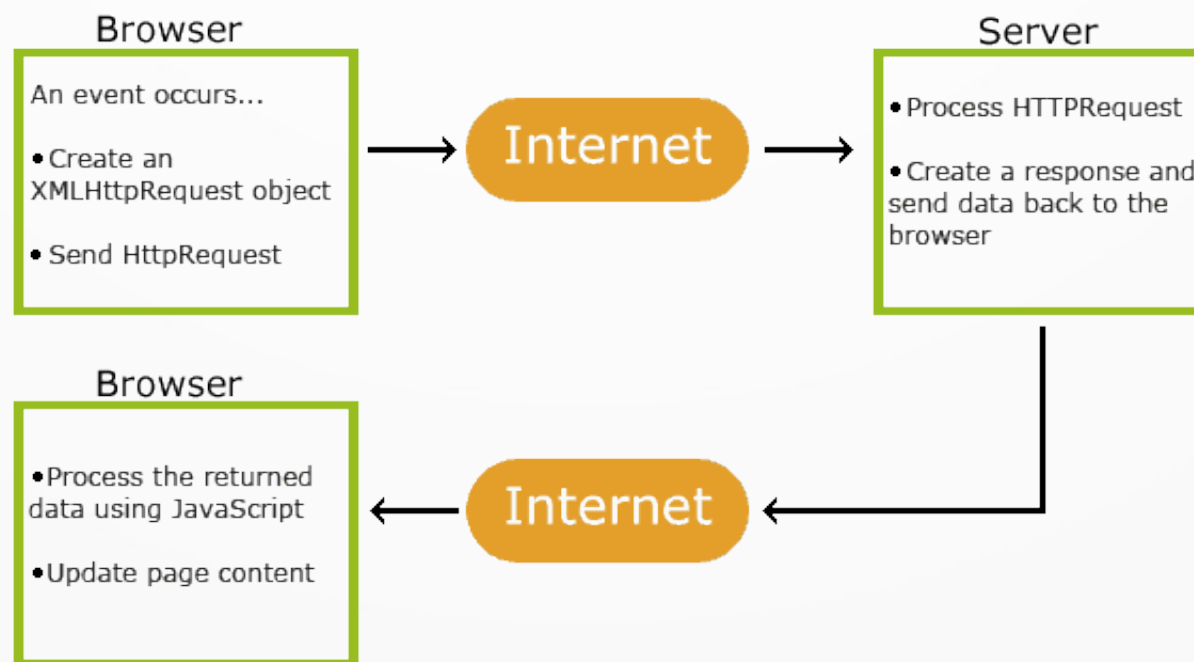
AJAX não é uma linguagem de programação, mas uma nova maneira de usar padrões existentes.

AJAX é a arte de trocar dados com um servidor e atualizar partes de uma página web sem ter que recarregar toda a página, assim criando páginas rápidas e dinâmicas.

# XMLHttpRequest

O objeto de JavaScript para realizar requisições sem recarregar a página. Um dos parâmetros que recebe pra abrir uma conexão (método `open()`) é um boolean indicando se a requisição é síncrona ou assíncrona

Requisições síncronas bloqueiam (travam) o navegador até a requisição concluir. Se a requisição for assíncrona o navegador não é bloqueado e o cliente pode continuar navegando enquanto a requisição é realizada: isso é AJAX



# Exemplo [https://www.w3schools.com/js/tryit.asp?filename=tryjs\\_ajax\\_first](https://www.w3schools.com/js/tryit.asp?filename=tryjs_ajax_first)

```
<!DOCTYPE html>
<html>
<head>
<script>
function loadDoc() {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
            var demo = document.getElementById("demo");
            demo.innerHTML = this.responseText;
        }
    };
    xhttp.open("GET", "ajax_info.txt", true);
    xhttp.send();
}
</script>
</head>

<body>

<div id="demo">
<h2>The XMLHttpRequest Object</h2>
<button type="button" onclick="loadDoc()">Change Content</button>
</div>

</body>
</html>
```

# Uma aplicação completa (MVC)

## Demonstração 3

AJAX (Asynchronous Javascript and XML)



**FIM**