UNIVERSIDADE PORTUCALENSE

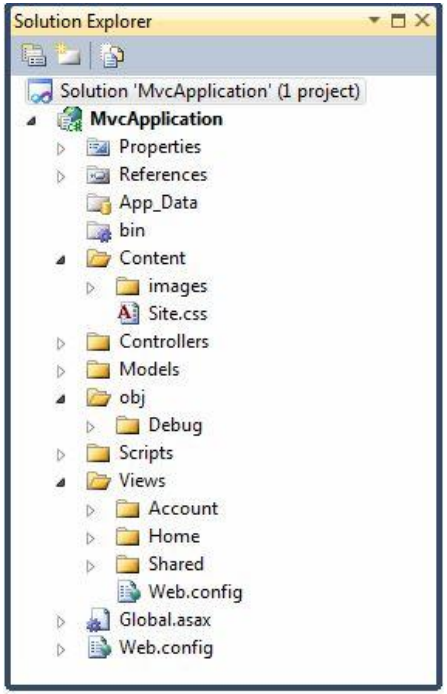
CONTENT

1. Model-view-controller
2. Structure
3. Strengths and Opportunities | Weaknesses and Threats
4. Benefits
5. Why use MVC?

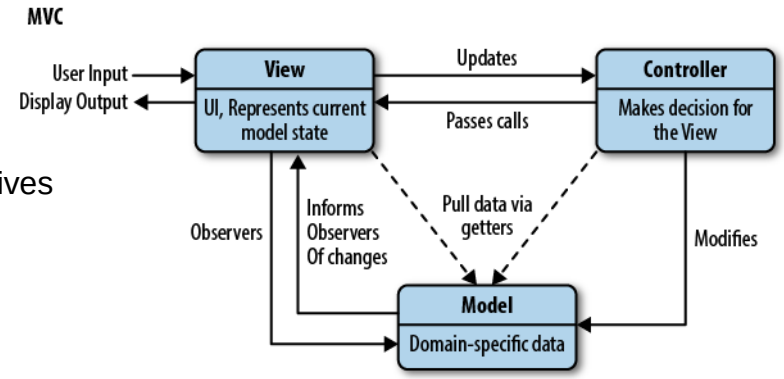
Model-view-controller

- Software design pattern

- Aims to isolate: business rules | display logic | interface | user
  - Allows the existence of several interfaces that can be altered without changing the business rules
- Allows
  - Higher flexibility
  - More code reusing opportunities



Structure



Allows dividing a project into well-defined layers, each with well-defined objectives

- ***Model***: application data, business rules, logic, functions
  - Manages the business process
  - Responds to controller requests
  - Shows the results on the *View*

- ***View***: any data representation (ex: table, diagram, …)

- ***Controller***:
  - Mediates the input, converting it to commands to the *Model* or the *View*
  - Based on behaviour
  - May be shared by several *Views*
  - Determines the exhibition on the *View*

Strengths and Opportunities | Weaknesses and Threats

- Separation: server side code / client side code , business logic (*model*) / view (*view*)
- Code robustness, reutilization and organization
- Multiple access and view points for the same model
- Easy access (independently of the type of interface)
- *Model* isolates and handles state management and data persistence
- Person in charge of architecture design will have great impact on the final solution
- Easy to switch between data layers and business rules
- *Model* is independent and separated from *controller* and *view*
- Controller used to join *view* and *model* to answer a request
- Less errors on the logical layer

- Each class must be well defined
- High level of complexity (all the details count)
- Necessity of understanding how the parts interact
- Separation between *model* and *view* (debugging may become more difficult)
- Necessity to rethink the application an put much effort into the architecture
- Security problems (if MVC is badly implemented)

Benefits

- Increase productivity / reduce time needed for development
- Uniform software structure
- Reduce code complexity
- Application maintaining and documenting
- Establishment of common wording for all the programmers
- Reusing system modules in other systems
- Using a set of patterns to solve bigger problems
- Supports building trustable software with already tested architectures
- Possibility of rewriting user interface or *controller* without changing the base model
- Reusing the interface for different applications with low effort
- Easy to maintain and add resources
- Reuse code and easy to maintain the code “clean”

Why use MVC?

We should use MVC if:

- The application needs an asynchronous connection to the back-end
- The application has a functionality that does not imply a refresh to the page
- Most of the data visualization and manipulation is made on the browser instead of the server
- The same data can be processed in different ways on the page
- The application has many interactions that change the data
- An incorrect choice may lead to reimplementing a functionality that is already implemented by a framework

UNIVERSIDADE
PORTUCALENSE

Do conhecimento à prática.