UPT UNIVERSIDADE PORTUCALENSE

CONTEÚDO

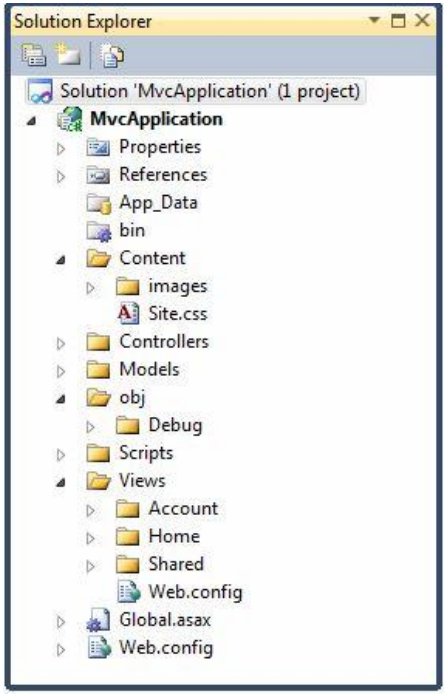
1. Model-view-controller
2. Estrutura
3. Forças e Oportunidades | Fraquezas e Ameaças
4. Benefícios
5. Por que usar o MVC?

Model-view-controller

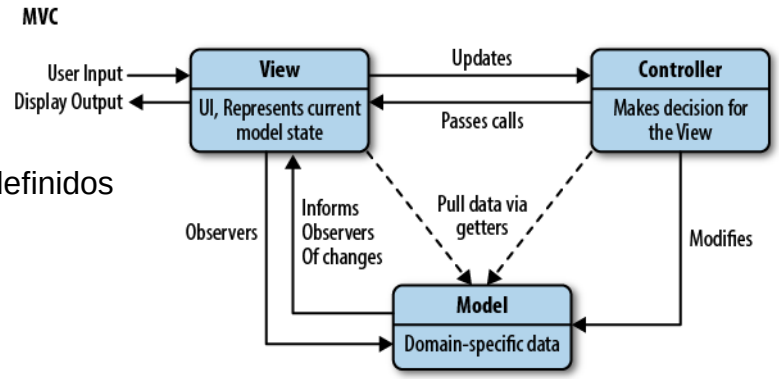
- Padrão de desenho de software

- Finalidade de isolar: regras de negócio | lógica de apresentação | interface | utilizador
  - Permite existência de várias interfaces que podem ser alteradas sem alterar as regras de negócio

- Proporciona
  - Maior flexibilidade
  - Mais oportunidades de reutilização de código



Estrutura



Possibilita divisão do projeto em camadas bem definidas, com objetivos bem definidos

- ***Model*** (modelo): dados da aplicação, regras de negócio, lógica, funções.
  - Gere o processo de negócio
  - Responde a pedidos do controlador
  - Apresenta resultados na *View*

- ***View*** (vista): qualquer saída de representação de dados (ex: tabela, diagrama, …)

- ***Controller*** (controlador):
  - Mediação da entrada, convertendo-a em comandos para o *Model* ou *View*
  - Baseado em comportamentos
  - Pode ser partilhado por várias *Views*
  - Responsável por determinar exibição na *View*

Forças e Oportunidades | Fraquezas e Ameaças

- Separação: código do lado do servidor / código do lado do cliente, lógica de negócio (*model*) / vista (*view*)
- Robustez, reutilização e organização de código
- Múltiplos acessos e pontos de vista para o mesmo modelo
- Facilidade de acesso (independente do tipo de interface)
- *Model* isola e trata gestão de estados e persistência de dados
- Responsável pelo desenho da arquitetura terá grande impacto na solução final
- Facilidade de permuta entre camada de dados e regras de negóco
- *Model* é autossuficiente e separado do *controller* e da *view*
- Controller *usado para* unir *view* e *model* para atender a um pedido
- Redução do número de erros na camada lógica

- Cada classe deve estar bem definida
- Nível elevado de complexidade (todos os detalhes contam)
- Necessidade de perceber como cada parte da aplicação interage com as outras
- Separação rigorosa entre o *model* e a *view* (pode tornar o *debugging* mais difícil)
- Necessidade de repensar a aplicação e de colocar grande parte do esforço na arquitetura
- Problemas de segurança (se o MVC for mal implementado)

Benefícios

- Aumento da produtividade / redução do tempo de desenvolvimento
- Uniformidade na estrutura do software
- Redução da complexidade do código
- Manutenção e documentação da aplicação
- Estabelecimento de um vocabulário comum para o projeto entre os vários programadores
- Reutilização de módulos do sistema noutros sistemas
- Utilização de um conjunto de padrões para resolver problemas maiores
- Suporte à construção de software confiável com arquiteturas já testadas
- Possibilidade de reescrita da interface com o utilizador ou do *controller* sem alterar o modelo base
- Reutilização da interface para diferentes aplicações com pouco esforço
- Facilidade na manutenção e adição de recursos
- Reaproveitamento de código e facilidade de o manter sempre limpo

Por que usar o MVC?

Deve utilizar-se o MVC sempre que se verificar uma das seguintes situações

- A aplicação precisa de uma ligação assíncrona para o back-end
- A aplicação tem uma funcionalidade que não resulta num refresh completo da página
- Grande parte da visualização ou manipulação de dados está no browser e não no servidor
- Os mesmos dados são processados de formas diferentes na página
- A aplicação tem muitas interações que modificam os dados
- Uma escolha incorreta pode levar à reimplementação de uma funcionalidade já implementada por uma framework

UNIVERSIDADE
PORTUCALENSE

Do conhecimento à prática.