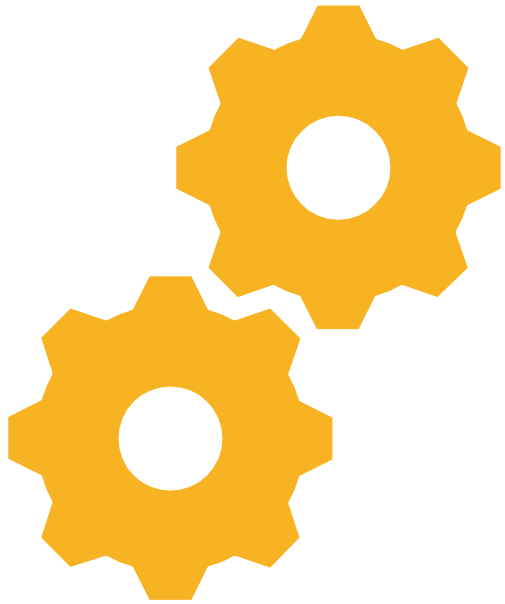
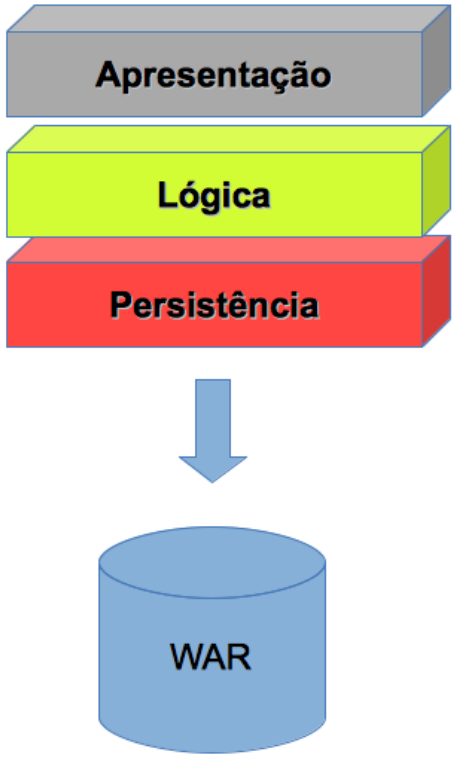


MVC, JSON E XML

PROF. ME. HÉLIO ESPERIDIÃO

Aplicação monolítica



A aplicação é um grande mononíto. Mantida por uma única equipe. Distribuída como um todo

APLICAÇÕES MONOLÍTICAS

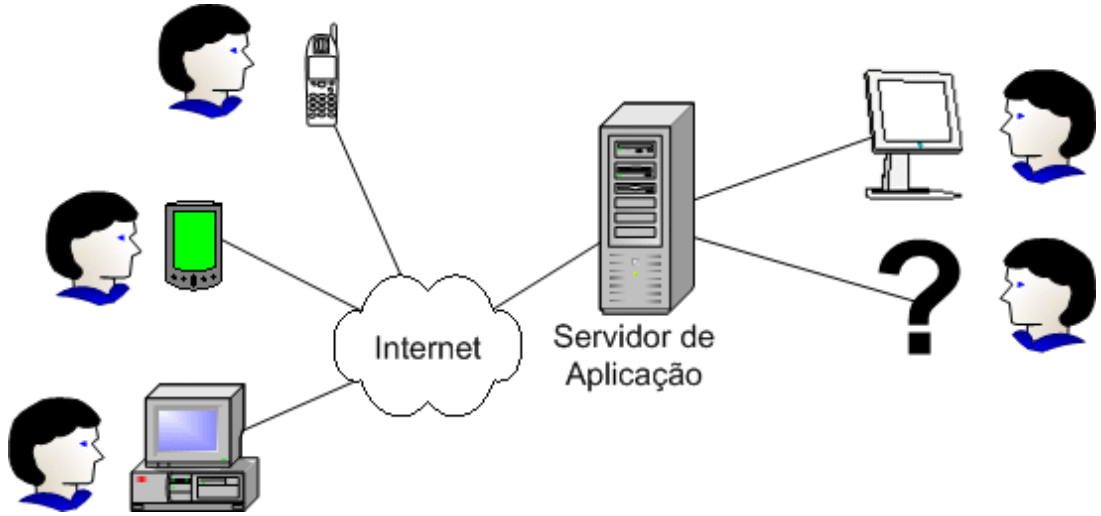
- Na época dos computares independentes um aplicativo era desenvolvido para ser usado em uma única máquina.
-
- Este aplicativo continha todas a funcionalidades em um único módulo gerado por uma grande quantidade de linhas de código e de manutenção nada fácil.
- As entradas do usuário , verificação , lógica de negócio e acesso a banco de dados estava presente em um mesmo lugar.

APLICAÇÕES EM DUAS CAMADAS

Cliente-servidor é um modelo computacional que separa clientes e servidores, sendo interligados entre si geralmente utilizando-se uma rede de computadores.

Cada instância de um cliente pode enviar requisições de dado para algum dos servidores conectados e esperar pela resposta.

O servidor disponível pode aceitar tais requisições, processá-las e retornar o resultado para o cliente.



APLICAÇÕES EM TRÊS CAMADAS

- Camada de apresentação (UI)
- Camada de aplicação (business logic)
- Camada de dados

HOW TO USE MODEL-VIEW-CONTROLLER

- Em 1979 nasce o padrão de projeto MVC.
- A implementação original foi descrita no artigo "**Applications Programming in Smalltalk-80: How to use Model-View-Controller**".

- A ideia de Reenskaug gerou um padrão de arquitetura de aplicação cujo objetivo é separar o projeto em três camadas independentes, que são o modelo, a visão e o controlador.

- Essa separação de camadas ajuda na redução de acoplamento e promove o aumento de **coesão** nas classes do projeto.

ACOPLAMENTO

- É o grau em que uma classe conhece a outra.
- Se o conhecimento da classe A sobre a classe B for através de sua interface, temos um baixo acoplamento, e isso é bom.
- Por outro lado, se a classe A depende de membros da classe B que não fazem parte da interface de B, então temos um alto acoplamento, o que é ruim.

COESÃO

- Uma classe deve ter apenas uma única responsabilidade e realizá-la de maneira satisfatória.

- Quando temos uma classe elaborada de forma que tenha um único propósito, dizemos que ela tem uma **alta coesão**;

- Quando temos uma classe com propósitos que não pertencem apenas a ela, temos uma **baixa coesão**;.

MODEL

- Sempre que você pensar em manipulação de dados, pense em model. Ele é **responsável** pela **leitura** e **escrita** de **dados,** e também de suas **validações**.

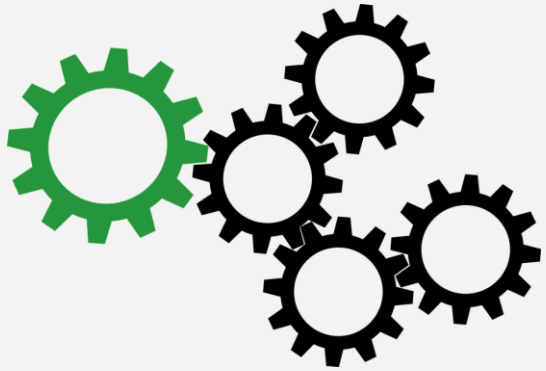
- O modelo (Model) é utilizado para manipular informações de forma mais detalhada, sendo recomendado que, sempre que possível, se utilize dos modelos para realizar consultas, cálculos e todas as regras de negócio do nosso site ou sistema. É o modelo que tem acesso a toda e qualquer informação sendo essa vinda de um banco de dados, arquivo XML.
- É a camada que contem a estrutura de dado atrás de uma parte específica da aplicação
- Usualmente portada em JSON.
- Responsável pela leitura manipulação e validação de dados, e também de suas validações.
- Responsável por tratar as regras de negócio.
- Obtém os dados e os traduz em informações relevantes para serem exibidas pela View.
- Notifica a view e controler associados quando há uma mudança em seu estado.

VIEW

- Simples: a camada de interação com o usuário. Ela apenas faz a **exibição** dos **dados**, sendo ela por meio de um **html** ou **xml**.

- A visão (view) é responsável por tudo que o usuário final visualiza, toda a interface, informação, não importando sua fonte de origem, é exibida graças a camada de visão.
- É a camada que exibe uma representação dos dados.
- É camada de interface com usuário (view).
- Também conhecida como cliente-side.
- Faz a exibição dos dados, utilizando-se de #HTML e/ou XML.
- É responsável por usar as informações modeladas para produzir interfaces de apresentação conforme a necessidade.

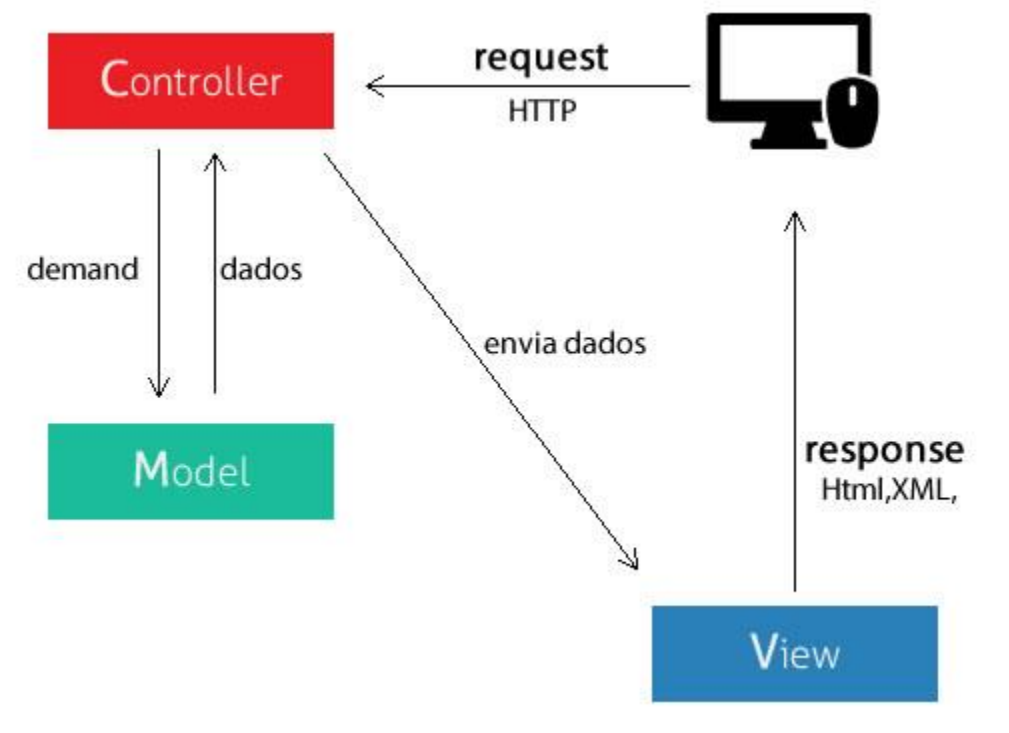
CONTROLLER

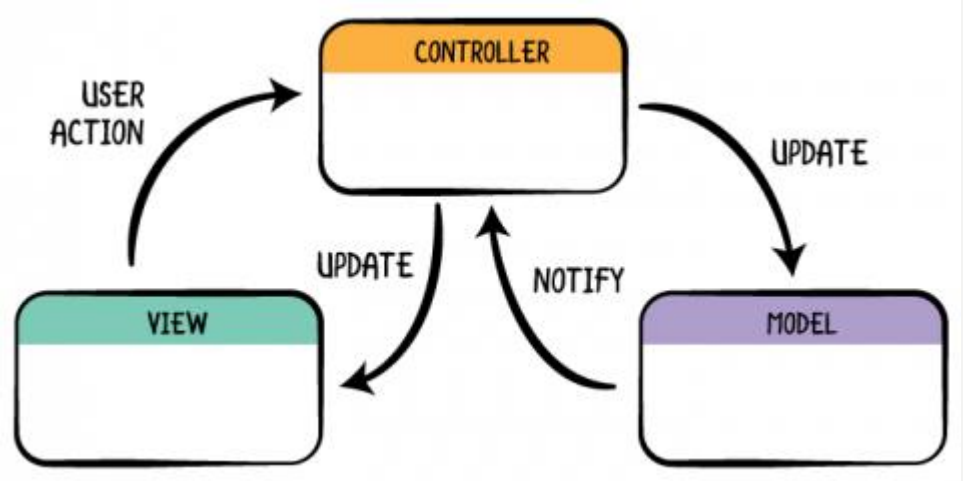


- O responsável por **receber** todas as **requisições** do **usuário**. Seus métodos chamados actions são responsáveis por uma página, controlando qual model usar.
- Controlar todo o fluxo de informação que passa pelo site/sistema. Decide "se", "o que", "quando" e "onde" deve funcionar.

- Define quais informações devem ser geradas, quais regras devem ser acionadas e para onde as informações devem ir, é na controladora que essas operações devem ser executadas. Em resumo, é a controladora que executa uma regra de negócio (modelo) e repassa a informação para a visualização (visão).

APLICAÇÃO WEB E DESKTOP





POR QUE UTILIZAR MVC?

- Aplicações estão cada vez mais complexas e qualquer tipo de alterações em uma das camadas não interfere nas demais.

- Facilita a atualização de layouts, alteração nas regras de negócio e adição de novos recursos.

- Em caso de grandes projetos, o MVC facilita muito a divisão de tarefas entre a equipe.

Facilita o reaproveitamento de código;

- Facilidade na manutenção e adição de recursos;
- Maior integração da equipe e/ou divisão de tarefas;
- Diversas tecnologias estão adotando essa arquitetura;
- Facilidade em manter o seu código sempre limpo;

JSON - XML

PROF. ME.
HÉLIO
ESPERIDIÃO

JSON - JAVASCRIPT OBJECT NOTATION

- É uma formatação leve de troca de dados.
- Para seres humanos, é fácil de ler e escrever.
- Para máquinas, é fácil de interpretar e gerar.
- É baseado em um subconjunto da linguagem de programação JavaScript, Standard ECMA-262 3a Edição -Dezembro - 1999.

JSON

- JSON é em formato texto e completamente independente de linguagem
- Formato ideal de troca de dados
- é um formato compacto, de padrão aberto independente, de troca de dados simples e rápida (parsing) entre sistemas

ESTRUTURA

- Uma coleção de pares nome/valor. Em várias linguagens, isto é caracterizado como um *object*, record, struct, dicionário, hash table, keyed list, ou arrays associativas.

- Uma lista ordenada de valores. Na maioria das linguagens, isto é caracterizado como uma *array*, vetor, lista ou sequência.

EXEMPLO - JAVASCRIPT

```
var myObj = { "name":"John", "age":31, "city":"New York" };
var myJSON = JSON.stringify(myObj);
window.location = "demo_json.php?x=" + myJSON;
```

EXEMPLO - JAVASCRIPT

```
var myJSON = '{ "name":"John", "age":31, "city":"New York" }';
var myObj = JSON.parse(myJSON);
document.getElementById("demo").innerHTML = myObj.name;
```

EXEMPLO ARRAY

```
<script>

var myObj, x,y;
myObj = {
    "name":"John",
    "age":30,
    "cars":[ "Ford", "BMW", "Fiat" ]
};
x = myObj.name;
y=myObj.cars[0];
document.getElementById("demo").innerHTML = x + " - " +y;

</script>
```

```
<script>

var myObj, i, j, x = "";
myObj = {
    "name":"John",
    "age":30,
    "cars": [
        { "name":"Ford", "models":[ "Fiesta", "Focus", "Mustang" ] },
        { "name":"BMW", "models":[ "320", "X3", "X5" ] },
        { "name":"Fiat", "models":[ "500", "Panda" ] }
    ]
}

for (i in myObj.cars) {
    x += "<h2>" + myObj.cars[i].name + "</h2>";
    for (j in myObj.cars[i].models) {
        x += myObj.cars[i].models[j] + "<br>";
    }
}


document.getElementById("demo").innerHTML = x;

</script>
```

EXEMPLO PHP

```php
<?php
$myObj->name = "John";
$myObj->age = 30;
$myObj->city = "New York";

$myJSON = json_encode($myObj);

echo $myJSON;
?>
```

```php
<?php
$myArr = array("John", "Mary", "Peter", "Sally");

$myJSON = json_encode($myArr);

echo $myJSON;
?>
```

```php
<?php
$json = '{"a":1,"b":2,"c":3,"d":4,"e":5}';

var_dump(json_decode($json));
var_dump(json_decode($json, true));

?>
```

XML (*EXTENSIBLE MARKUP LANGUAGE*)

- A principal característica do XML, de criar uma infraestrutura única para diversas linguagens, é que linguagens desconhecidas e de pouco uso também podem ser definidas sem maior trabalho e sem necessidade de ser submetidas aos comitês de padronização.

- O XML é um formato para a criação de documentos com dados organizados de forma hierárquica, como se vê, frequentemente, em documentos de texto formatados, imagens vetoriais ou bancos de dados.

- Separação do conteúdo da formatação
- Simplicidade e legibilidade, tanto para humanos quanto para computadores
- Possibilidade de criação de tags sem limitação
- Criação de arquivos para validação de estrutura (chamados DTDs)
- Interligação de bancos de dados distintos
- Concentração na estrutura da informação, e não na sua aparência

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<receita nome="pão" tempo_de_preparo="5 minutos" tempo_de_cozimento="1 hora">
  <titulo>Pão simples</titulo>
  <ingredientes>
    <ingrediente quantidade="3" unidade="xícaras">Farinha de Trigo</ingrediente>
    <ingrediente quantidade="7" unidade="gramas">Fermento</ingrediente>
    <ingrediente quantidade="1.5" unidade="xícaras" estado="morna">Água</ingrediente>
    <ingrediente quantidade="1" unidade="colheres de chá">Sal</ingrediente>
  </ingredientes>
  <instrucoes>
    <passo>Misture todos os ingredientes, e dissolva bem.</passo>
    <passo>Cubra com um pano e deixe por uma hora em um local morno.</passo>
    <passo>Misture novamente, coloque numa bandeja e asse num forno.</passo>
  </instrucoes>
</receita>
```

```
<script>
var parser, xmlDoc;
var text = "<bookstore><book><title>Everyday Italian</title><author>Giada De Laurentiis</author>" +
"<year>2005</year></book></bookstore>";

parser = new DOMParser();
xmlDoc = parser.parseFromString(text,"text/xml");

document.getElementById("demo").innerHTML =
xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;
</script>
```

JSON VERSUS XML

- O JSON pode ser considerado concorrente da XML na área de troca de informações. Vejamos algumas das principais semelhanças e diferenças entre os modelos de marcação das informações:

SEMELHANÇAS

- Representam informações no formato texto.;
- Ambos podem ser utilizados para transportar informações em aplicações AJAX.
- Ambos são independentes de linguagem.

- Dados representados em XML e JSON podem ser acessados por qualquer linguagem de programação, através de API's específicas.

DIFERENÇAS:

- Json Não é uma linguagem de marcação. (Não possui tags de abertura e de fechamento);
- Json É tipicamente destinado para a troca de informações, enquanto XML possui mais aplicações. Por exemplo: existem bancos de dados no formato XML e estruturados em SGBD XML nativo.