CSE417: WEB ENGINEERING

Daffodil International University

LEARNING OUTCOMES

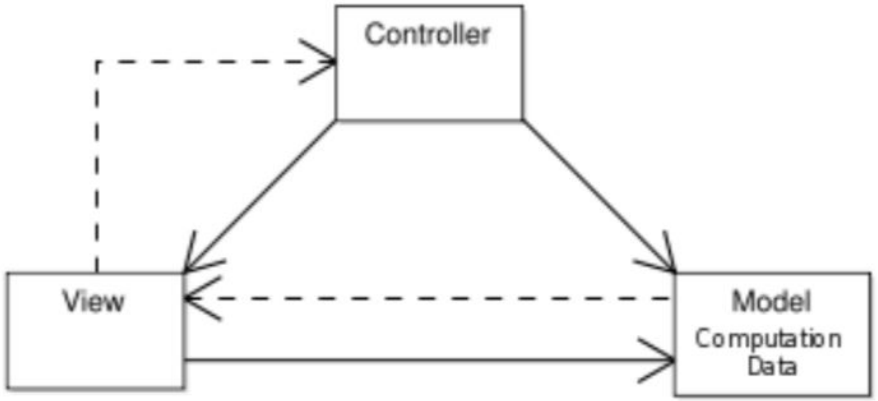
- ✔ You will know MVC design pattern
- ✔ You will be able to apply Project management techniques

DISTRIBUTED SYSTEMS: FUNDAMENTAL QUESTIONS

- Software developers have to consider a wide, but rather stable, range of questions including:
  - Where can or should computations take place?
  - Where can or should data be stored?
  - How fast can data be transferred/communicated?
  - What is the cost of data storage/computations/communication depending on how/where we do it?
  - How robustly/securely can data storage/computations/communication be done depending on how/where we do it?
  - How much energy is available to support data storage/computations/communication depending on how/where we do it?
  - What is the legality of data storage/computations/communications depending on how/where we do it?
- The possible answers to each of these questions is also rather stable, but the 'right' answers change

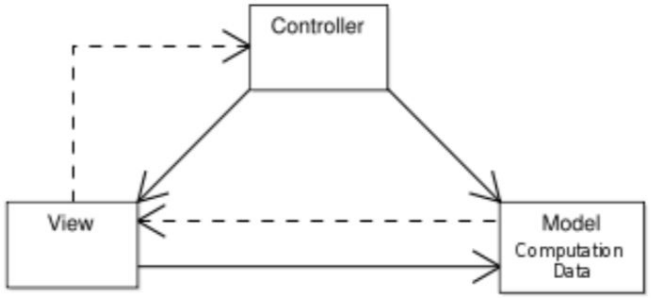
DISTRIBUTED SYSTEMS: MVC(1/3)

- We use the Model-View-Controller(MVC) software design pattern to discuss some of these questions in more detail:
  - The model manages the behavior and data
  - The view renders the model into a form suitable for interaction
  - The controller receives user input and translates it into instructions for the model



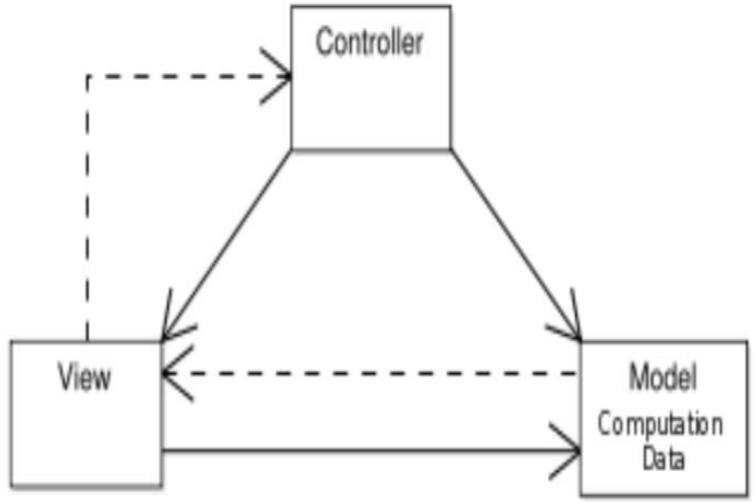
DISTRIBUTED SYSTEMS: MVC(2/3)

- Where should the view be rendered?
  - On the user's computer
  - On a central server (farm) possibly shared by a multitude of users
- Where should the behaviour of the model be computed?
  - Close to the user,
    - on a single computer exclusively used by the user
  - Away from the user,
    - on a central server (farm) shared by a multitude of users
  - Distributed,
    - on several computers owned by a large group of users



DISTRIBUTED SYSTEMS: MVC(3/3)

- Where should the data for the model be held?
  - Close to the user,
    - on a single computer exclusively used by the user
  - Away from the user,
    - on a central server (farm) shared by a multitude of users
  - Distributed,
    - on several computers owned by a large group of users



DISTRIBUTED SYSTEMS: FUNDAMENTAL QUESTIONS

- Software developers have to consider a wide, but rather stable, range of questions
- The possible answers to each of these questions is also rather stable
- The 'right' answer to each these questions will depend on
  - the domain in which the question is posed
  - available technology
  - available resources
- The 'right' answer to each of the questions changes over time
- We may go back and forth between the various answers
- The reasons for that are not purely technological, but includes
  - legal factors
  - social factors
  - economic factors

A BASIC EXAMPLE

In order for a program to get data from a database, it has to undergo a list of actions:

1. Connect to the database server
2. Select a database
3. Query the database
4. Fetch the data
5. Use the Data

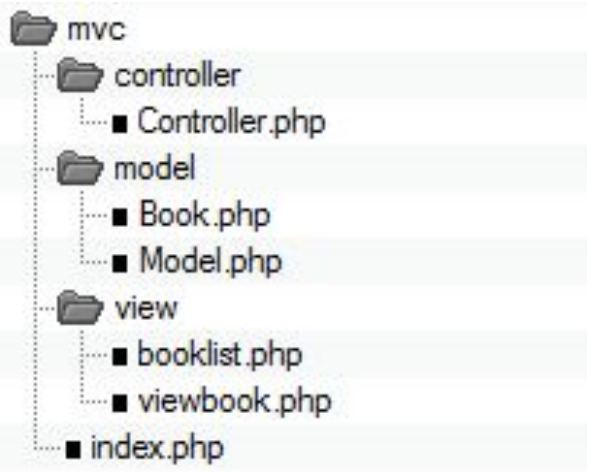
A **framework** may handle steps 1-4 for you, so that your responsibilities are reduced to:

6. Tell the framework to fetch the data
7. Use the data

- There are many frameworks that use MVC design pattern.
  - Laravel, CodeIgniter, Symfony, CakePHP etc. are few of them

EXAMPLE

directory structure



model/Book.php

```php
<?php
    class Book {
        public $title;
        public $author;
        public $description;
        public function __construct($title, $author, $description){
            $this->title = $title;
            $this->author = $author;
            $this->description = $description;
        }
    }
?>
```

EXAMPLE

model/Model.php

```php
<?php
include_once("model/book.php");
class Model {
    public function getBookList(){
        // instead of these values(harcoded),
        //we use SQL queries to output data
        $example_data = array( "Jungle Book" => new Book("Jungle Book", "R. Kipling", "A classic book."),
                               "Moonwalker" => new Book("Moonwalker", "J. Walker", ""),
                                "Harry Potter" => new Book("Harry Potter", "J.K.Rowling", "Fantasy"));
        return $example_data;
    }
    public function getBook($title){
        //We used getBookList() function to manipulate data
        //Again, we use a real database to do this operations here
        $allBooks = $this->getBookList();
        return $allBooks[$title];  }
    }
?>
```

EXAMPLE

view/abook.php

```
<html>
<head></head>
<body>
    <?php

        echo 'Title:' . $book->title . '<br/>';
        echo 'Author:' . $book->author . '<br/>';
        echo 'Description:' . $book->description . '<br/>';
    ?>
</body>
</html>
```

EXAMPLE

view/booklist.php

```
<html>
<head></head>
<body>
    <table>
        <tr>
            <td>Title</td>
            <td>Author</td>
            <td>Description</td>
        </tr>
        <?php
            foreach ($books as $book){
                echo '<tr>
                        <td><a href="index.php?book='.$book->title.'">'.$book->title .'</a></td>
                        <td>'.$book->author .'</td>
                        <td>'.$book->description.'</td>
                        </tr>';
            }
        ?>
    </table>
</body>
</html>
```

EXAMPLE

controller/Controller.php

```php
<?php
include_once("model/Model.php");

class Controller {
    public $model;
    public function __construct(){
        $this->model = new Model();
    }

    public function invoke(){
        if (!isset($_GET['book'])){
            //no specific book is requested,
            //show all available books
            $books = $this->model->getBookList();
            include 'view/booklist.php';
        }else{
            //show the requested book
            $book = $this->model->getBook($_GET['book']);
            include 'view/abook.php';
        }
    }
}
?>
```

index.php

```php
<?php
    include_once("controller/Controller.php");
    //interactions start at index
    //forwarded to controller
    $controller = new Controller();
    $controller->invoke();
?>
```

Remember, everything starts at index.php

Apply this basic pattern in your project!

PROJECT MANAGEMENT

- **Deliver on time and on schedule and in accordance with the requirements**
- **Budget and schedule constraints**
- The product is **intangible**
  - Cannot be seen or touched.
  - Project **managers can not see progress by simply looking at the artifact**
- Many software projects are 'one-off' projects
    - Large software projects are usually different from previous projects
    - Experience does not help.
- Software processes are variable and organization specific
    - cannot predict when a process is likely to lead to development

PROJECT MANAGEMENT ACTIVITIES(1/2)

- Project planning
  - Project managers are responsible for planning. estimating and scheduling project development and assigning people to tasks.
- Reporting
  - Project managers are usually responsible for reporting on the progress of a project to customers and to the managers of the company developing the software.
- Risk management
  - Project managers assess the risks that may affect a project, monitor these risks and take action when problems arise.

PROJECT MANAGEMENT ACTIVITIES(2/2)

- People management
  - Project managers have to choose people for their team and establish ways of working that leads to effective team performance.
- Proposal writing
  - The first stage in a software project may involve writing a proposal to win a contract to carry out an item of work. The proposal describes the objectives of the project and how it will be carried out.

PROJECT MANAGEMENT CHALLENGES

- **Unique software systems:** the experience from the past project is too little to be able to make reliable cost estimates.
- **Extremely technical leadership perspective:** dominated by technology freaks.
- **Poor planning:** many projects are characterize by unclear or incomplete planning objectives, frequent changes to planning objectives, defects in project organization.
- **Development Challenges**
  - Individuality of programmers
  - High number of alternative solutions
  - Rapid technological change
- **Monitoring Challenges**
  - The immaterial state of software products

- **EXERCISE**
  - What are the drawbacks and benefits of using MVC design pattern?
  - Apply MVC while structuring your project
  - Write briefly about obstacles faced in your project
- **READINGS**
  - https://developer.chrome.com/apps/app_frameworks
  - https://www.guru99.com/mvc-tutorial.html

ACKNOWLEDGEMENT

- This module is designed and created with the help from following sources-
  - https://cgi.csc.liv.ac.uk/~ullrich/COMP519/
  - http://www.csc.liv.ac.uk/~martin/teaching/comp519/
  - Materials of MVC, S Rahman Shammi, Daffodil International University