Laravel Framework for Web Applications

Laravel Framework for Web Applications

Trainer's Guide

APTECH LIMITED

Contact E-mail: ov-support@onlinevarsity.com

First Edition - 2019

Aptech®
Unleash your potential

Aptech LEARNING®
Unleash your potential

Dear Learner,

We congratulate you on your decision to pursue an Aptech Worldwide course.

Aptech Ltd. designs its courses using a sound instructional design model – from conceptualization to execution, incorporating the following key aspects:

- **Scanning the user system and needs assessment**

Needs assessment is carried out to find the educational and training needs of the learner.

Technology trends are regularly scanned and tracked by core teams at Aptech Ltd. TAG* analyzes these on a monthly basis to understand the emerging technology training needs for the Industry.

An annual Industry Recruitment Profile Survey# is conducted during August - October to understand the technologies that Industries would be adapting in the next 2 to 3 years. An analysis of these trends & recruitment needs is then carried out to understand the skill requirements for different roles & career opportunities.

The skill requirements are then mapped with the learner profile (user system) to derive the Learning objectives for the different roles.

- **Needs analysis and design of curriculum**

The Learning objectives are then analyzed and translated into learning tasks. Each learning task or activity is analyzed in terms of knowledge, skills and attitudes that are required to perform that task. Teachers and domain experts do this jointly. These are then grouped in clusters to form the subjects to be covered by the curriculum.

In addition, the society, the teachers, and the industry expect certain knowledge and skills that are related to abilities such as learning-to-learn, thinking, adaptability, problem solving, positive attitude etc. These competencies would cover both cognitive and affective domains.

A precedence diagram for the subjects is drawn where the prerequisites for each subject are graphically illustrated. The number of levels in this diagram is determined by the duration of the course in terms of number of semesters etc. Using the precedence diagram and the time duration for each subject, the curriculum is organized.

- **Design & development of instructional materials**

  The content outlines are developed by including additional topics that are required for the completion of the domain and for the logical development of the competencies identified. Evaluation strategy and scheme is developed for the subject. The topics are arranged/organized in a meaningful sequence.

  The detailed instructional material – Training aids, Learner material, reference material, project guidelines, etc.- are then developed. Rigorous quality checks are conducted at every stage.

- **Strategies for delivery of instruction**

  Careful consideration is given for the integral development of abilities like thinking, problem solving, learning-to-learn etc. by selecting appropriate instructional strategies (training methodology), instructional activities and instructional materials.

  The area of IT is fast changing and nebulous. Hence, considerable flexibility is provided in the instructional process by specially including creative activities with group interaction between the students and the trainer. The positive aspects of Web based learning –acquiring information, organizing information and acting on the basis of insufficient information are some of the aspects, which are incorporated, in the instructional process.

- **Assessment of learning**

  The learning is assessed through different modes – tests, assignments & projects. The assessment system is designed to evaluate the level of knowledge & skills as defined by the learning objectives.

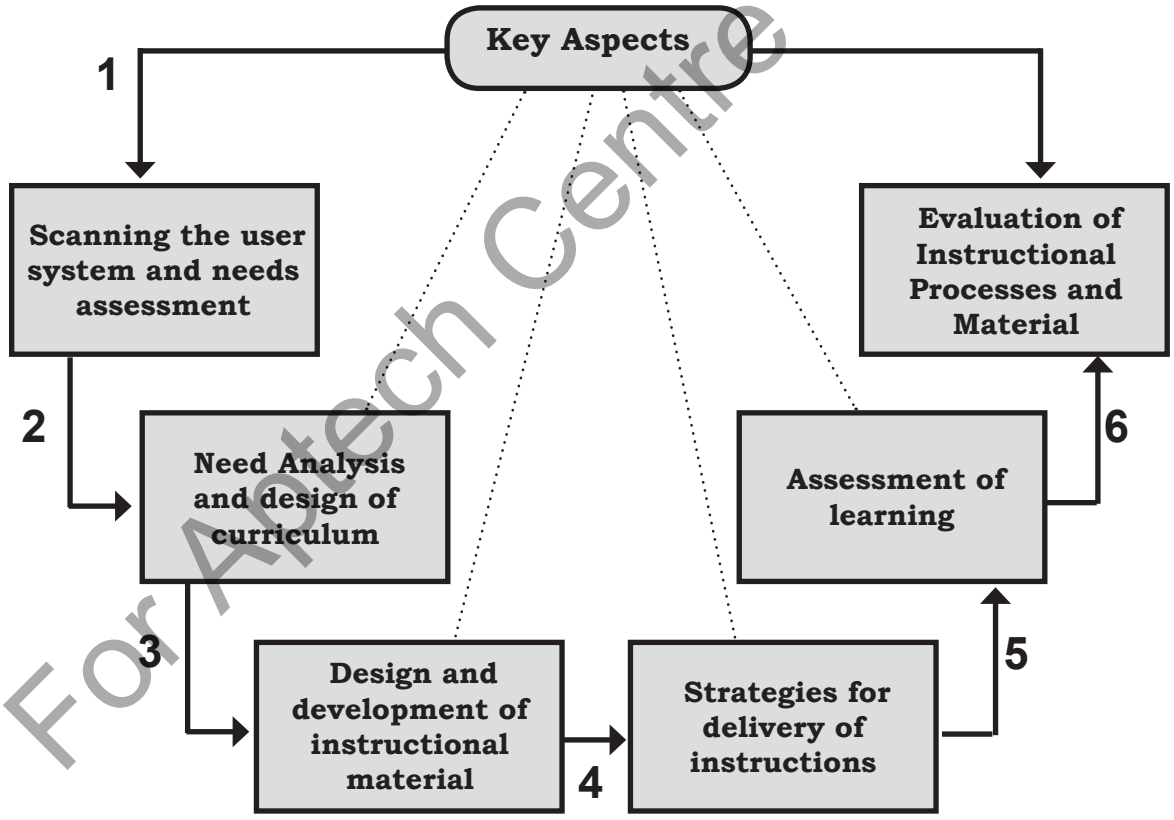
- **Evaluation of instructional process and instructional materials**

  The instructional process is backed by an elaborate monitoring system to evaluate - on-time delivery, understanding of a subject module, ability of the instructor to impart learning. As an integral part of this process, we request you to kindly send us your feedback in the reply pre-paid form appended at the end of each module.

*TAG – Technology & Academics Group comprises of members from Aptech Ltd., professors from reputed Academic Institutions, Senior Managers from Industry, Technical gurus from Software Majors & representatives from regulatory organizations/forums.

Technology heads of Aptech Ltd. meet on a monthly basis to share and evaluate the technology trends. The group interfaces with the representatives of the TAG thrice a year to review and validate the technology and academic directions and endeavors of Aptech Ltd.

Aptech New Products Design Model



“A little learning is a dangerous thing,
but a lot of ignorance is just as bad”

Preface

The book 'Laravel Framework for Web Applications' Trainer's Guide provides introduction to Laravel framework, blade templating engine, and Eloquent ORM. In addition to this it contain concepts of routing, database configuration, controller utilization, and how to create applications using Laravel. The faculty/trainer should teach the concepts in the theory class using the slides. This Trainer's Guide will provide guidance on the flow of the session and also provide tips and additional examples wherever necessary. The trainer can ask questions to make the session interactive and also to test the understanding of the students.

This book is the result of a concentrated effort of the Design Team, which is continuously striving to bring you the best and the latest in Information Technology. The process of design has been a part of the ISO 9001 Certification for Aptech-IT Division, Education Support Services. As part of Aptech's quality drive, this team does intensive research and curriculum enrichment to keep it in line with industry trends.

We will be glad to receive your suggestions.

Design Team

“Practice is the best of all instructors.”

Aptech LEARNING
Unleash your potential

Table of Contents

Sessions

“The future depends on what we do in the present.”

Session 1: Setting Up the Environment

1.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. Prepare a question or two that will be a key point to relate the current session objectives.

1.1.1 Teaching Skills

To teach this session, you should be well-versed with components to set up the development environment. You should be able to set up PHP and composer. You should be able to explain the basics of VirtualBox, Vagrant, Homestead, and Laravel. Describe them how to verify installations and versions of various software.

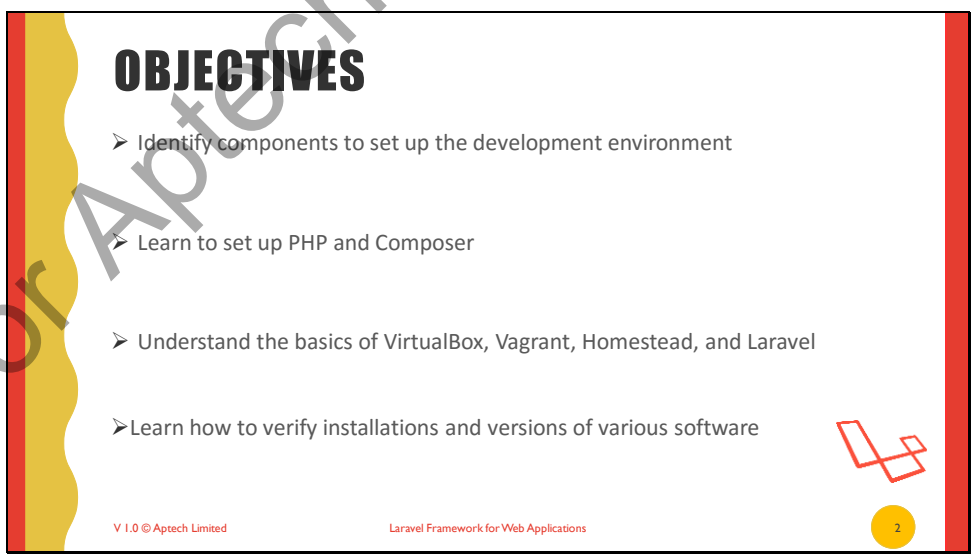
For teaching in the class, you are expected to use slides and LCD projectors.

Tips:
It is recommended that you test the understanding of the students by asking questions in between the class.

In-Class Activities
Follow the order given here during In-Class activities:

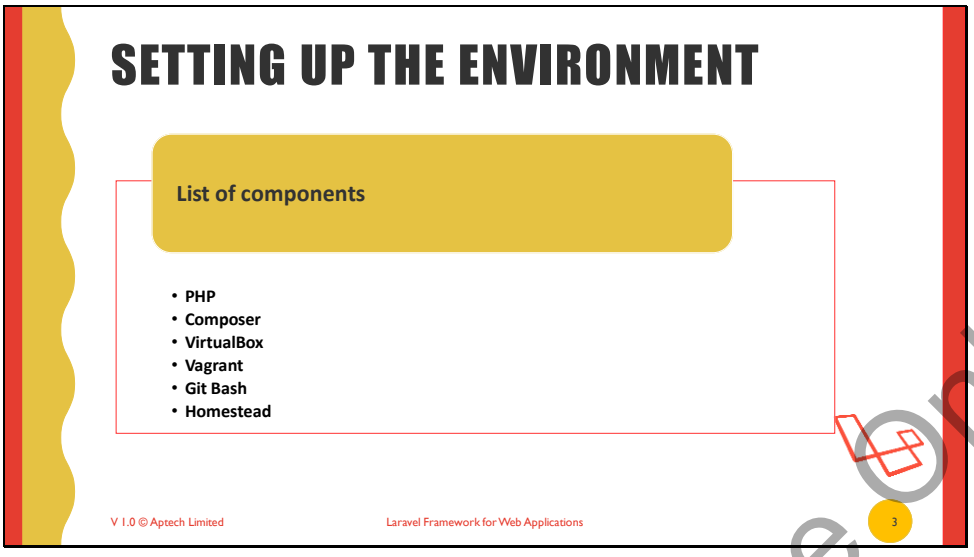
Overview of the Session
Give the students an overview of the current session in the form of session objectives. Show slide 2 of the presentation to students.

Show Slide 2



Instruction(s) to the trainer:
Using this slide, give a brief introduction about this session.

Show Slide 3



Instruction(s) to the trainer:

Using slide 3, explain the various components required to develop Web applications with Laravel in detail. Explain that PHP should be installed on the Windows host operating system to use composer.

• **PHP**: To be installed on the Windows host operating system.

• **Composer**: Used to automate installation of Homestead along with vagrant.

• **VirtualBox**: Allows running guest VMs.

• **Vagrant**: Is an automation tool for building and managing virtual machine environments.

• **Git Bash**: Helps use git commands and provides Unix-type commands that can be run on Windows and used to generate SSH keys.

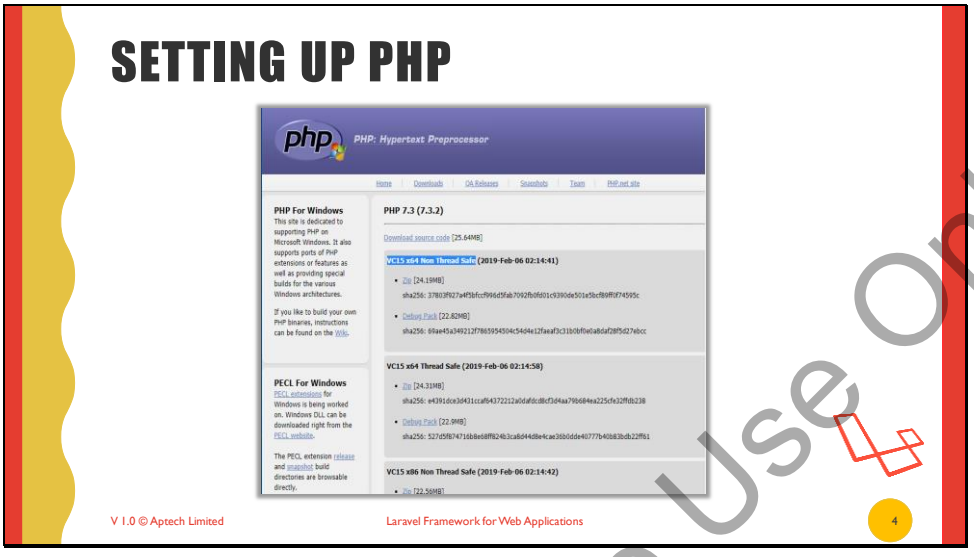
• **Homestead**: Is a virtual environment to efficiently code Web applications in Laravel using PHP.

For more information about Laravel, refer https://www.gangboard.com/blog/laravel-tutorial/
For more information on Homestead, refer https://laravel.com/docs/5.8/homestead
To learn more about PHP, refer https://php.net/

In-Class Questions:
What is VirtualBox?
Answer: It is a free and open source hypervisor that will allow running guest VMs, such as the Homestead environment.

Activity:
Ask the students to make groups and give a presentation on the Laravel and its features.

Show Slide 4



Instruction(s) to the trainer:
Using slide 4, explain the steps to set up PHP environment on Windows operating system.
Explain that to use composer and homestead on Windows, PHP binaries are required:
Download them from PHP's official Website https://windows.php.net/download/

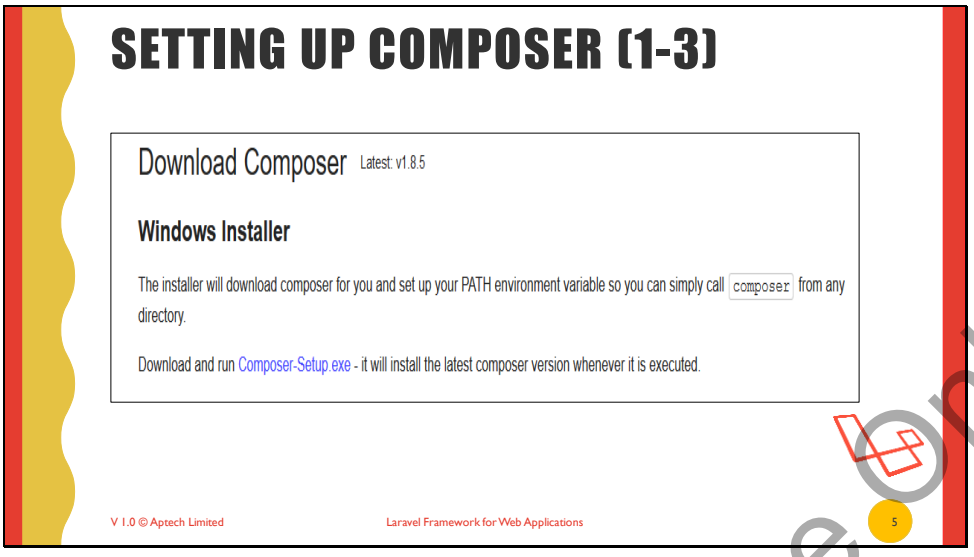
To complete the installation, ask them to follow the steps:
Step 1: Download the latest x86/x64 Non Thread Safe version of PHP from the official Website. The image in the slide 4 shows different PHP versions.

Step 2: Add the path of the extracted php files to the environment variables:
https://secure.php.net/manual/en/faq.installation.php#faq.installation.addtopath

Activities:
Ask the students to create a document listing out the various features of PHP.

For more information on PHP, refer:
https://skillcrush.com/2012/04/11/php/
https://www.php.net/manual/en/history.php.php

Show Slides 5 to 7



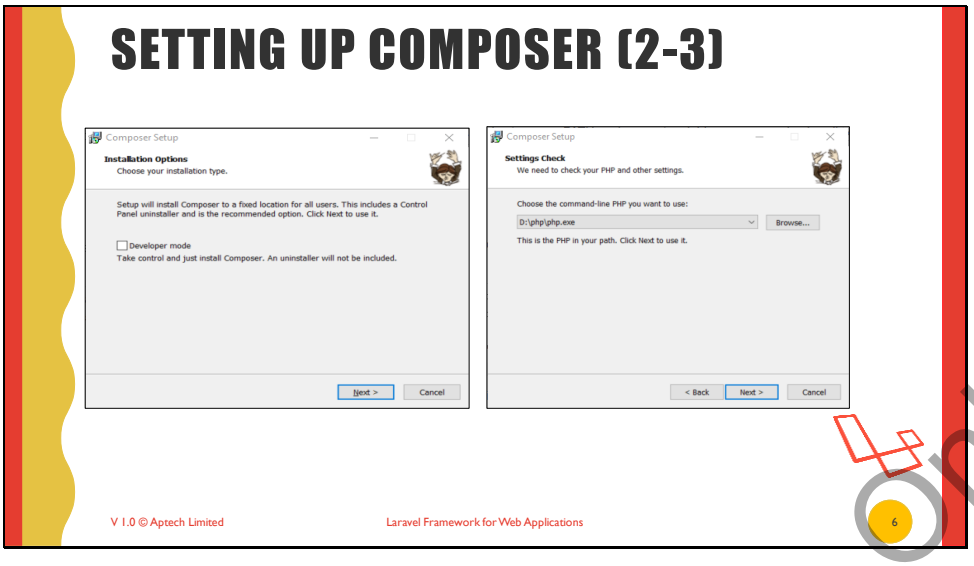
Instruction(s) to the trainer:

Using slide 5 to 7, explain the steps for setting up the composer. Explain the composer to the students and its requirement for setting up the environment. Explain that it is a dependency management tool for PHP. In the host operating system, Composer will be used to automate the installation of Homestead along with Vagrant. Download the composer from the official Website https://getcomposer.org/download/.

In the Website, click **Composer-Setup.exe** as shown in slide 5. This will download the setup for installing composer on Windows.

For more on composer, refer:
https://decodeweb.in/php/php-frameworks/laravel-framework/what-is-dependency-management-in-laravel/

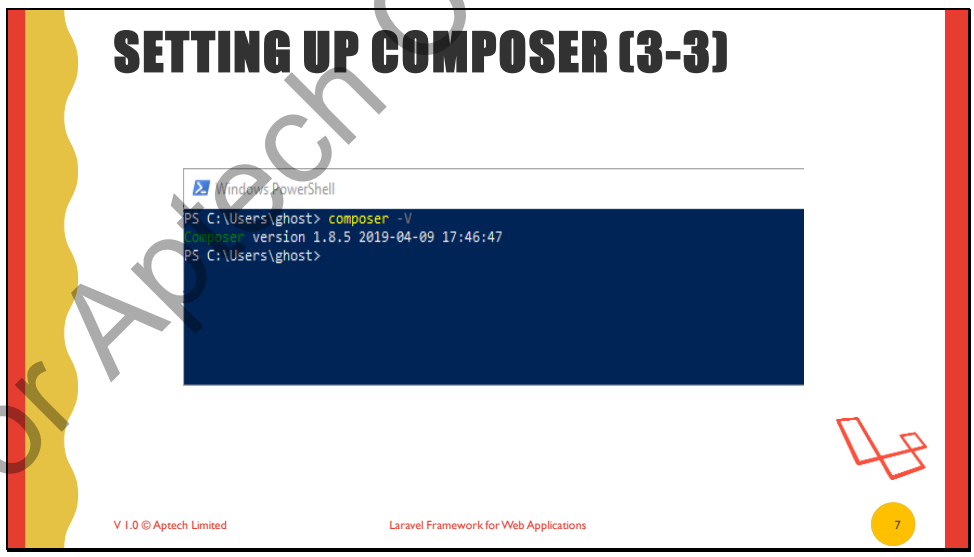
Ask:
Why is composer addressed as a dependency manager?
Accept the answers from the students and explain the reason to the students.



Instruction(s) to the trainer:
Using slide 6, show the installation options page and settings check page. Once the composer is downloaded as explained in slide 5, start the setup to initiate the installation wizard. Next, perform the following steps:

Step 1: On the **Installation Options** page shown in slide 6, click **Next** to proceed further.

Step 2: On the **Settings Check** page shown in slide 6, select the required php version exe path.



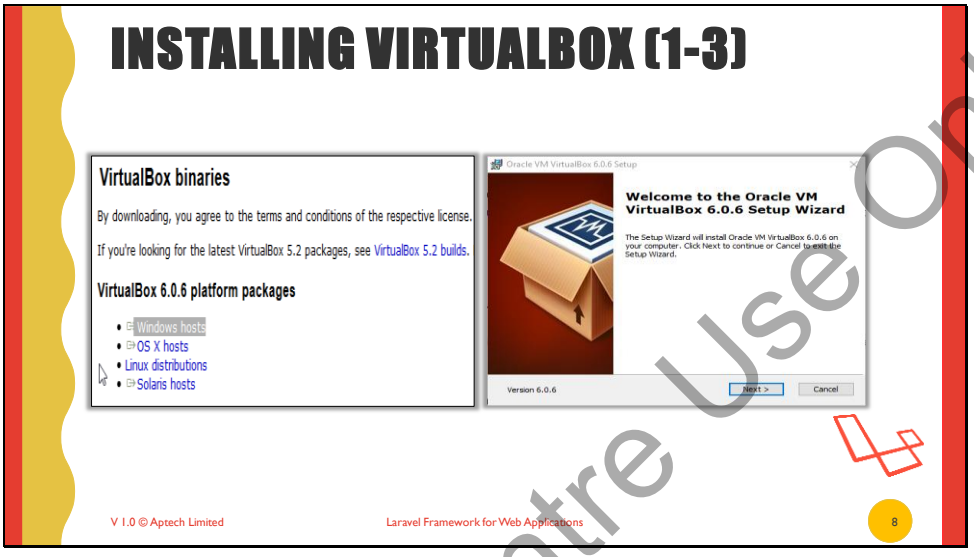
Instruction(s) to the trainer:
Using slide 7, explain the consecutive steps in the installation process.

Step 3: Click **Next** on all successive pages till the wizard tells you that the Composer is successfully installed.

Step 4: To verify installation, open cmd or PowerShell and type the command as shown in the slide 7.

Activity:
Ask the students to prepare a document that explains composer in detail. (This document can be maintained for future reference.)

Show Slides 8 to 10



Instruction(s) to the trainer:
Using slide 8 to 10, explain the steps involved to download and install Virtual Box. Refer following link for more information:

https://www.virtualbox.org/wiki/Downloads

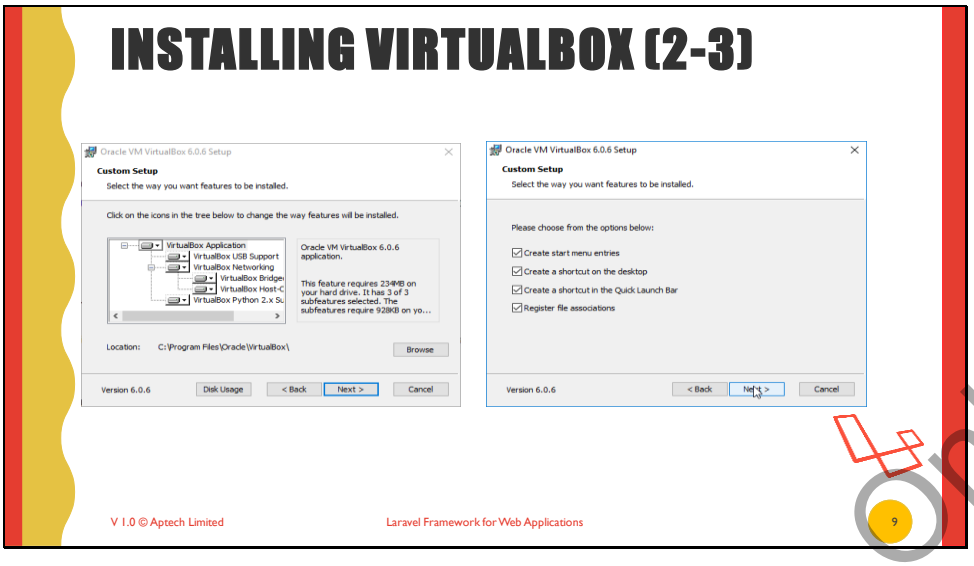
Explain that it is a free and open source hypervisor that will allow running guest VMs, such as the Homestead environment. Explain the concept to hypervisor to the students.

Virtual box is installed as follows:

Step 1: To download the setup, on the Website, click **Windows hosts** as shown in slide 8.

Step 2: On the Welcome to the Oracle VM VirtualBox 6.0.6 Setup Wizard page as shown in slide 8, click **Next**.

For more knowledge on Virtualbox, refer:
https://www.howtogeek.com/66734/htg-explains-what-is-a-hypervisor/



Instruction(s) to the trainer:
Using slide 9, show and explain the further steps for installing a virtual box.

Step 3: On the **Custom Setup** page as shown in slide 9, click **Next** to proceed further.

Step 4: To select the way for installing features, select the required options and click **Next**.

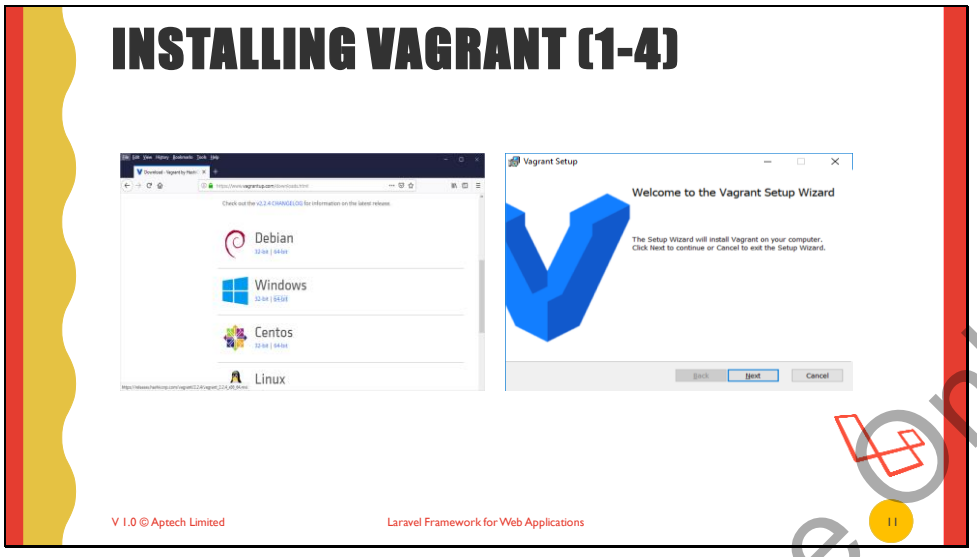
Instruction(s) to the trainer:
Using slide 10, show the last steps for installing a virtual box.

Step 5: On the Warning page, click **Yes** to proceed further.

Step 6: On the Installation Complete page, click **Finish**.

Activity:
Ask the students to prepare a document that explains the VirtualBox in detail. (This document can be maintained for future reference.)

Show Slides 11 to 14



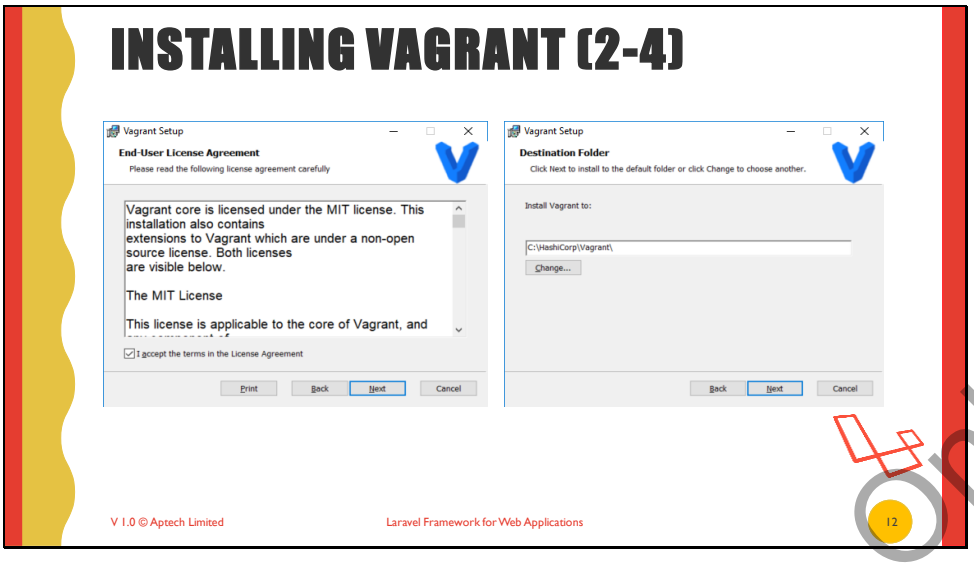
Instruction(s) to the trainer:
Using slide 11 to 14, explain the installation of Vagrant using the following steps. Explain Vagrant in detail to the students. Explain that it is an automation tool for building and managing virtual machine environments. Homestead is a Vagrant box.

Step 1: On the official Website https://www.vagrantup.com/downloads.html as shown in slide 11, select either 32-bit or 64-bit.

Step 2: On the Welcome to the Vagrant Setup Wizard page as shown in the slide, click **Next**.

For more information on Vagrant, refer:
https://www.vagrantup.com/intro/index.html

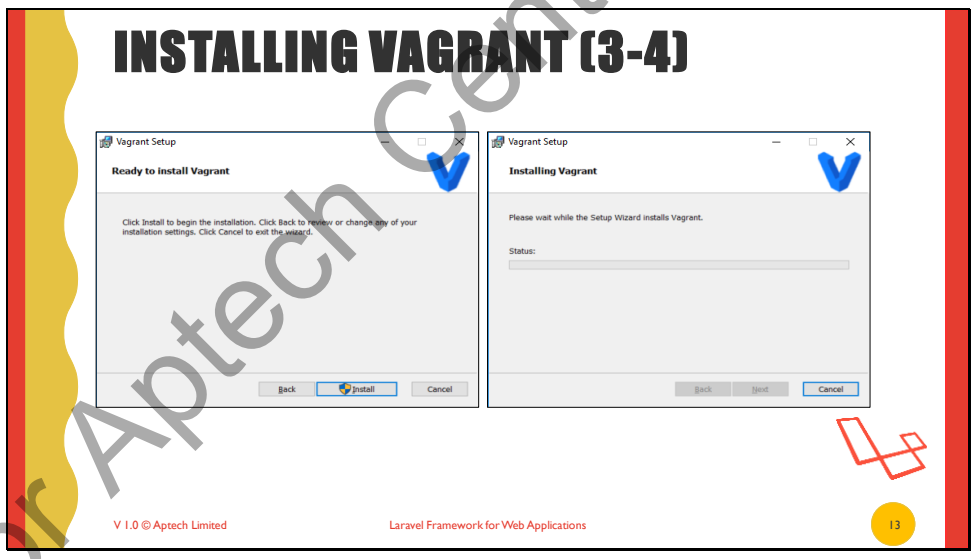
In-Class Questions:
What is Vagrant?
Answer: It is an automation tool for building and managing virtual machine environments.



Instruction(s) to the trainer:
Using slide 12, explain the remaining steps for installing Vagrant.

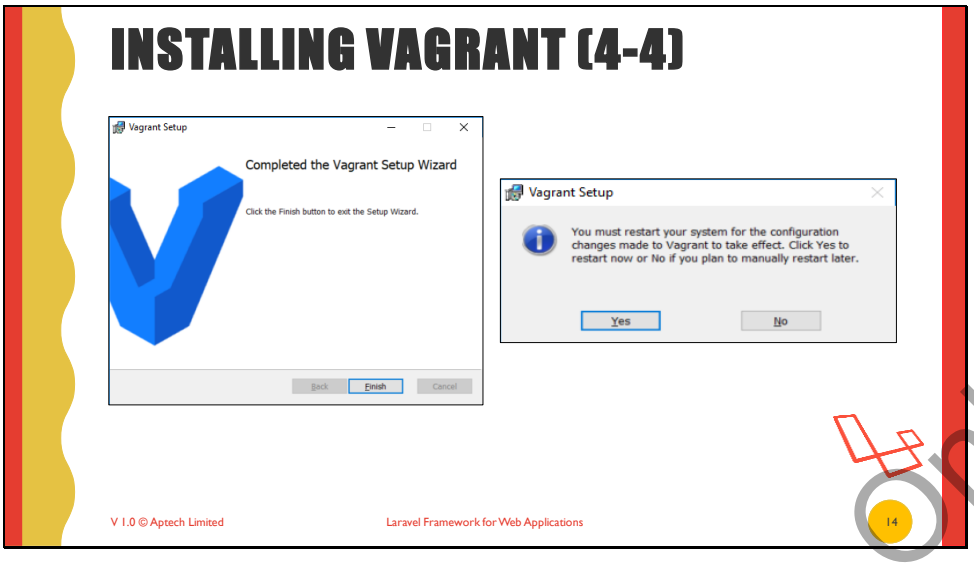
Step 3: On the **End-User License Agreement** page as shown in slide 12, click **Next.**

Step 4: On the **Destination Folder** page, choose the installation directory and click **Next**.



Instruction(s) to the trainer:
Using slide 13, explain next step for installation.

Step 5: On the **Ready to install Vagrant** page that displays as shown in slide 13, click **Install** to begin the installation process**.**



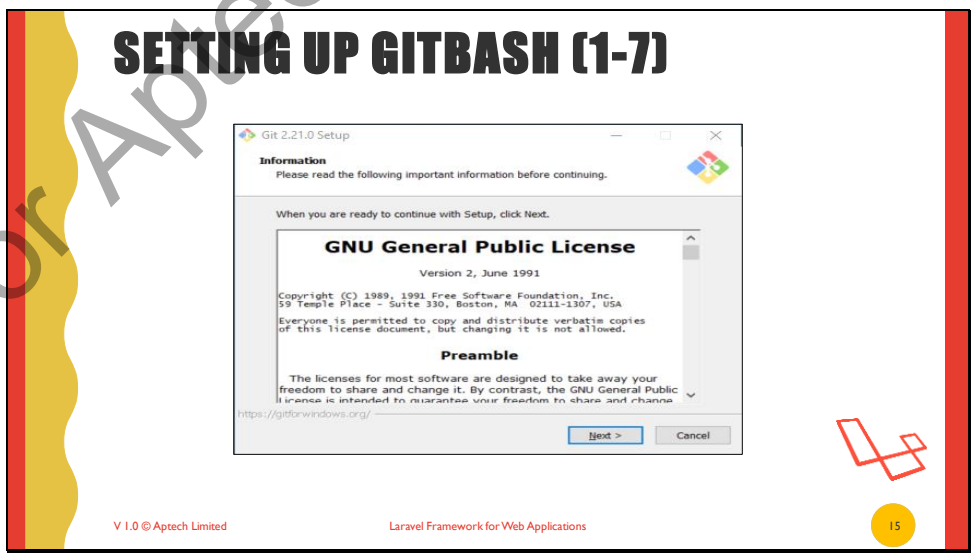
Instruction(s) to the trainer:
Using slide 14, explain remaining steps for installation.

Step 6: Once the installation is complete, on the Completed Vagrant Setup page, click **Finish**.

Step 7: For the configuration changes to take effect, a dialog box displays that prompts to restart the system. Click **Yes**.

Activity:
Ask the students to prepare a document that explains Vagrant in detail. (This document can be added to the previous documents and maintained for future reference.)

Show Slides 15 to 21



Instruction(s) to the trainer:
Using slide 15 to 21, explain the steps to set up Git Bash. Explain Git Bash and mention that it is a shell that comes along with Git for Windows.

It helps use git commands and provides Unix-type commands that can be run on Windows. Git Bash will be used to generate SSH keys. Explain that it provides an emulation layer and Bash stands for Bourne Again Shell. To download the Git executable file, visit the Website:

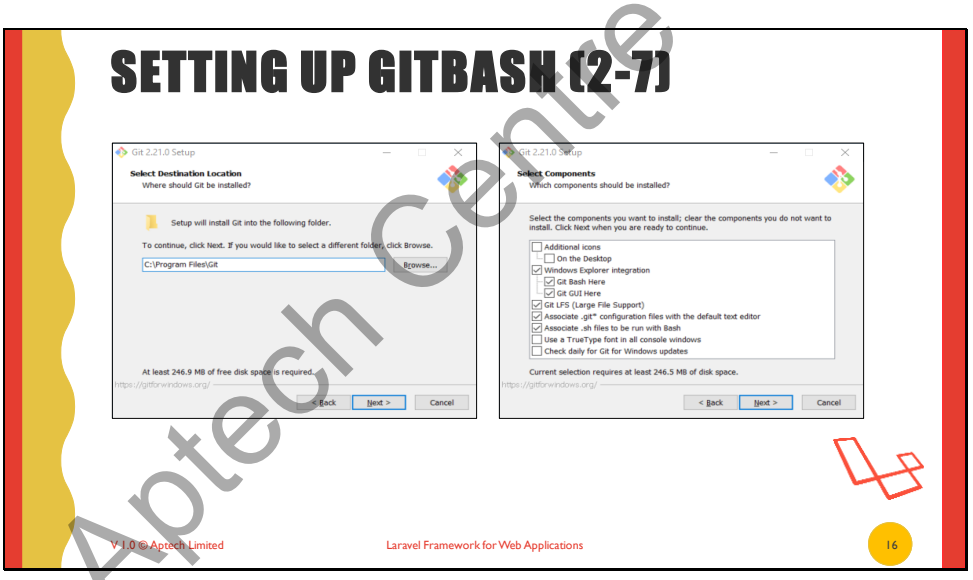
https://git-scm.com/download/win

After downloading is finished, follow the steps:

Step 1: The **Information** page displays GNU License information as shown in slide. Click **Next** to visit the next screen.

For more information on Git Bash, refer https://www.atlassian.com/git/tutorials/git-bash

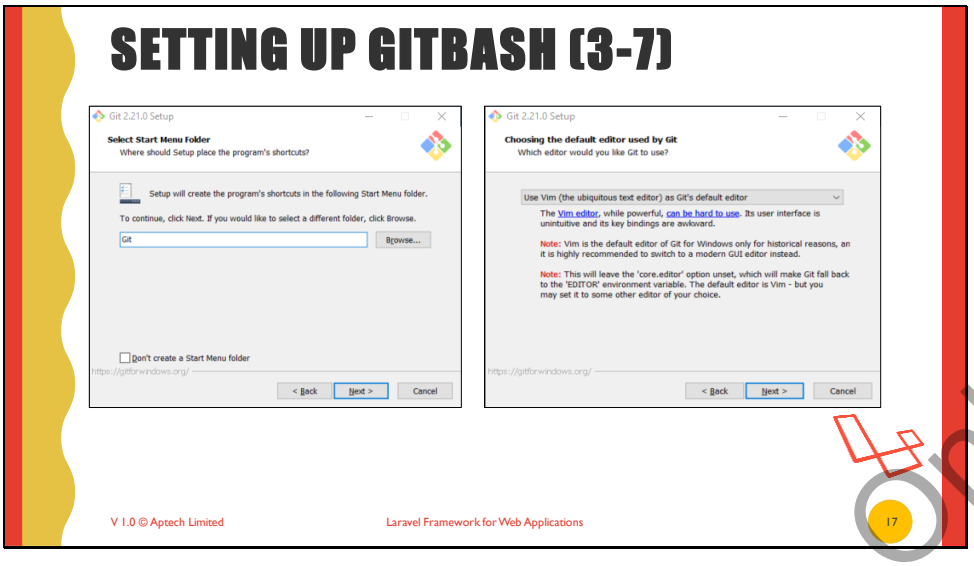
In-Class Questions:
What is Git Bash?
Answer: It is a shell that comes along with Git for Windows. It helps use git commands and provides Unix-type commands that can be run on Windows.



Instruction(s) to the trainer:
Using slide 16, explain the consecutive steps involved to set up Git Bash.

Step 2: On the **Select Destination Location** page, select the required destination as shown in the slide and click **Next** to visit the next screen.

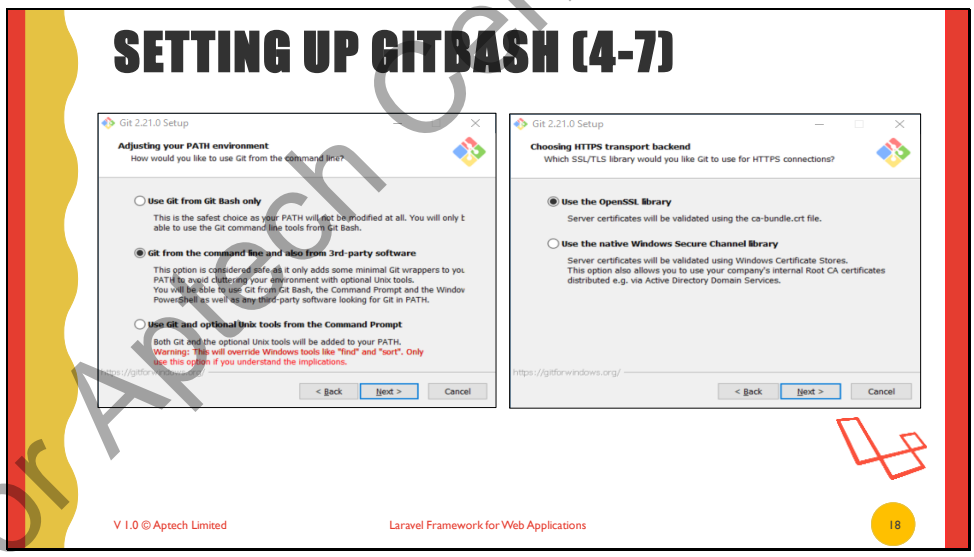
Step 3: On the **Select Components** page, select the required components as shown and click **Next** to visit the next screen.



Instruction(s) to the trainer:
Using slide 17, explain the remaining steps to set up Git Bash.

Step 4: On the **Select Start Menu Folder** page, select the required folder name as shown in slide 17 and click **Next** to visit the next screen.

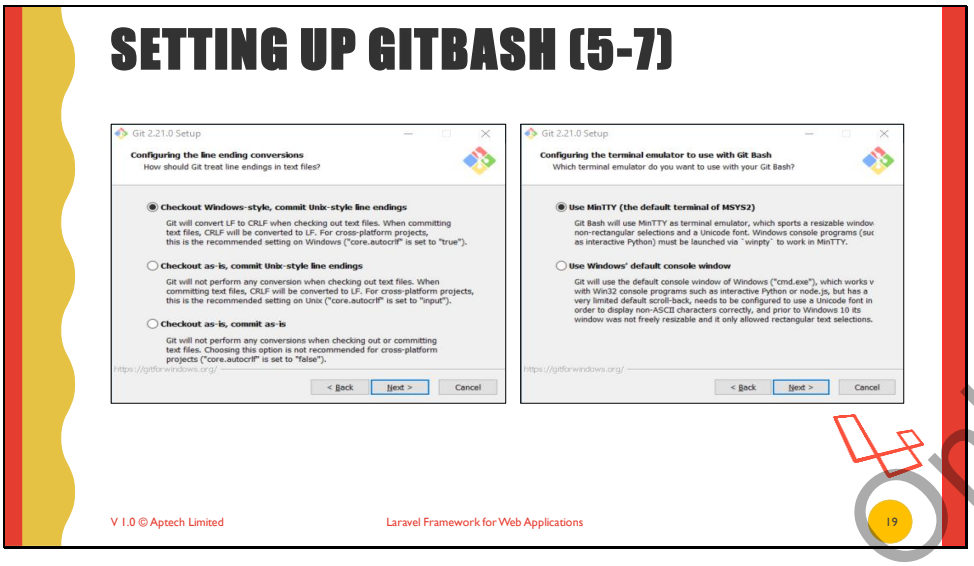
Step 5: To choose the default editor, select the editor you want to use as shown and click **Next** to visit the next screen.



Instruction(s) to the trainer:
Using slide 18, explain the next steps to setup Git Bash.

Step 6: Select Git from the command line on the Path Environment page as shown in slide 18 and click **Next** to visit next screen.

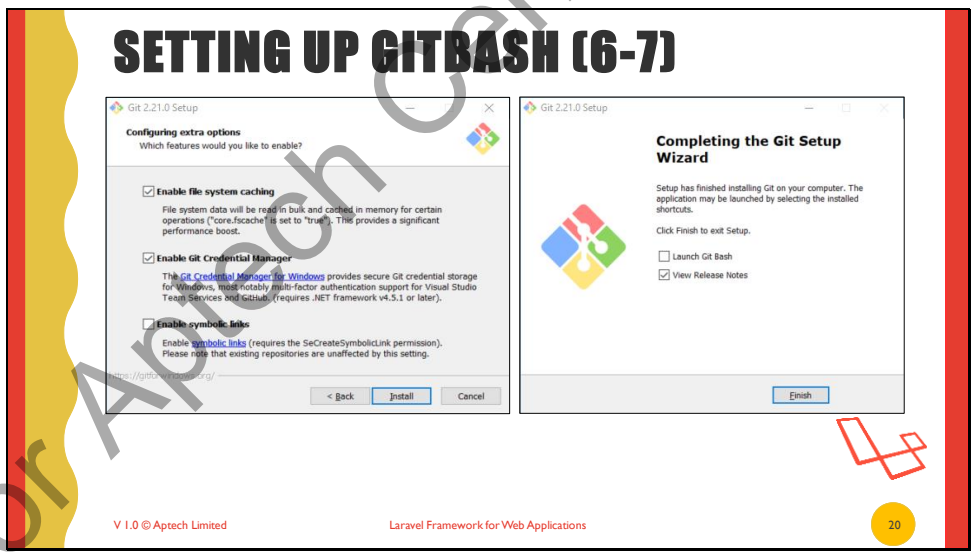
Step 7: Select the OpenSSL library as the HTTPS transport backend on the HTTPS transport backend page as shown and click **Next** to visit next screen.



Instruction(s) to the trainer:
Using slide 19, explain the next steps to set up Git Bash.

Step 8: Select checkout text files in Windows-style on the line ending conversions page as shown and click **Next** to visit next screen.

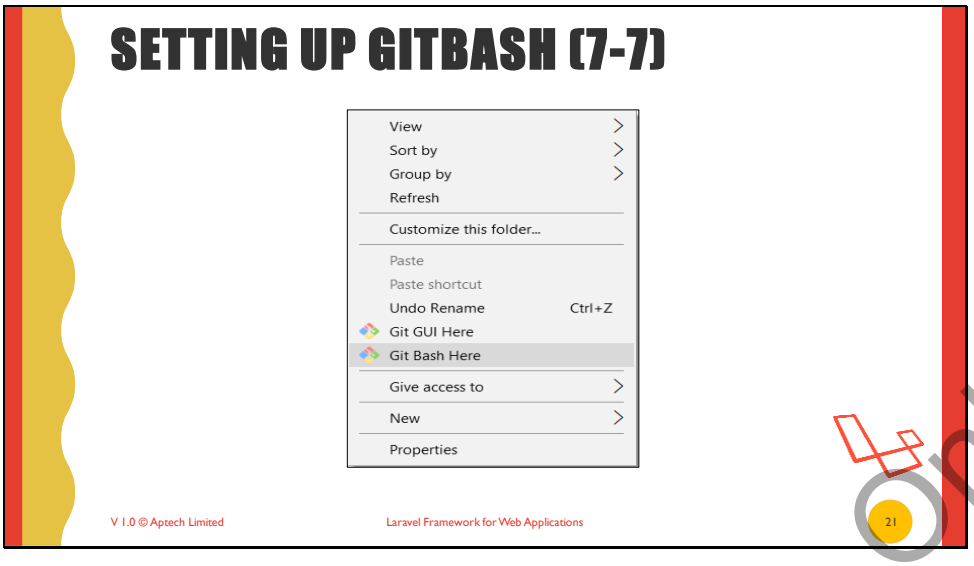
Step 9: To use MinTTY as the terminal emulator, select **Use MinTTY** as shown and click **Next** to proceed further.



Instruction(s) to the trainer:
Using slide 20, explain the next steps involved to set up Git Bash.

Step 10: To configure extra options, select the required options on the **Configuring extra options** page as shown in slide 20 and click **Install** to start the installation.

Step 11: On the **Completing the Git Setup Wizard** page, click **Finish** after the installation is over.

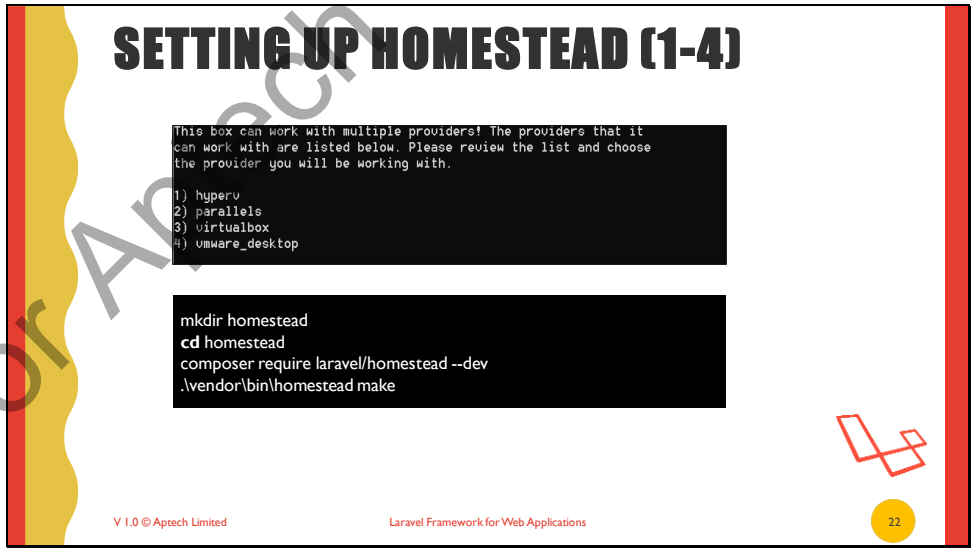


Instruction(s) to the trainer:
Using slide 21, show and explain the remaining steps to set up Git Bash.

Step 12: To verify the installation, right-click any empty space on the desktop and select **Git Bash Here** as shown in the slide.

Activity:
Ask the students to prepare a document that explains the Git Bash in detail. (This document can be appended to previous documents and can be maintained for future reference.)

Show Slides 22 to 26



Instruction(s) to the trainer:
Using slide 22 to 26, explain the steps involved in setting up Homestead. Mention that Homestead is a vagrant box. Explain that Homestead is a virtual environment that provides resources to efficiently code Web applications in Laravel using PHP. It comes with PHP and composer pre-installed in it.

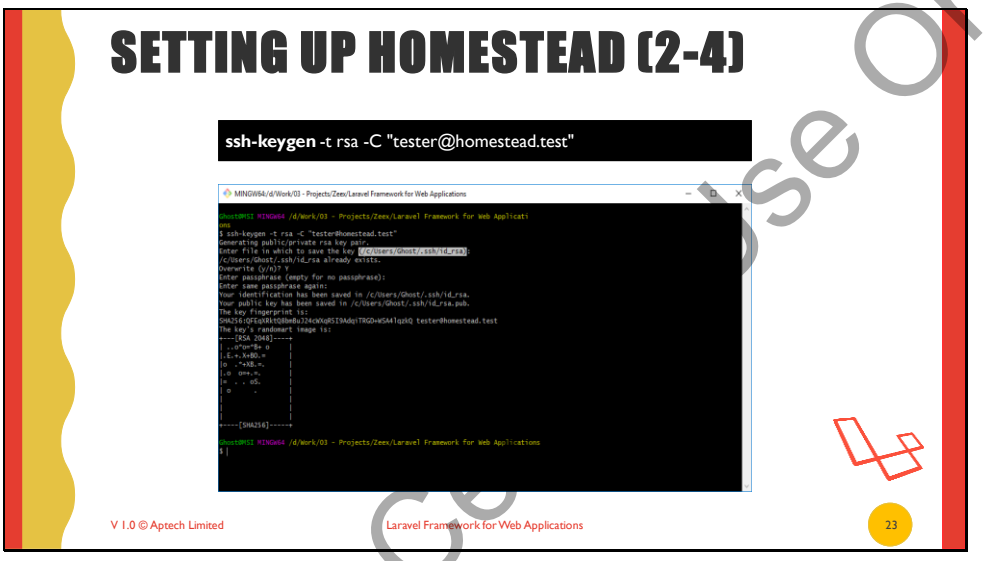
To install Homestead, perform the following steps:

Step 1: Use the command **vagrant box add laravel/homestead** on any other command line tools such as cmd, PowerShell, and Git Bash.

Step 2: Select **virtualbox** as hypervisor when prompted as shown in the slide.

Step 3: Use the composer command as shown in the second code snippet.

For more information on Homestead, refer:
https://laravel.com/docs/5.8/homestead

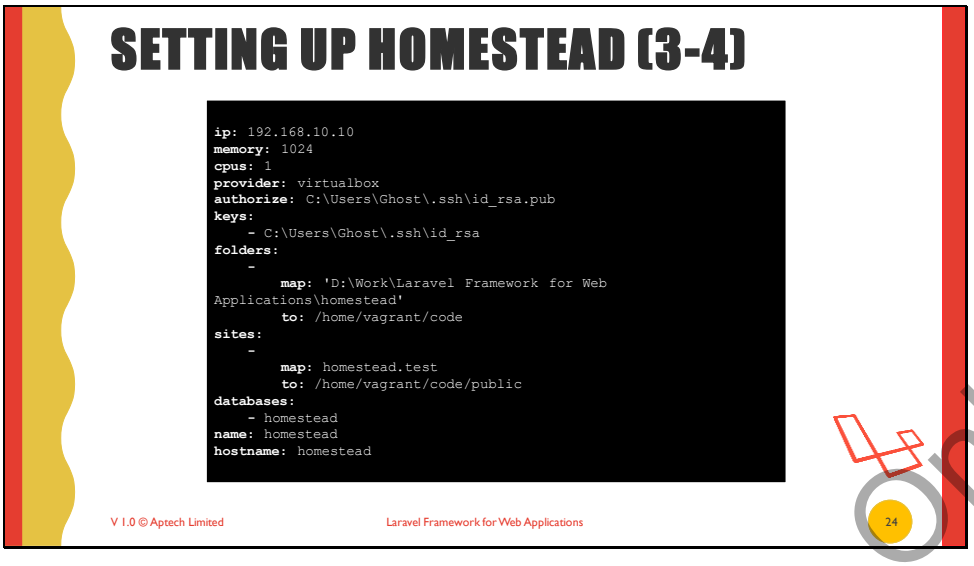


Instruction(s) to the trainer:

Using slide 23, explain the remaining steps involved in setting up Homestead.

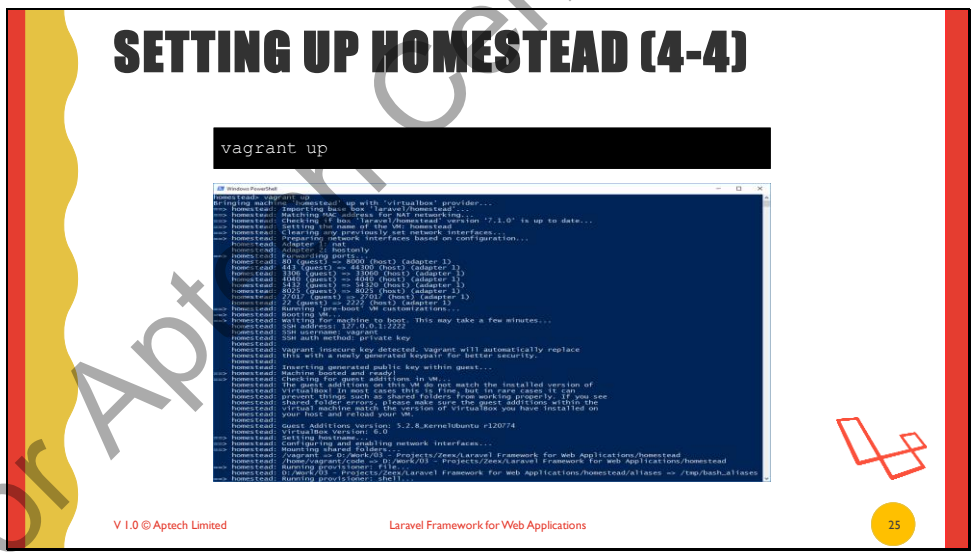
Step 4: To configure the ssh keys, start Git Bash and input command as shown in the code snippet.

Step 5: This displays a command line wizard as shown in the image to generate ssh keys.



Instruction(s) to the trainer:
Using slide 24, explain the remaining steps in setting up Homestead.

Step 6: To convert the Unix style Address into Windows style, map **/c/Users/Ghost/.ssh/id_rsa** to **C:\Users\Ghost\.ssh\id_rsa**.
Step 7: In the Homestead directory that is created, modify the **Homestead.yaml** file and change the values depending on the desired locations as shown. Note down the IP Address and host in the .yaml file.



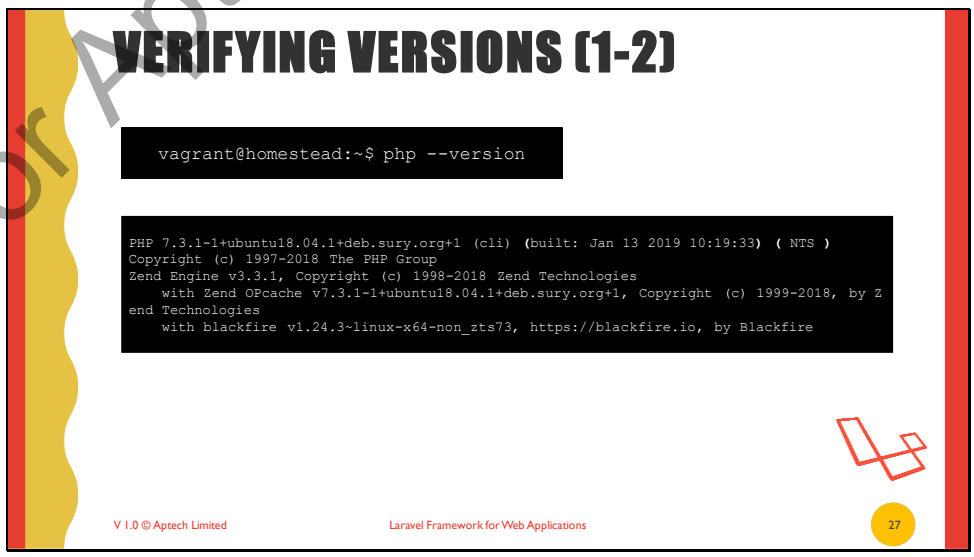
Instruction(s) to the trainer:
Using slide 25, explain the steps in setting up Homestead environment.

Step 1: Open a **powershell/cmd/git** bash window in the homestead directory and run the command as shown in the Code Snippet.
Step 2: The homestead environment is successfully configured as shown in the image.

Activity:
Ask the students to prepare a document that explains the Homestead in detail. This document is the final document that can be appended to previous documents. This will help students maintain a document that contains explanation of everything required for setting up the environment so that they can use them for future reference.

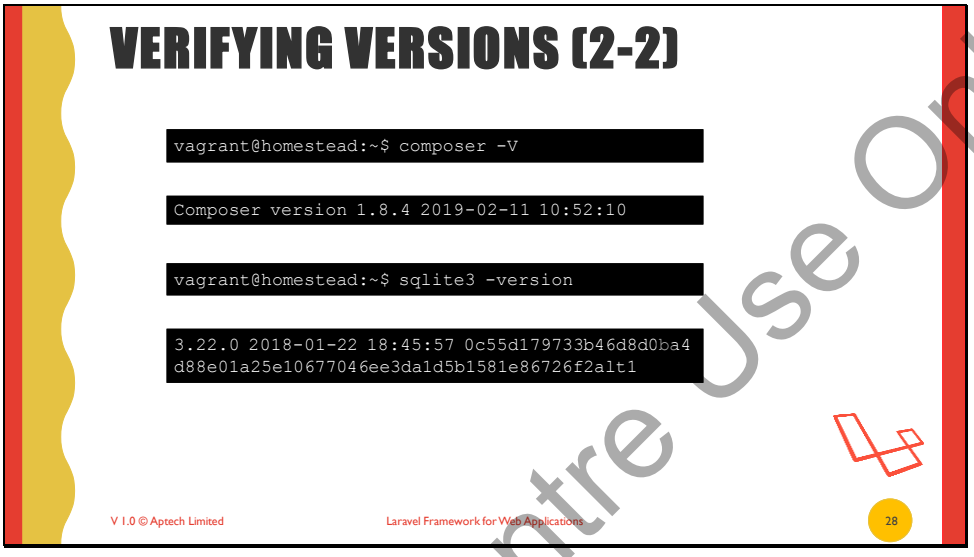
Instruction(s) to the trainer:
Using slide 26, explain Laravel command line utility.
Explain that once the Homestead virtual environment is set, it will include all tools required to create a Laravel application. The Laravel command line utility can be used to create an application based on the Laravel framework. This is required as Laravel is not an application but a framework. The second code snippet verifies the installation and check version number for newly deployed Homestead environment. This prompts for a password as shown in the slide. The default password is **vagrant**.

Show Slides 27 and 28



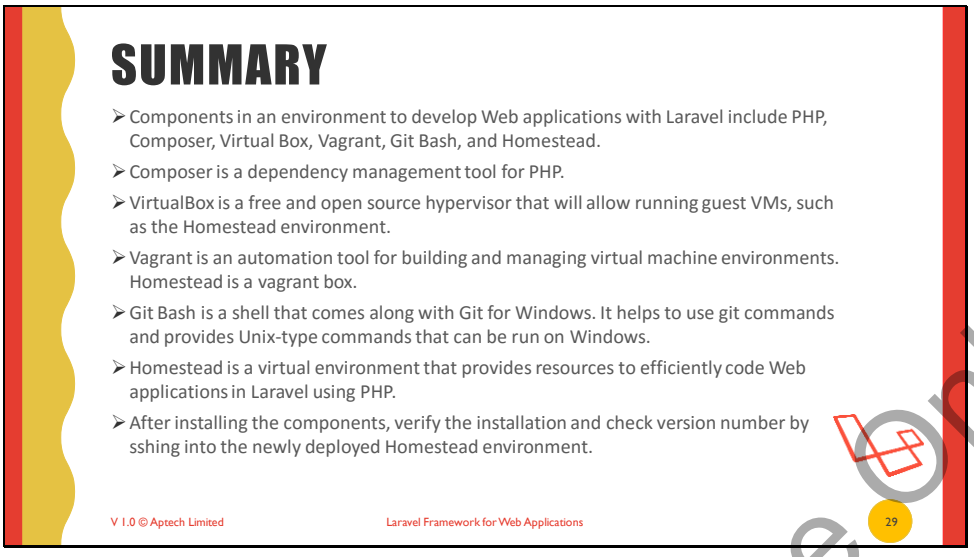
Instruction(s) to the trainer:
Using slide 27 and 28, explain how to verify versions of the different components. Using slide 27, show how to verify the PHP version in the ssh session. The code snippet for the same is shown in the slide as well as the output is also displayed.

Activity:
Ask the students to make groups and prepare a presentation on all the components required to set up an environment for developing Web application in Laravel.



Instruction(s) to the trainer:
Using slide 28, show the code snippets to verify composer version as well as SQLite version in the ssh session. The output for both is also displayed in the slide that shows the versions of each one of them. Similarly, the code snippet to verify the Homestead version in the ssh session can be shown as follows:
vagrant@homestead:~$ ./code/vendor/bin/homestead –V
The output obtained is as follows:
Laravel Homestead 8.2.0

Show Slide 29



Instruction(s) to the trainer:
Summarize the key concepts taught in this session using this slide.

1.2 Post Class Activities for Faculty

You should familiarize yourself with the topics of the next session.

Tips: You can also check the Articles/Blogs/Expert Videos uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session. You can also connect to online tutors on the OnlineVarsity site to ask queries related to the sessions.

Session 2: Introduction to Laravel

2.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. You should revisit topics of the previous session for a brief review. Here, you can ask students to recall the key topics from previous session. Prepare a question or two that will be a key point to relate the current session objectives.

2.1.1 Teaching Skills

To teach this session, you should be well-versed with the details of Laravel framework and understand the need for frameworks. You should be able to describe the features of Laravel. Also, tell them how to view the Laravel Directory structure.

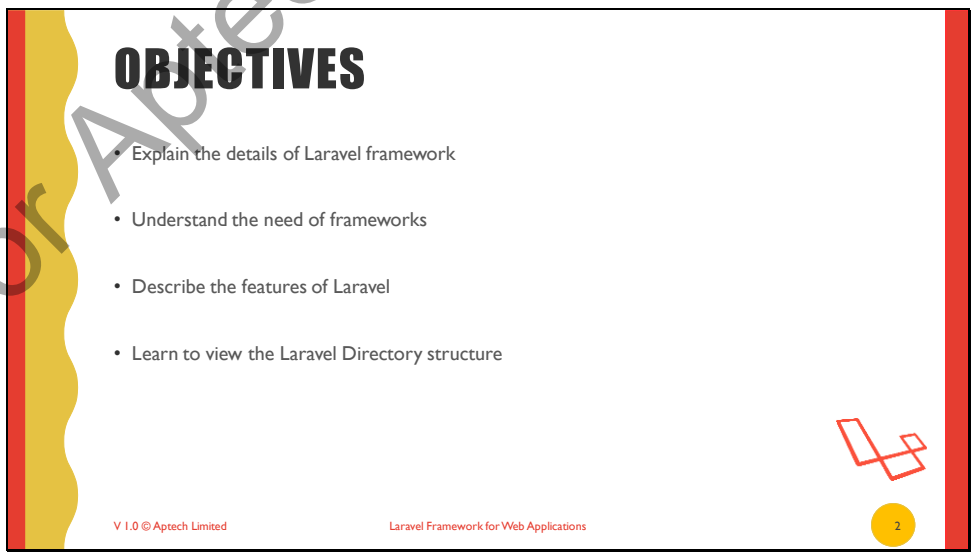
For teaching in the class, you are expected to use slides and LCD projectors.

Tips:
It is recommended that you test the understanding of the students by asking questions in between the class.

In-Class Activities
Follow the order given here during In-Class activities:

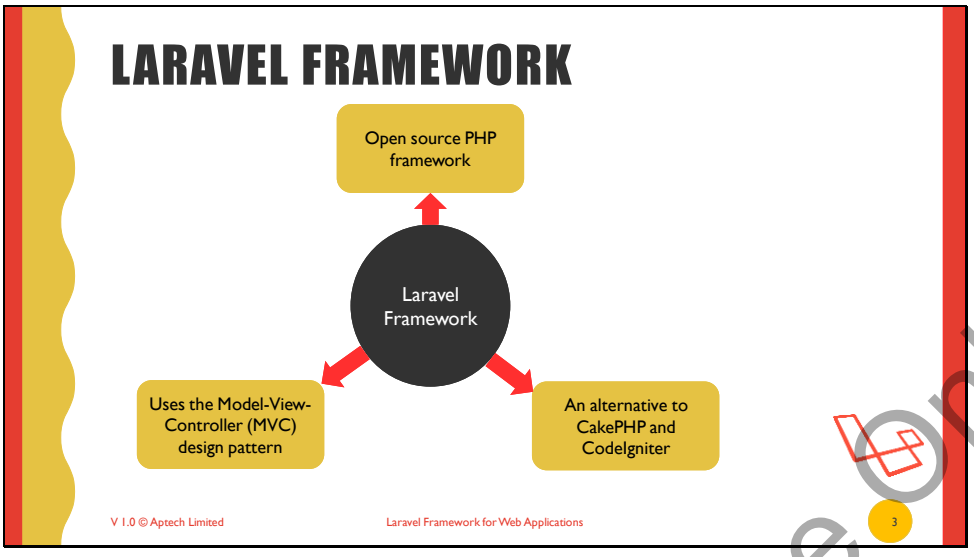
Overview of the Session
Give the students an overview of the current session in the form of session objectives. Show the students slide 2 of the presentation.

Show Slide 2



Instruction(s) to the trainer:
Using slide 2, give a brief introduction about this session.

Show Slide 3

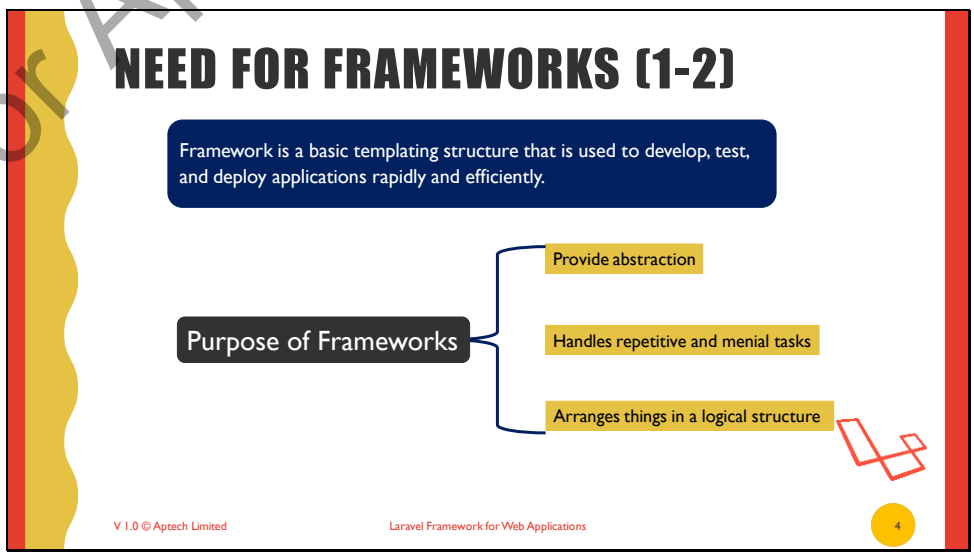


Instruction(s) to the trainer:
Using slide 3, give a brief introduction of Laravel to the students. Laravel is an open source PHP framework for developers working on Web application development. Laravel uses the Model-View-Controller (MVC) design pattern to facilitate development of Web applications in a quick and easy manner.

Additional information:
Official documentation for Laravel 4.2 defines it as follows:
Laravel is a Web application framework with expressive, elegant syntax. Laravel eases the use of common tasks used in the majority of Web projects, such as authentication, routing, sessions, and caching.

For more information on Laravel framework, refer https://www.valuecoders.com/blog/technology-and-apps/laravel-best-php-framework-2017/

Show Slide 4



Instruction(s) to the trainer:
Using slide 4, explain the need for a framework. Give a brief explanation about PHP. It is one of the most widely supported programming languages, especially by the open source community. It is very popular among Web developers. There is immense amount of PHP libraries that are available on the Internet as open source resources. Also, tell them that those libraries can be used for various purposes during application development as per requirement.

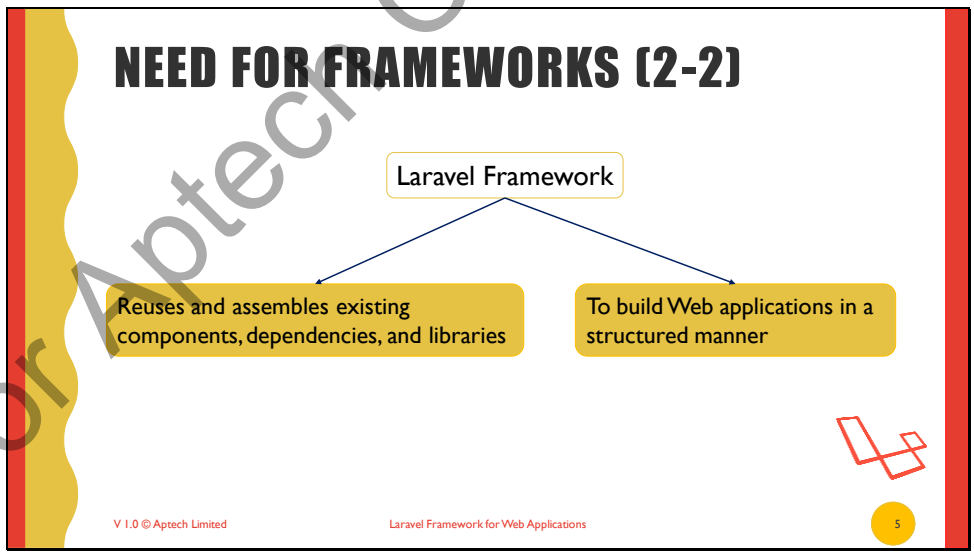
Need for frameworks: Even though various frameworks are available, it is absolutely essential for a framework to meet the standards and guidelines defined for writing Web application codes. This is a time-consuming process. It also takes effort to find out accurate information on the Internet, install dependencies, configure each library, and include them in different files. Frameworks are important in these scenarios.

The framework helps a developer to focus on the actual problem and application logic, and provides tools to perform tasks easily.

For more information on PHP framework, refer
https://www.jotform.com/blog/discussing-php-frameworks/

Ask:
What are the basic features of PHP?

Show Slide 5

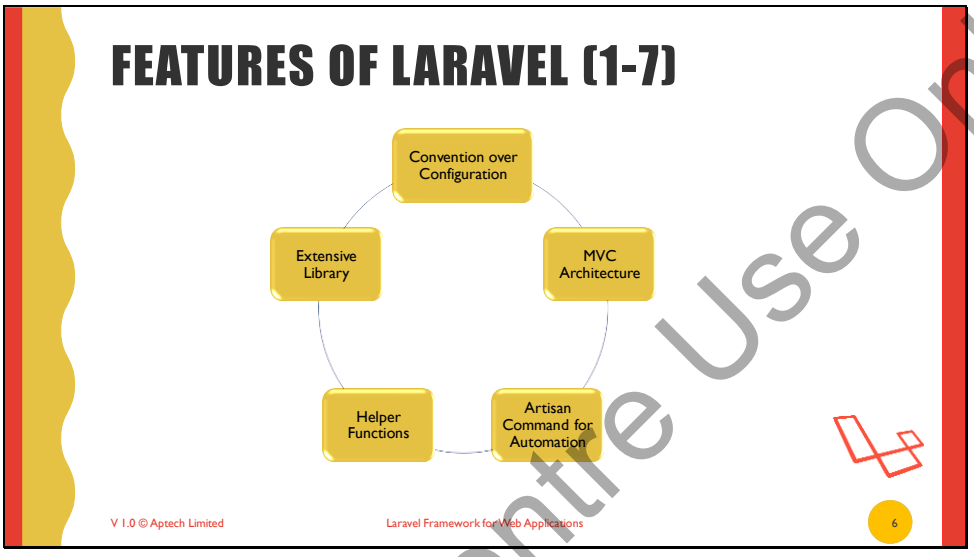


Instruction(s) to the trainer:
Using slide 5, explain important features of Laravel framework. Reusing and assembling existing components, dependencies, and libraries help to automate time-consuming tasks, which in turn helps a Web developer to build Web applications in a structured manner.

For more information on Laravel framework, refer:
https://medium.com/techcompose/why-laravel-is-consider-as-one-of-the-best-php-framework-in-2018-4f40def9c69c

Activity:
Ask students to make a group and create a presentation to include various frameworks available in market for Web development.

Show Slide 6



Instruction(s) to the trainer:
Using slide 6, give a brief idea of the important features of Laravel. Slides 6 to 12 explain in detail each of these features.

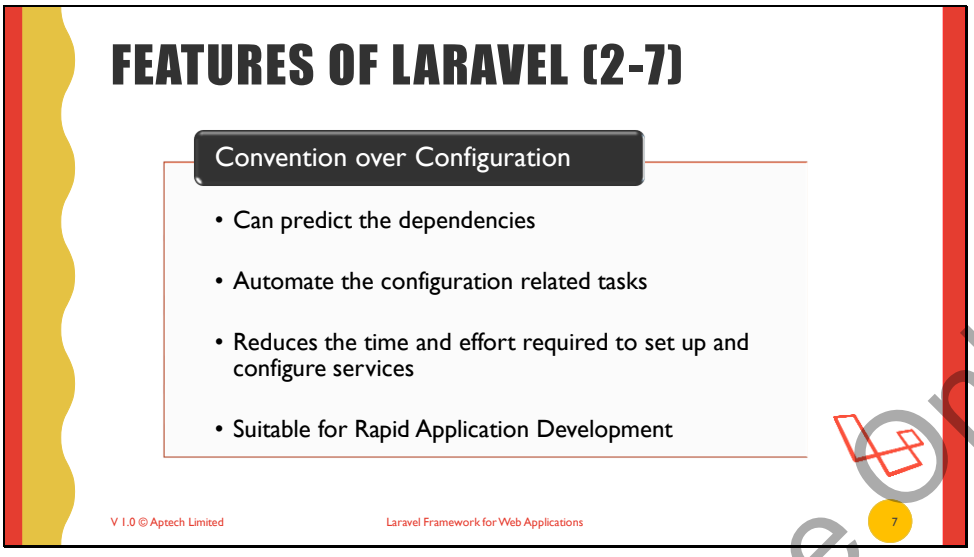
Ask:

Q1. Are you aware about the concept of MVC?
Accept the responses and explain in detail what MVC architecture is.

Activity: Ask them to search and list other features of Laravel, which makes it best framework.

For more information, you can refer: https://www.pixelcrayons.com/blog/web/8-top-features-stats-facts-about-php-laravel-framework/

Show Slide 7

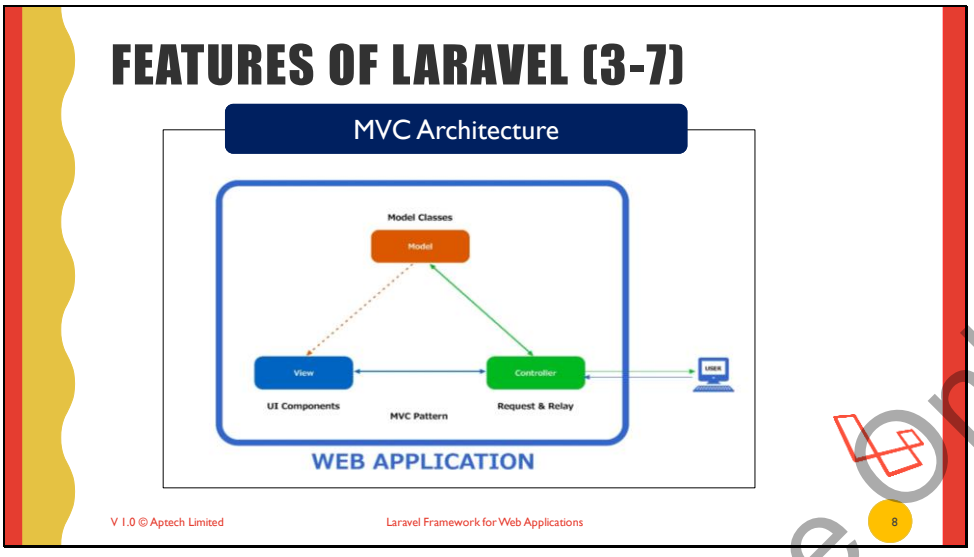


Instruction(s) to the trainer:
Using slide 7, explain in detail the shown feature of Laravel. Give the following example: One simple example of Convention over Configuration can be seen in the process of using models to retrieve records stored in a table. If a Model is created with the name **Employee**, then it will implicitly try to retrieve records from a table name **employees**, which is the plural snake case name of Employee. If the table name is not the plural snake case name of the Model, then the table name would have to be explicitly defined in the code of the Model.

Also, explain Rapid Application Development. It is a development model. It includes rapid prototyping. Developers use this model to make multiple iterations and updates to software without a development schedule from scratch.

For more information on Rapid Application Development, refer https://kissflow.com/rad/rapid-application-development-framework-to-boost-app-development/

Show Slide 8



Instruction(s) to the trainer:
Using slide 8, explain Laravel follows the Model–View–Controller (MVC) architecture. As Laravel follows MVC architecture, programming logic is divided into three categories as shown in the slide.

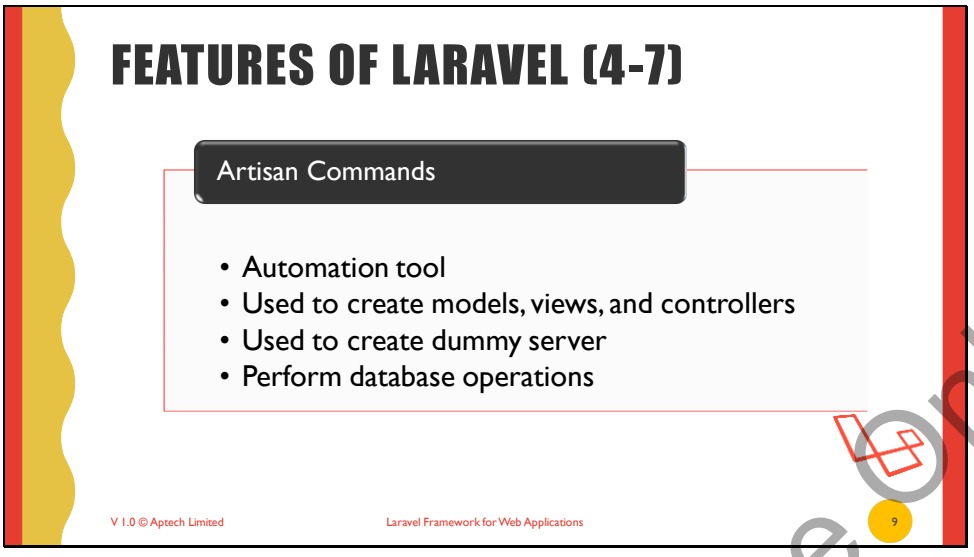
Models - Models define the way data is visualized and stored. Models are objects that represent resources utilized in an application. Usually, they are used to handle records in a database. Hence, models can be used to represent different entities, such as an employee or a user or an item. Models are objects that are instances of the Eloquent's base Model class. This object can be efficiently used to hold records from a database table and perform operations on it. The Eloquent ORM programming techniques allow models to interact more with records in a database.

Views - Views define the way Web pages appear. They are responsible for generating Web pages returned as responses with the help of a controller. They can be written as standard PHP files or can be dynamically built using the Blade Templating language, which gives a modular approach of creating them. Blade allows complex views to be created from simpler layouts.

Controllers - Controllers define the requisite responses for different requests. In simpler terms, controllers take a request and then, depending on the request generate appropriate responses.
Similar to a server, controllers are where the actual processing of requests takes place, such as handling form submissions and interacting with the databases.
In-Class Questions:
On which architecture Laravel is based on?
Answer: Model–View–Controller (MVC).

What are the logical parts of an MVC architecture?
Answer: Model, View, Controller.

Show Slide 9



Instruction(s) to the trainer:
Using slide 9, explain that the Artisan tool is included in the Laravel suite. As it is an automation tool, it helps to set up and configure various components for operations related to Web application development. It creates models, views, and controllers pre-filled with a template. It can also be used for automating Web browser.

Also, demonstrate few examples from students guide to the students for detailed understanding of Artisan command.

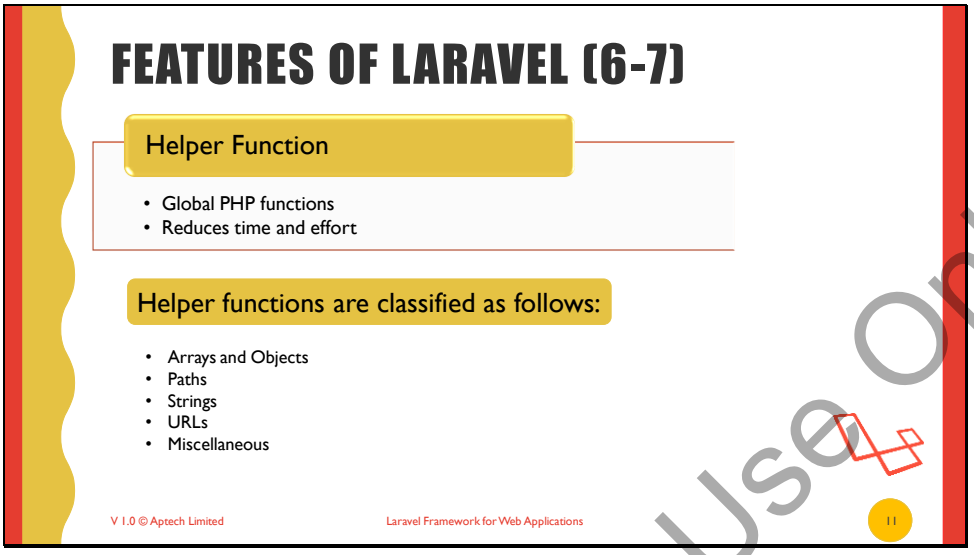
To know more about Artisan commands, refer:
https://laravel.com/docs/5.0/commands

Activity:
Ask students to create document, which list Artisan Commands with their usage and example for their future reference.

Show Slide 10

Using slide 10, display list of artisan commands available and explain frequently used commands to the students.

Show Slide 11



Instruction(s) to the trainer:
Using slide 11, explain what are helper functions and the need for helper functions. Helper functions are the global PHP functions included in Laravel. Without the helper functions, the developer may need to import different libraries. As the helper functions are included in Laravel, they help in reducing time and efforts of a developer.

The classification of the helper functions in Laravel is also shown in the slide. Explain each one of them in detail.
Arrays and Objects: Some operations include sorting values(Arr:sort()), getting the value of the first (head()) or last element (last()), and setting values at a position (Arr:set()).
Paths: Examples are path of the resource directory (resource_path()), path of the app directory (app_path()), and path of the configuration directory (config_path()).
Strings: For example, finding if it contains another substring(Str::contains()), and transforming the string into camel case(Str::camel()).
URLs: For instance, sanitizing a url into a secure url.
Miscellaneous: The operations that do not fit any categories are present in miscellaneous, such as finding values of environment variables (env()), finding current date (today()), and redirecting the user to another url (redirect()).

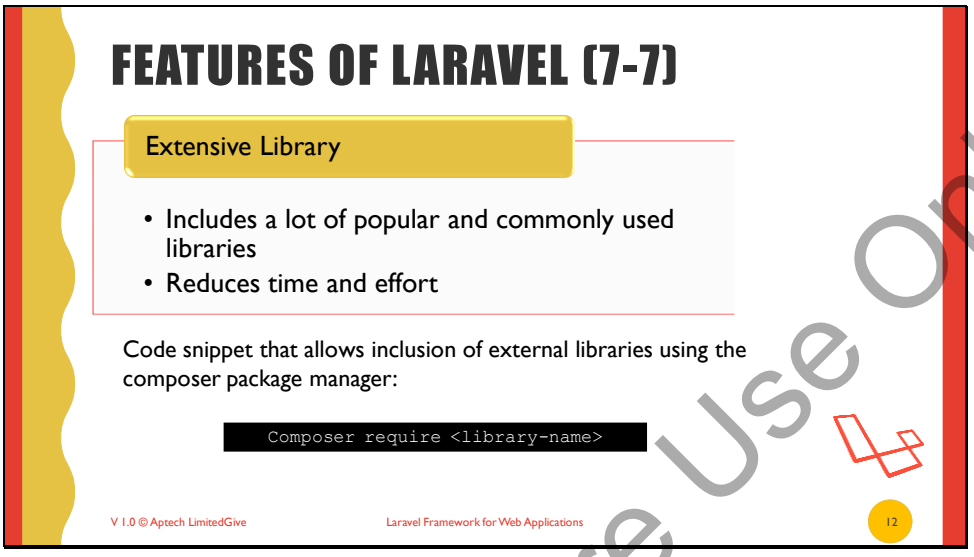
Also, demonstrate example of helper function to get clear understanding of it.

For more information on helper functions, refer https://laravel-news.com/creating-helpers

In-Class Questions:
How are helper functions useful in saving time and efforts of a Web developer?

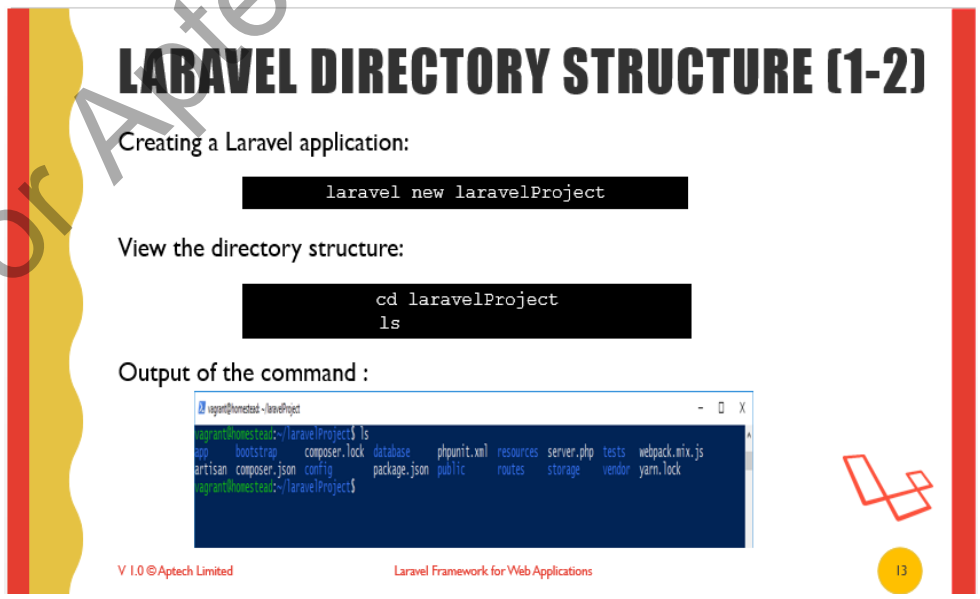
Answer: Many functions might be required during Web application development. This needs a developer to import different libraries. As these global PHP functions are already included in Laravel framework, the developers need not import these functions thus, saving time and effort.

Show Slide 12



Instruction(s) to the trainer:
Using slide 12, explain the importance of extensive library feature of Laravel. Without the extensive library feature, the libraries would have to be manually installed and configured when creating a Web Application.
Give example of authentication, which is used to authenticate users while taking care of security issues such as handling attacks that includes SQL Injection, Authentication bypass, and Cross-site Request forgery, encryption, and logging.

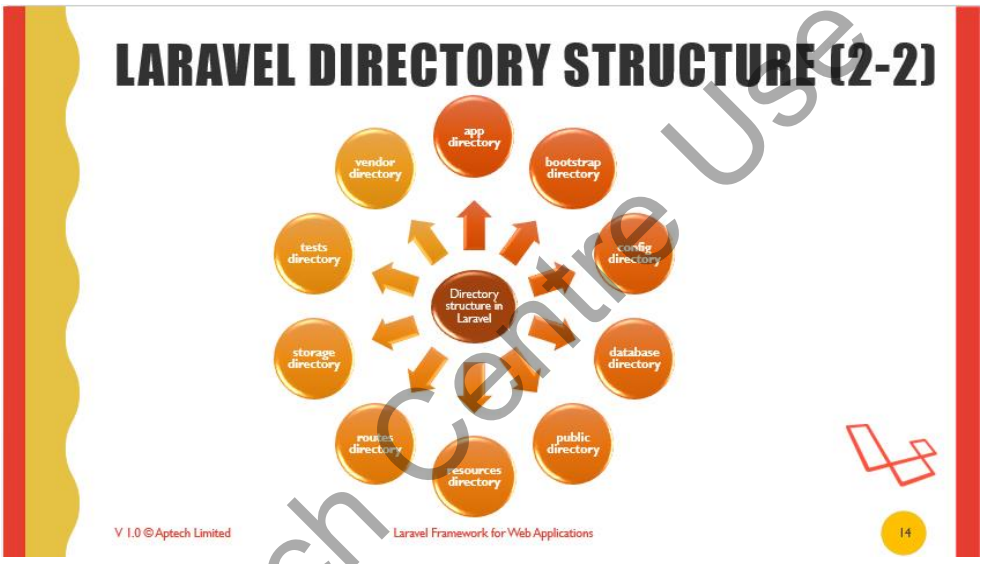
Show Slide 13



Instruction(s) to the trainer:
Using slide 13, show method to explore directory structure to students. Tell them that, for this, first they have to create a Laravel application by executing the command shown on the slide in the Homestead directory through ssh. This creates a new directory **laravelProject** which contains all the files and directories required for the Laravel Framework. Also, tell them how to view the directory structure and the output that shows the sample directory structure as shown on the slide.

Ask:
Which command is used to view the directory structure?
Accept the correct responses and explain the view directory structure command in detail.

Show Slide 14

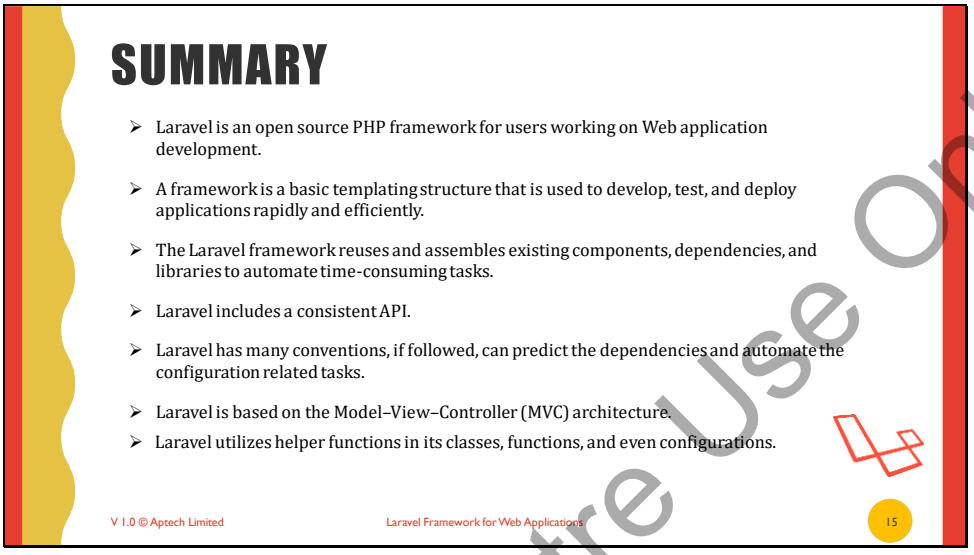


Instruction(s) to the trainer:
Using slide 14, illustrate the logical parts of a directory structure in Laravel.

- **app directory**: Contains the core code that runs the Laravel application. Most classes are stored in this directory and its subdirectories.
- **bootstrap directory**: Includes the code that collates all files spread throughout the directories and run the application.
- **config directory**: Includes files related to configuration of different components.
- **database directory**: Contains database migrations, seeds, model factories, and the SQLite database that will be used in subsequent sessions.
- **public directory**: Includes files that must be exposed to the Internet. Thus, the index.php file calls the app.php file in the bootstrap directory to start the application whenever it receives a request. All HTML, CSS, and Image files are also stored in this directory.
- **resources directory**: Includes the views that will be used to display the final Web page as well as the un-compiled assets related to JavaScript and CSS.
- **routes directory**: Provides the route definition for the application that will match the request with its response.

- **storage directory**: Includes data generated by the Laravel framework for storage and cache.
- **tests directory**: Contains the test specifications for automated testing.
- **vendor directory**: Contains information on third party dependencies imported by the composer dependency manager.

Show Slide 15



Instruction(s) to the trainer:

Using slide 15, summarize the key concepts taught in this session.

2.2 Post Class Activities for Faculty

You should familiarize yourself with the topics of the next session.

Tips: You can also check the Articles/Blogs/Expert Videos uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session. You can also connect to online tutors on the OnlineVarsity site to ask queries related to the sessions.

Session 3: Writing a Simple Application

3.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. You should revisit topics of the previous session for a brief review. Here, you can ask students to recall the key topics from previous session. Prepare a question or two that will be a key point to relate the current session objectives.

3.1.1 Teaching Skills

To teach this session, you should be well-versed with creating a simple Web application with Laravel. You should be able to explain how to use routes to serve different Web pages. Also, you should be able to tell them how views can enhance presentation of an application, how to use the Blade templating engine, and explain them how to add Master Template and Nested Views.

For teaching in the class, you are expected to use slides and LCD projectors.

Tips:

It is recommended that you test the understanding of the students by asking questions in between the class.

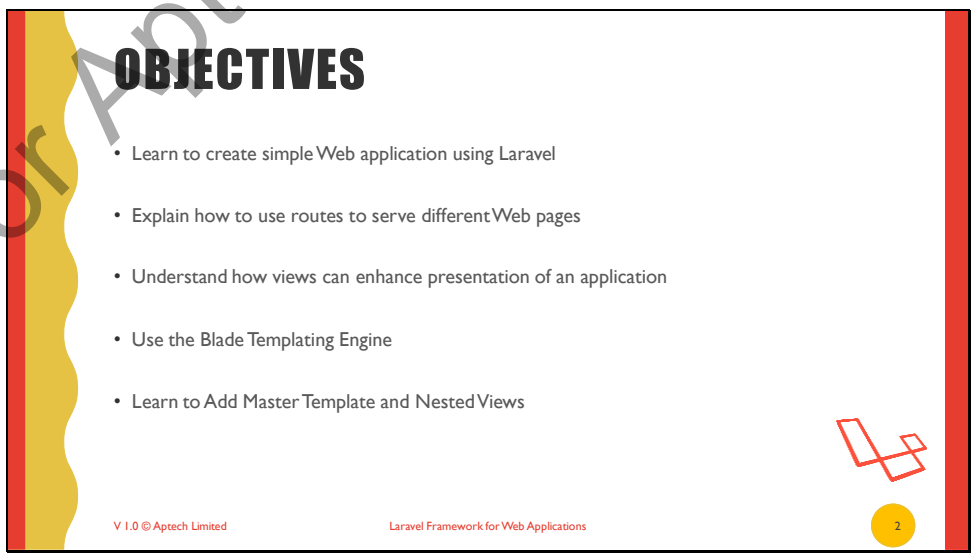
In-Class Activities

Follow the order given here during In-Class activities:

Overview of the Session

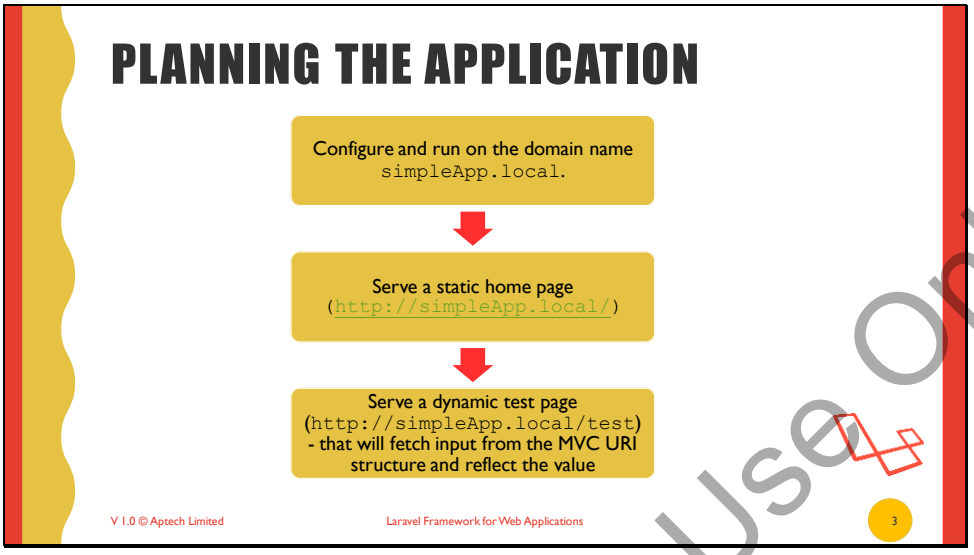
Give the students an overview of the current session in the form of session objectives. Show slide 2 of the presentation to students.

Show Slide 2



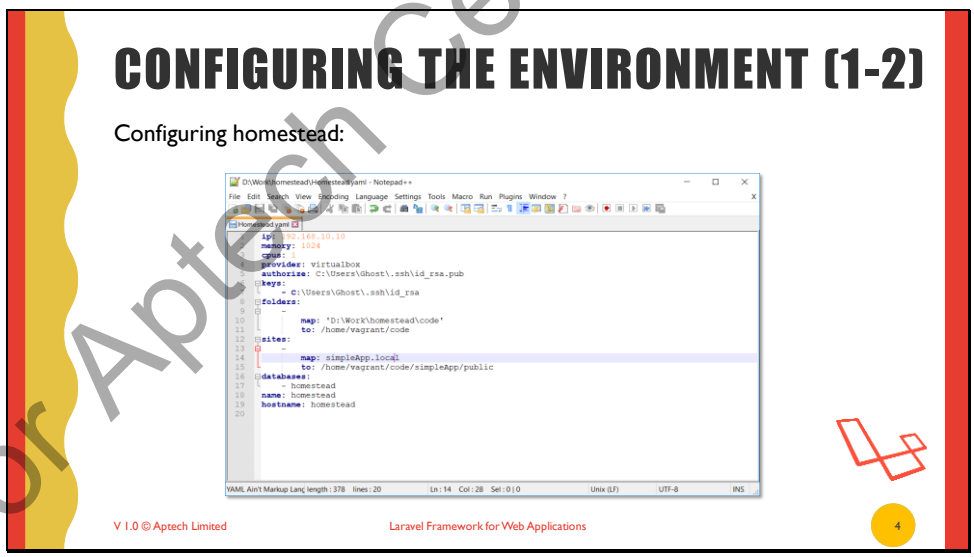
Instruction(s) to the trainer:
Using slide 2, give a brief introduction about this session.

Show Slide 3



Instruction(s) to the trainer:
Using slide 3, discuss the steps shown in the slide to develop a new Laravel application called simpleApp.

Show Slide 4



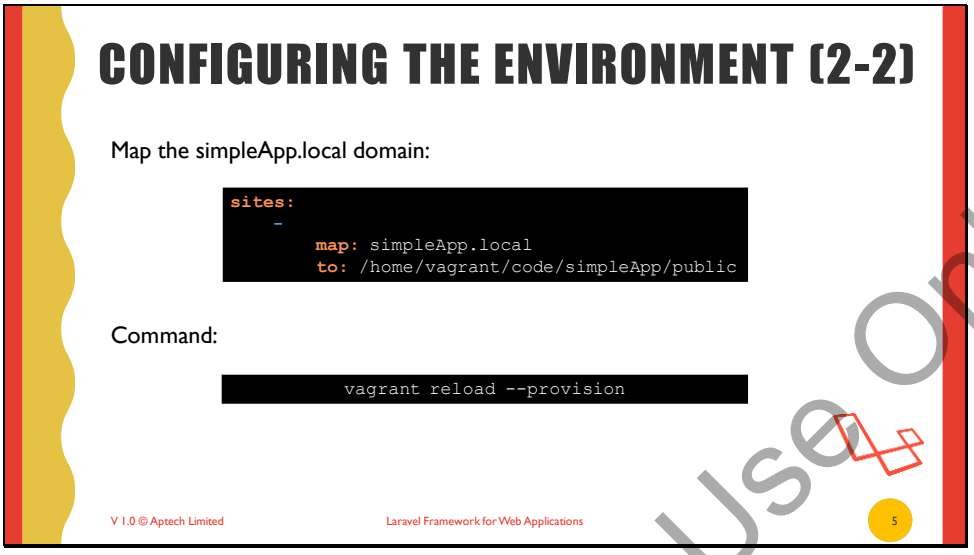
Instruction(s) to the trainer:
Using slide 4, explain step 1 to configure the homestead environment.

Step 1: Modify the Homestead.yaml file to reflect the changes as shown in the image in the slide.

Configuration file is in the Homestead directory.

Activity:
Ask the students to create a document explaining Laravel Homestead in detail.

Show Slide 5



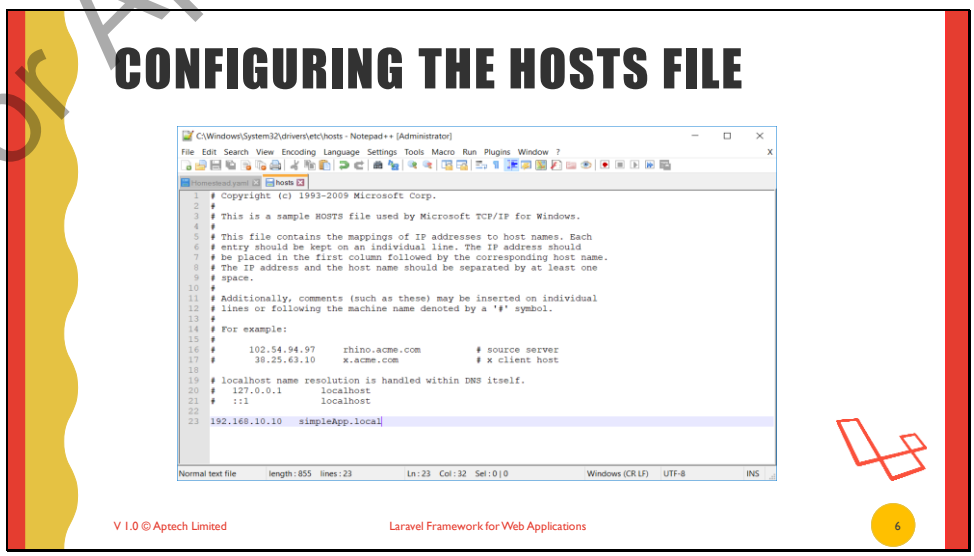
Instruction(s) to the trainer:
Using slide 5, explain Step 2, Step 3, and Step 4 for configuring homestead environment.

Step 2: To map the simpleApp.local domain to the /home/vagrant/code/simpleApp/public path in the Homestead environment as shown on slide.

Step 3: Save the updates to the file.

Step 4: Open a terminal in the same folder and type the command as shown on slide.

Show Slide 6



Instruction(s) to the trainer:
Using slide 6, explain the steps required to configure the hosts file. Explain the meaning of hosts file to the students.

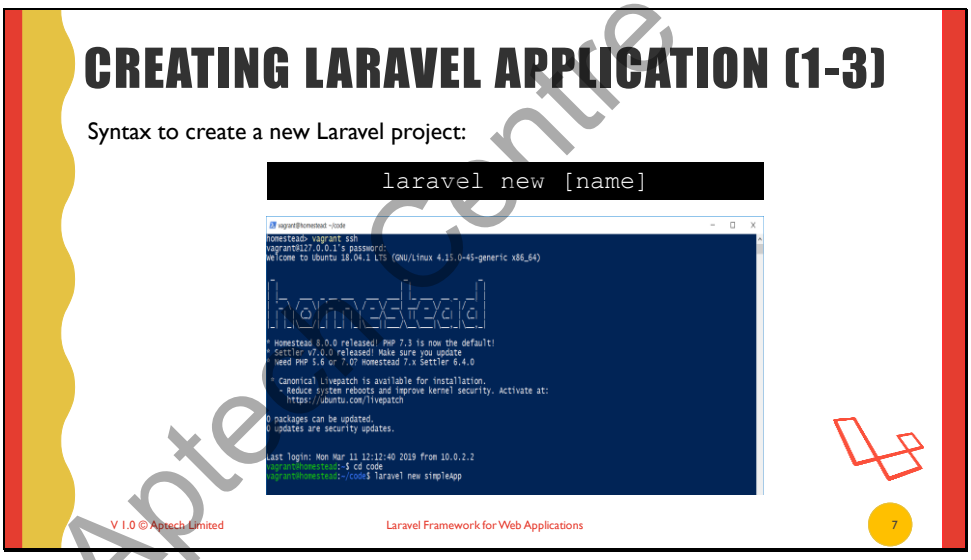
To save changes to the hosts file, open a text editor with administrator privileges using the following steps:

Step 1: Right-click the editor in the **Start** menu.
Step 2: Click **Run as Administrator**. The image in the slide displays the sample hosts file that appears.

For more information on hosts file modification, refer:
https://www.inmotionhosting.com/support/website/how-to/modifying-your-hosts-file

Ask:
What is **hosts** file?

Show Slide 7



Instruction(s) to the trainer:

Using slides 7 to 9, explain the steps to create a Laravel application.

Step 1: To SSH into the Homestead environment, type **vagrant SSH**. All vagrant related code should be executed in the Homestead directory. The interface to the Linux command line on the Homestead box appears.

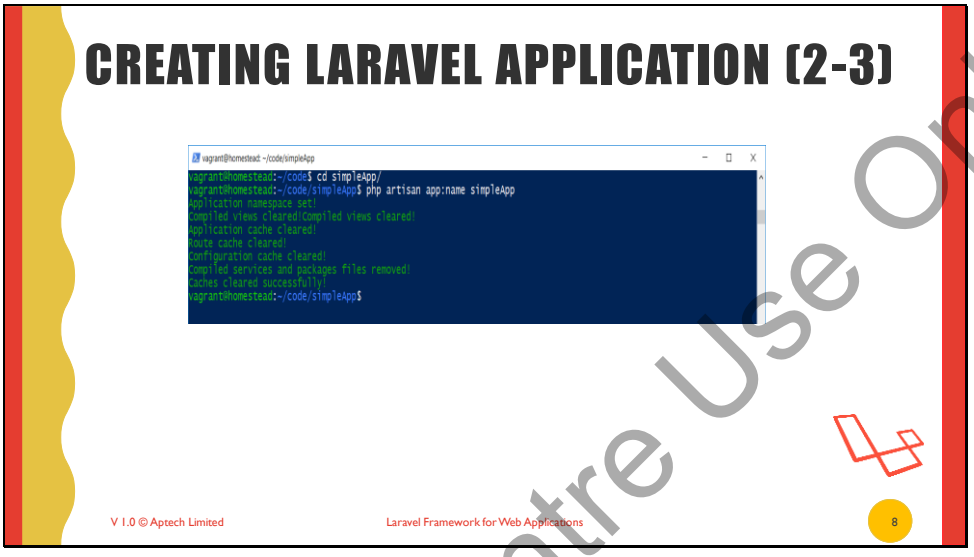
Step 2: The syntax to create a new Laravel project is as shown in the slide. A new folder called **name** is created. Although the name of the folder is not significant, the Laravel application should be inside the folder with path **/home/vagrant/code/simpleApp/.**

To create the application at the required path, use the code snippet:

cd code
laravel new simpleApp

The new Laravel project is created as displayed in the slide.

Show Slide 8

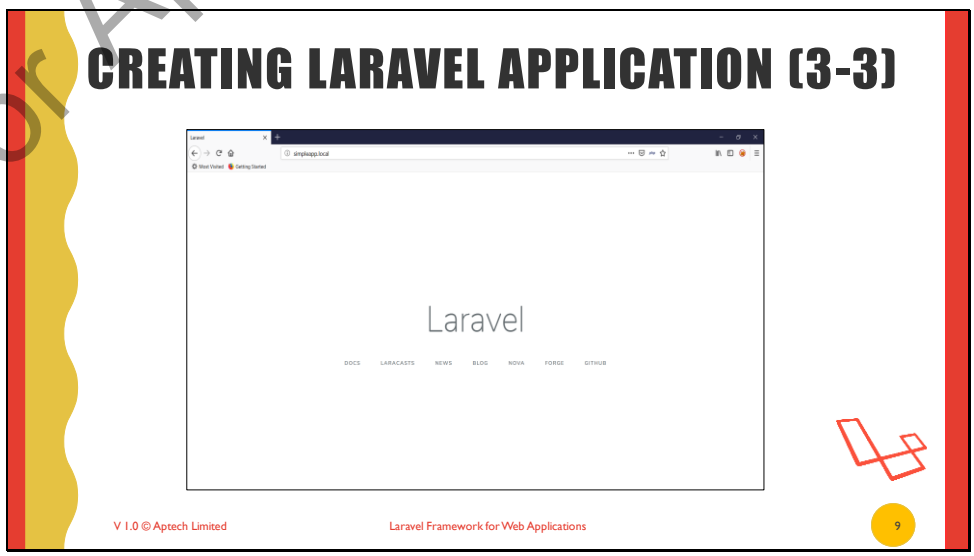


Instruction(s) to the trainer:
Step 3: To name the application, use the command as shown in the Code Snippet:

cd simpleApp
php artisan app:name simpleApp

The image in the slide displays the application renamed as **simpleApp**, which will be used for the PHP namespace.

Show Slide 9



Instruction(s) to the trainer:

Step 4: To view the running Web application, open the browser and navigate to http://simpleApp.local. The image in the slide shows the default welcome page.

Show Slide 10

Instruction(s) to the trainer:
Using slide 10, explain routing in Laravel. Mention that Web based routes are located at **simpleApp\routes\Web.php**. Each Laravel application includes the following pre-configured route as shown in Code Snippet:

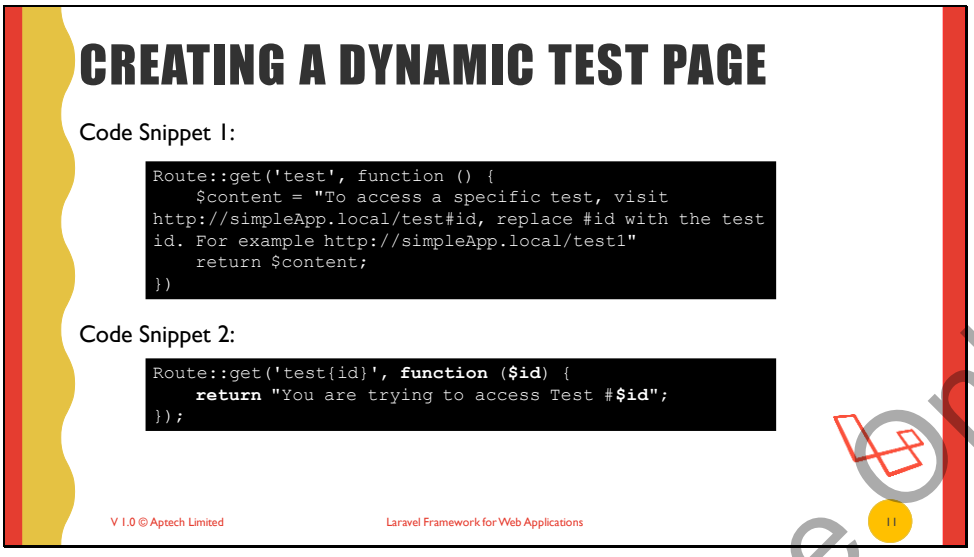
```
Route::get('/',function () {
 return view('welcome');
});
```

By default, Laravel has a Route class, which has more than one method to handle each type of HTTP request, such as get and put. The get method uses the URI, two arguments, and the closure function that is executed in response to the URI.

For more information on routing, refer:
https://laravel.com/docs/5.8/routing

Activity:
Ask the students to form groups and give a presentation on the concept of routing.

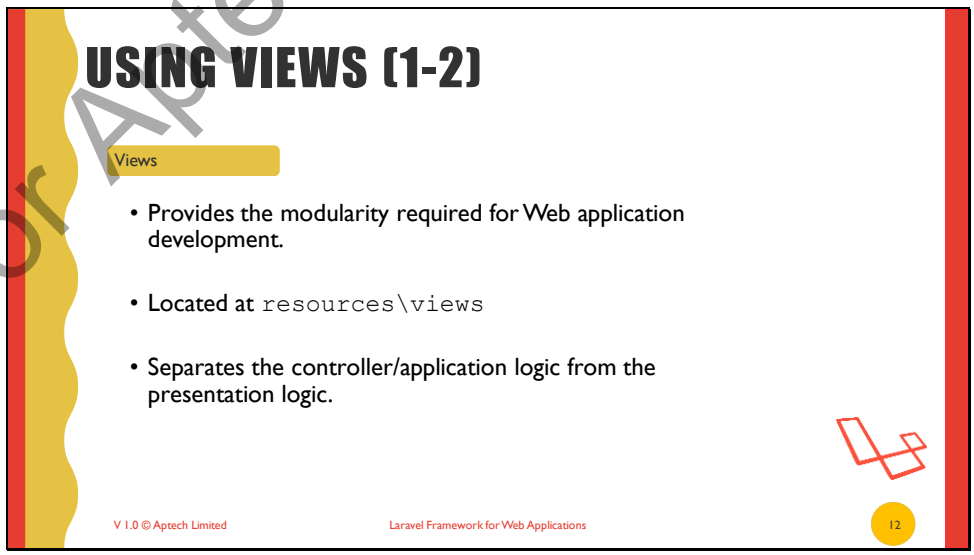
Show Slide 11



Instruction(s) to the trainer:
Using slide 11, show the steps to create a dynamic Web page. The steps involved are as follows:

Step 1: As shown in the first code snippet in the slide, add another route in the **Web.php** file.
Step 2: After saving the file, type **http://simpleApp.local/test** in the browser. There are two Web pages that have been created by editing a few configuration files.
Step 3: Add the route to serve multiple pages on the server as shown in the second Code Snippet in this slide.
Step 4: After saving the file, open **/test198, /test202**, and any other combination for **http://simpleApp.local/test101**.

Show Slide 12

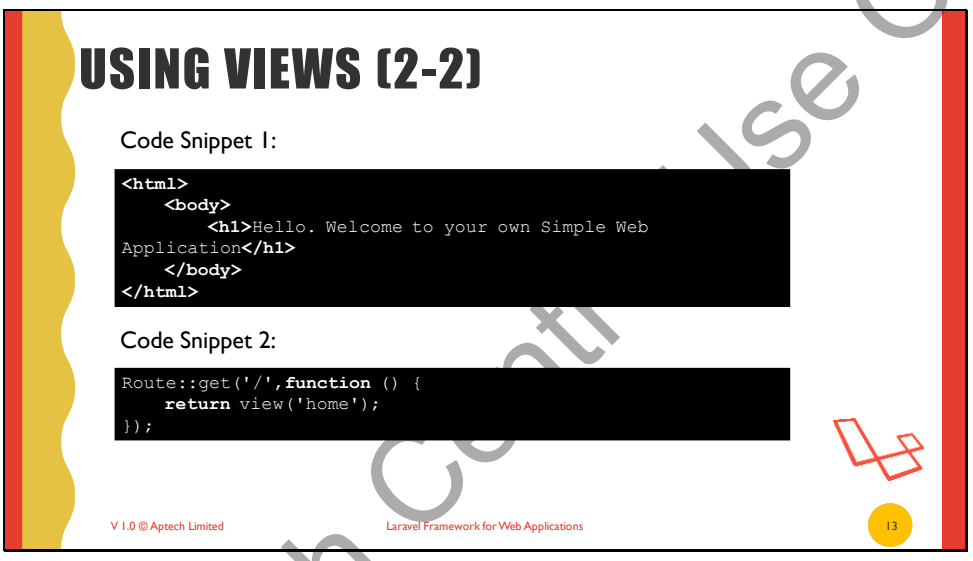


Instruction(s) to the trainer:
Using slide 12, explain views in detail. Mention that in MVC framework V stands for view. The Views that are stored in the simpleApp\resources\views\ directory can be used to present the PHP and HTML code, so that the Web application uses the route file (routes\Web.php) to define the routes.

For more information on views, refer:
https://laravel.com/docs/5.8/views

Ask:
In which directory are Views stored?

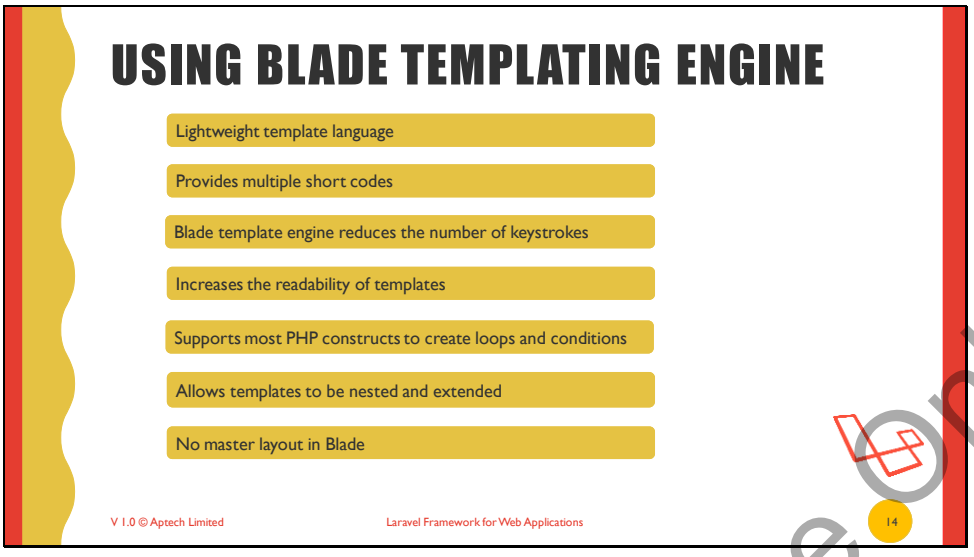
Show Slide 13



Instruction(s) to the trainer:
Using slide 13, show the steps to create a view for the Home page:

Step 1: Open notepad++ and create a new file using the Code Snippet 1 as shown on the slide.
Step 2: Save the file as **home.php** at **resource\views**.
Step 3: Replace the get method for URI \ in the Web routes **Web.php** file as shown in Code Snippet 2 of the slide.

Show Slide 14



Instruction(s) to the trainer:
Using slide 14, explain blade templates to the students. Explain that Blade is a simple but powerful templating engine of Laravel. Unlike other PHP templating engines, Blade does not add any overheads to the application. The Blade template engine reduces the number of characters when coding. This means, it simplifies PHP scripts. For example, the blade engine has a shortcode {{ $output }} for PHP code <?php echo htmlentities($output); ?>.

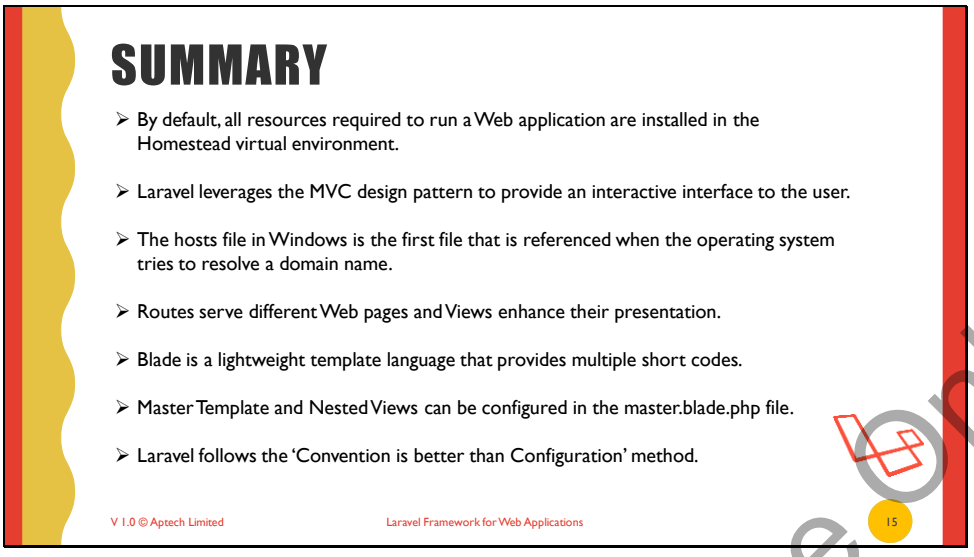
Mention that although there is no master layout in Blade, it is possible to add master layout by creating a **master.blade.php** file in the **views** directory. Then, use **extend** blade directive in other templates to make it a master layout.

For more information on blade templates, refer:
https://www.cloudways.com/blog/create-laravel-blade-layout/
https://laravel.com/docs/5.8/blade

Activity:
Ask the students to find out more Blade codes that replaces PHP codes.

Ask:
What are the benefits of using Blade engine?

Show Slide 15



Instruction(s) to the trainer:
Using slide 15, summarize the key concepts taught in this session.

3.2 Post Class Activities for Faculty

You should familiarize yourself with the topics of the next session.

Tips: You can also check the Articles/Blogs/Expert Videos uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session. You can also connect to online tutors on the OnlineVarsity site to ask queries related to the sessions.

Session 4: Creating a Web Application

4.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. You should revisit topics of the previous session for a brief review. Here, you can ask students to recall the key topics from previous session. Prepare a question or two that will be a key point to relate the current session objectives.

4.1.1 Teaching Skills

To teach this session, you should be well-versed with creating a new Laravel application. You should be able to explain the integration of Laravel application with database. Also, you should be able to explain them how to create table schema and describe process migration. In addition, you should be able to explain them Web page creation using Master Blade Layout, Forms, and Routes.

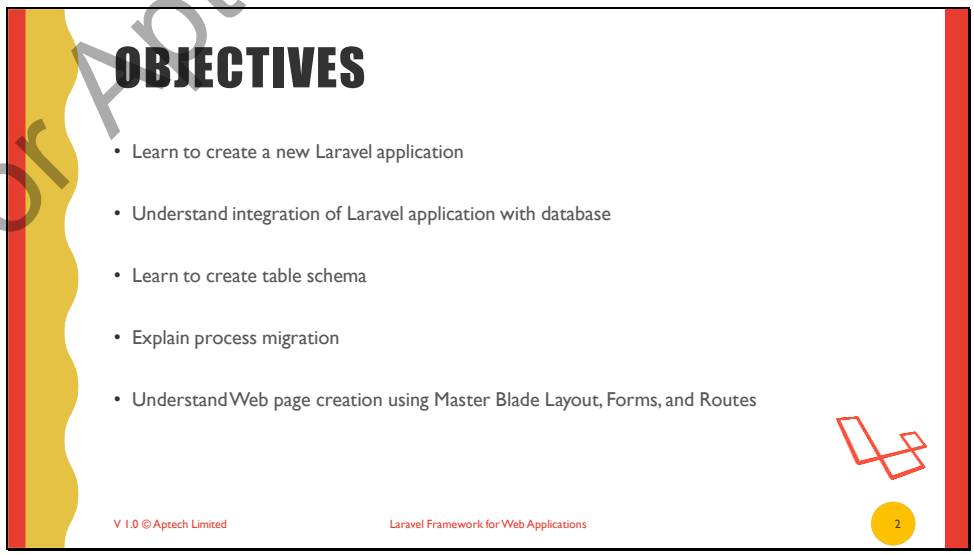
For teaching in the class, you are expected to use slides and LCD projectors.

Tips:
It is recommended that you test the understanding of the students by asking questions in between the class.

In-Class Activities
Follow the order given here during In-Class activities:

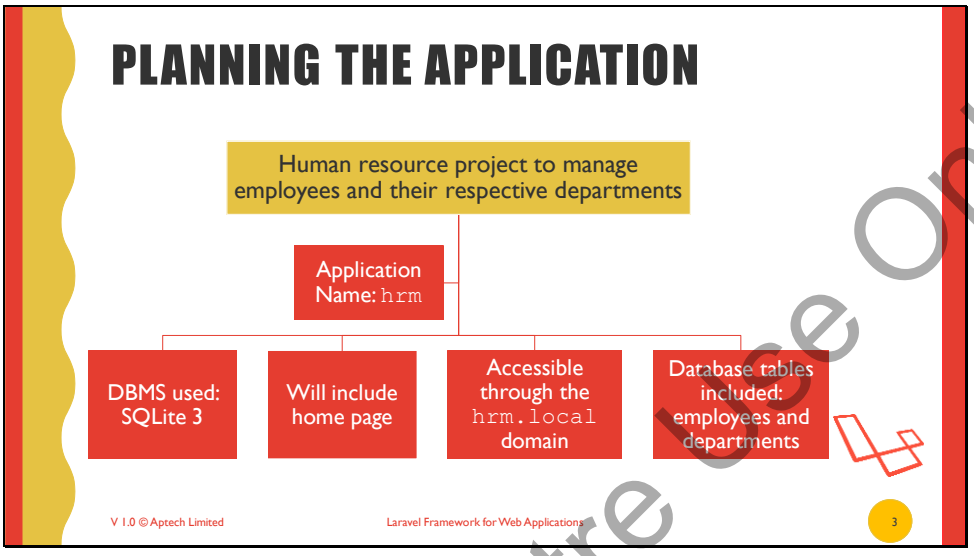
Overview of the Session
Give the students an overview of the current session in the form of session objectives. Show the students slide 2 of the presentation.

Show Slide 2



Instruction(s) to the trainer:
Using slide 2, give a brief introduction about this session to the students. Explain each objective in detail.

Show Slide 3



Instruction(s) to the trainer:
Using slide 3, discuss the application that is to be developed. Explain the students that before developing the actual project, proper planning needs to be done.

Here an application for a human resource project is considered. The project will help to manage employees and the department they work for.

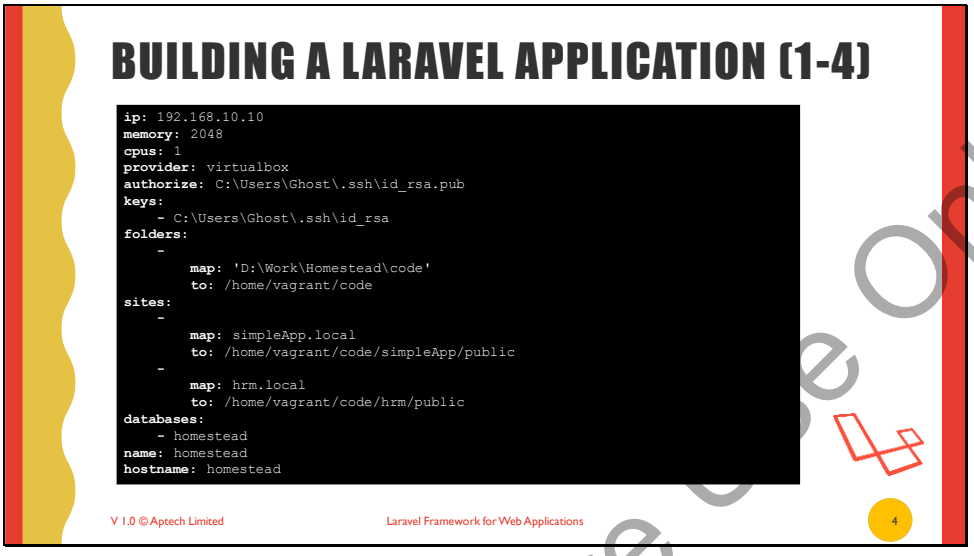
Give the explanation of the project as follows:
As shown in the slide, the name of the application is **hrm**. SQLite 3 will be used as the Database Management System (DBMS) for this application. Home page created is such that it will have links to other pages such as Add, Display, Update, and Delete. These pages will implement CRUD operations respectively and will be accessible through the hrm.local domain. There will be two database tables: employees and departments. The employees table will include columns for ID, Name, Department, and Salary. The departments table will have Department ID, Department Name, Manager Name, and Manager ID.

Additional information:
The term CRUD operations is short for Create, Read, Update, and Delete operations.

Activity:
Ask the students to form groups and layout a plan for developing a student progress report using CRUD operations. The students should be able to discuss the same in class.

For more information on CRUD operations, refer https://stackify.com/what-are-crud-operations/

Show Slide 4



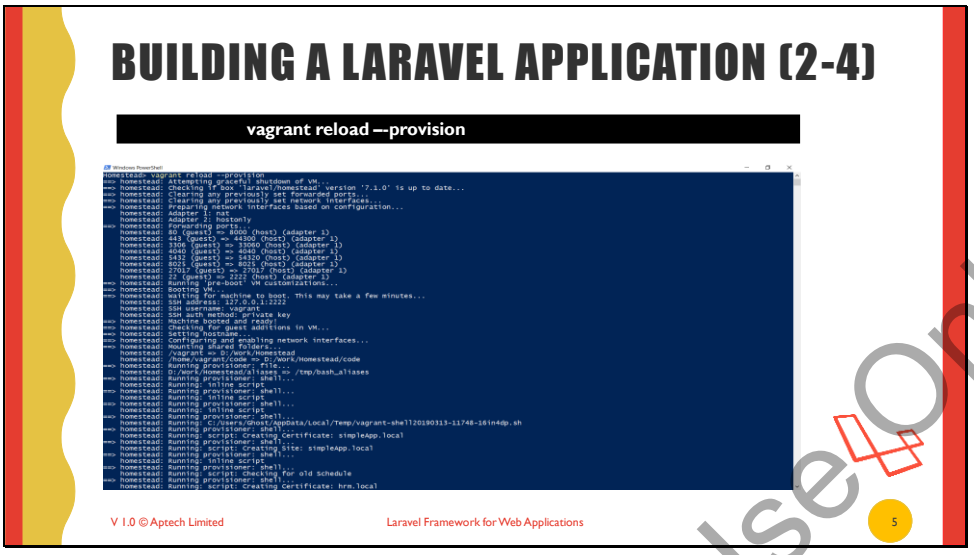
Instruction(s) to the trainer:
Using slides 4 to 7, explain the steps involve in building Laravel application. The steps are as follows:

Step 1: Edit the **Homestead.yaml** file to map **hrm.local** to the **Homestead/code/hrm/public** folder as shown in the slide and save the changes.

Ask:
What is Laravel Homestead?

Accept the correct answers and then describe in detail Laravel Homestead and its features.

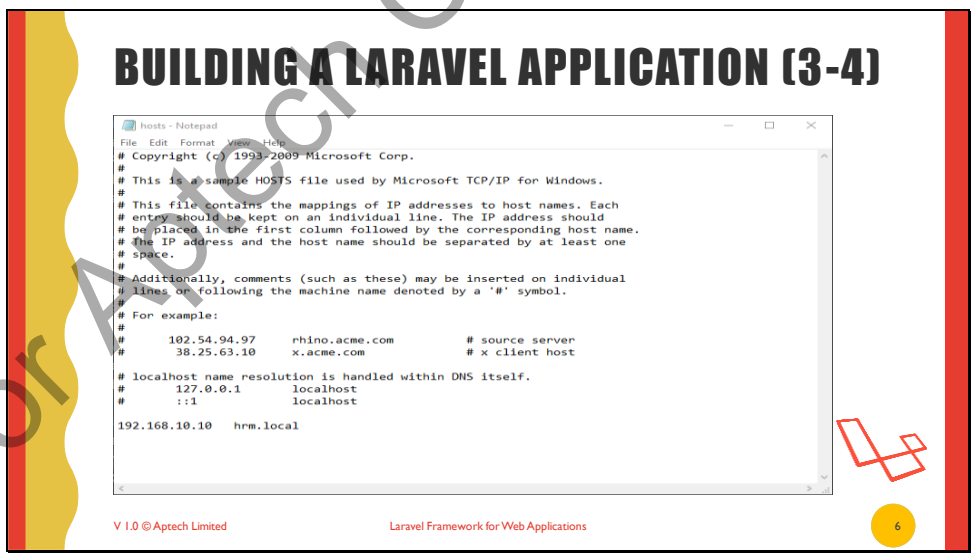
Show Slide 5



Instruction(s) to the trainer:
Using slide 5, explain the next step in building Laravel application.

Step 2: To reload provision in homestead, run **vagrant reload –provision** command as shown in slide 5 in a terminal inside the Homestead directory. The output is as shown in the slide.

Show Slide 6



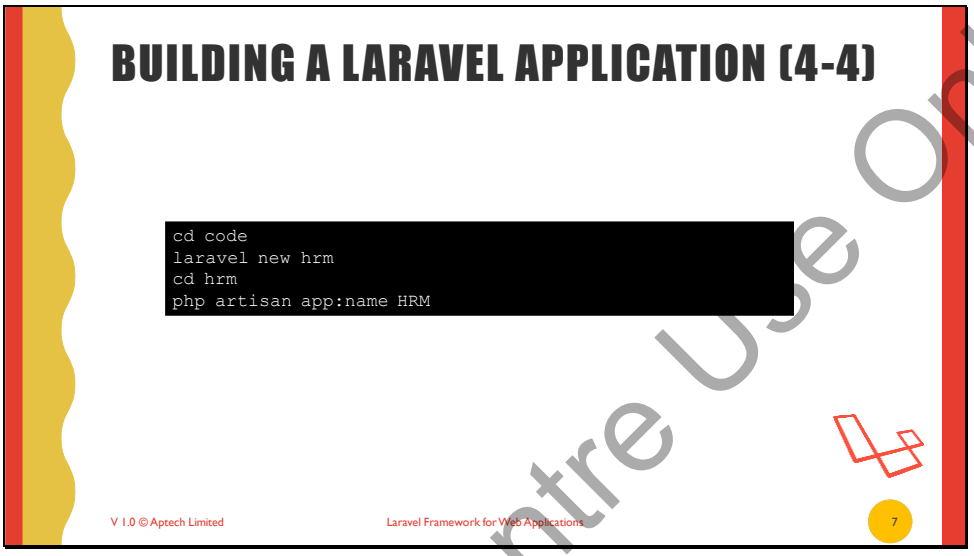
Instruction(s) to the trainer:
Using slide 6, explain next step involved in building the Laravel application:

Step 3: To add the mapping of the IP address, add an entry for **hrm.local** in the hosts file corresponding to the IP 192.168.10.10 as shown in slide 6.

Explain them in detail how to map IP address.

Ask:
In which file they have to make entry, while mapping IP address?
Accept the correct responses and explain the details of hosts file.

Show Slide 7



Instruction(s) to the trainer:
Using slide 7, explain the remaining steps for building Laravel application:

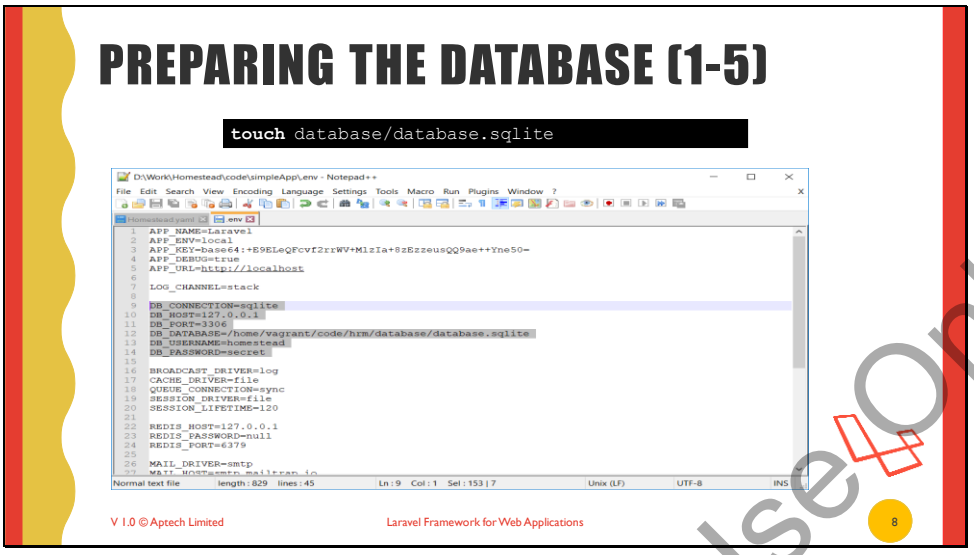
Step 4: To ssh into the vagrant machine, use the command **vagrant ssh**.
Step 5: To create a new Laravel application in the hrm folder, navigate to the code directory as shown in slide 7. The **php artisan app:name HRM** in the code snippet names the application as HRM.
Step 6: To view the default Laravel page, browse to **http://hrm.local**.

For more information on building Laravel application, you can refer to the following links:

https://laravel-news.com/your-first-laravel-application

https://www.codementor.io/syedikramshah/step-by-step-guide-to-building-your-first-laravel-application-9sd0fttcc

Show Slide 8



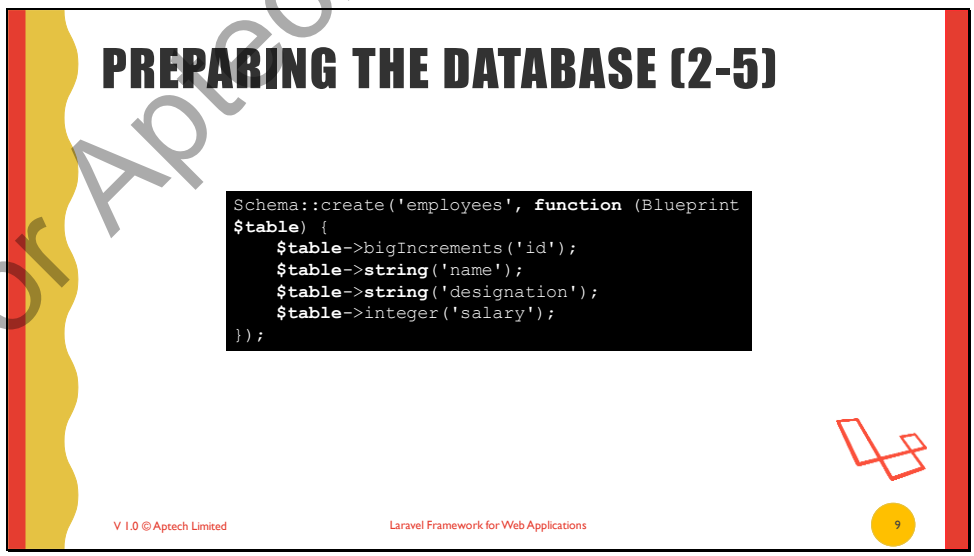
Instruction(s) to the trainer:

Using slide 8, explain the code snippet to the students for creating a blank file. Tell them that this blank file is created before configuring the newly created Laravel application as SQLite dumps the entire database in a file. Next open the **.env** file in the **hrm** directory in notepad++ and edit the highlighted lines as shown in the slide. The application is now configured to use SQLite as its DBMS software.

Activity:

Ask the students to prepare a document explaining SQLite in detail.

Show Slide 9



Instruction(s) to the trainer:
Using slide 9, explain to the students how to create a table schema. Explain that schema is the structure of the table that will be created for the application. The Schema builder class in Laravel helps in creating tables. In addition, the DB query builder class helps in implementing CRUD operations. They also perform other operations without any SQL code written specifically for the purpose. Slide 9 shows the code snippet to create the employees table using the Schema builder class. The Schema class instance takes in two arguments - table name and a closure function containing the structure of the table. The $table instance defines the structure of the table.

Explain the example code in detail to the students. The employees table will have six columns such as ID, Name, Designation, Salary, Created at, and Updated at. The last two columns are created automatically by Laravel. The created at column will contain the timestamp of the time and date at which the database was created. The Updated at column will always contain the timestamp of the time and data when the table was last modified.

Additional information:

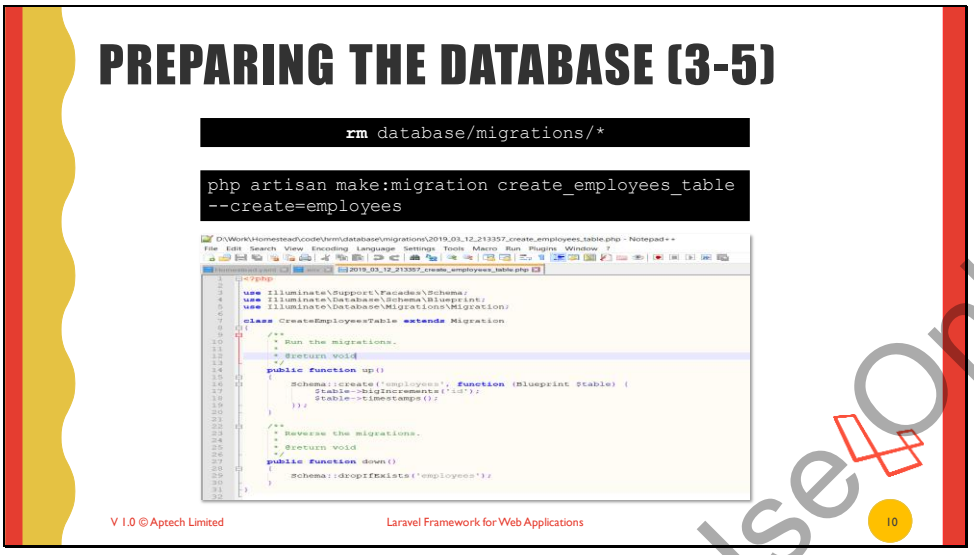
Laravel, by default, constraints the autoIncremental column as the primary key.

In-Class Questions:
Q1. What is the function of $table?
Answer: The function of $table instance is to define the structure of the table.

Q2. Name the two arguments schema class.
Answer: The Schema class instance takes in two arguments - table name and a closure function containing the structure of the table.

For more information on schema, refer https://laravel.com/docs/5.0/schema

Show Slide 10



Instruction(s) to the trainer:
Using slide 10, explain migrations features of Laravel in detail. It acts as a version control system for a database that includes many tables. Migrations construct, reconstruct, and deconstruct tables and their schemas. They are primarily used when shifting one DBMS system to another or for creating a backup of systems. Migration can be implemented by editing the .env file and running the migrations command.

The first code snippet shown in slide 10 is used to delete the migrations that have been shipped with Laravel. The second code snippet is used to create a new migration. [timestamp]_create_employees_table at the database/migrations/ location. The migration includes a pre-written code for creating the employees table using the --create=employees parameter.

The migration file includes the following two methods:
up() - This method is used to create a table.
down() - This method is used to drop the table.

Additional information:

By **convention**, routes are the preferred way to create records, not tables. Tables must be created using the Migration method.

In-Class Questions:
Q1. How are migrations implemented?
Ans: Migration can be implemented by editing the .env file and running the migrations command.

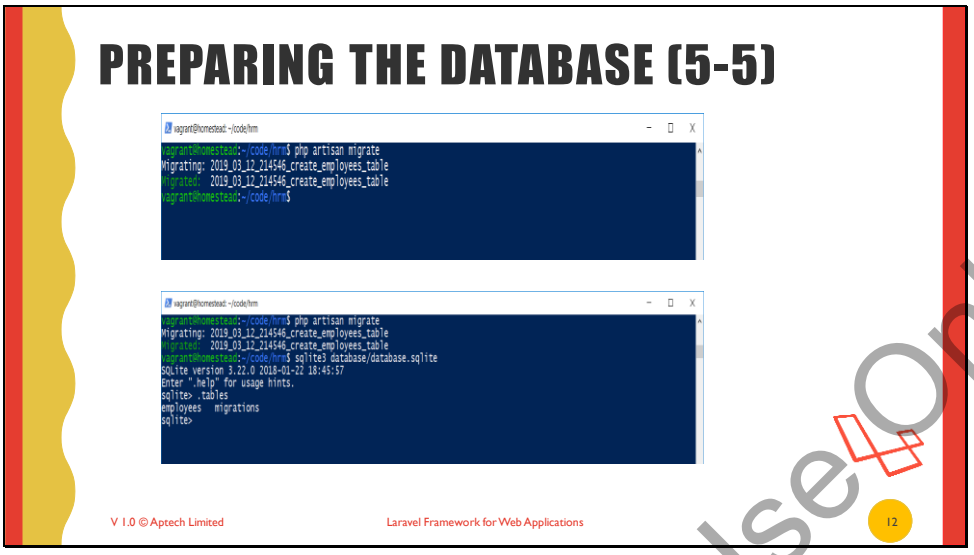
Activities:
Ask the students to prepare a presentation on the topic, Migration and give a short explanation on up() and down() methods.

For more information on migration features, refer https://laravel.com/docs/5.8/migrations and https://vegibit.com/what-are-migrations-in-laravel/

Show Slide 11

Instruction(s) to the trainer:
Using slide 11, explain the Schema builder code used in slide 9 can be passed in the up section to create the table. Using slide 11, show the final code. To create the table, execute this code and run the command **php artisan migrate.**

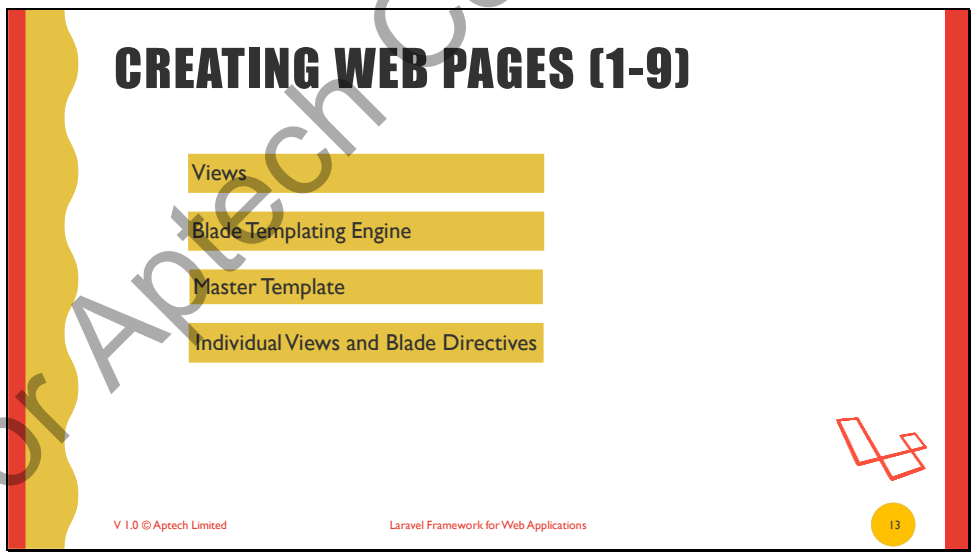
Show Slide 12



Instruction(s) to the trainer:

Using slide 12, explain that the first image in the slide shows successful migration. In order to verify that the table is created, enter the command as shown in the second image. It shows that employees table is created and ready for use. Next, repeat the procedure for table creation to create the departments table.

Show Slide 13



Instruction(s) to the trainer:

Using slide 13, explain each step in detail about creating Web pages. The Blade templating engine allows the views to be written more efficiently by providing a modular approach for writing Web pages through templates/layouts. Most Web pages have many components such as the Web Application logo, name, menu bar, navigation bar, header, footer, and many more.

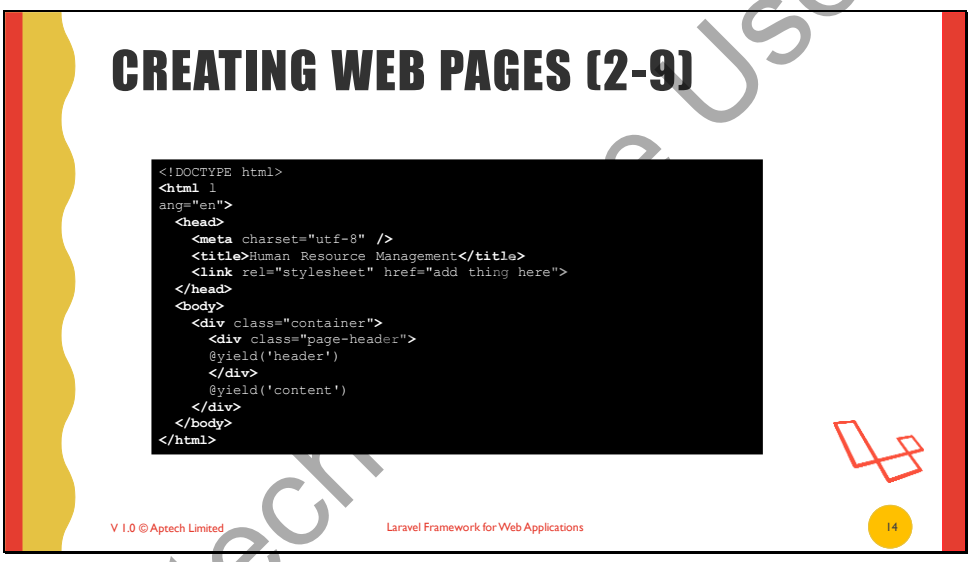
These basic contents of a Web application can be written in a single template called master template. This template can define sections that will be reused throughout the application and can embed contents from other templates. This page specific content is saved inside other templates.

Activity:

Ask the students to form groups and give presentations on Blade templating engine.

Ask:
Q1. What is a master template?
Q2. How does Blade templating engine allows to write Views more efficiently?

Show Slide 14



Instruction(s) to the trainer:
Using slide 14, show the steps involved in writing Master Blade layout.

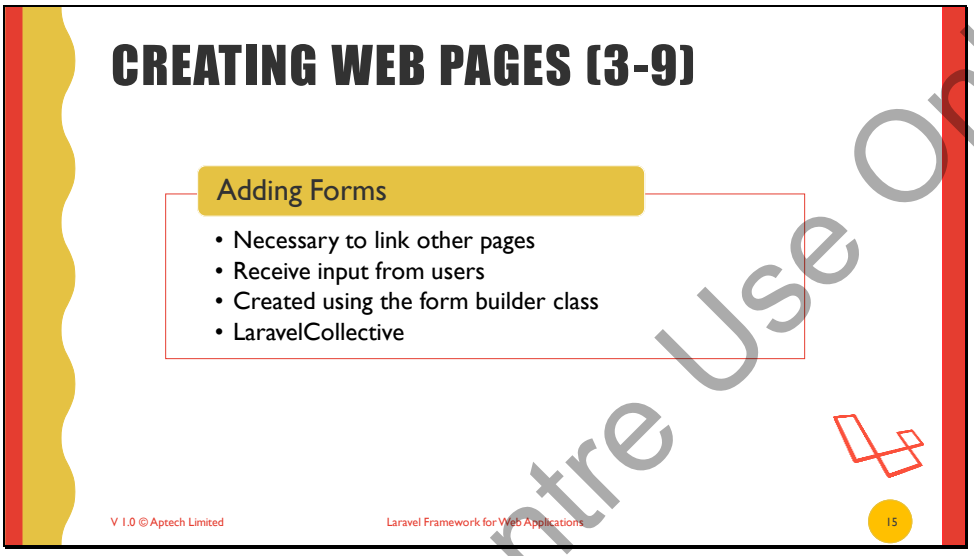
Step 1: To create a file, use the code as shown in the slide.
Step 2: Save the file at the location - resources/views/master.blade.php.
The Bootstrap CSS framework has a better UI/UX, which uses the asset() helper function. It helps to avoid writing of absolute paths and creates a working relative path, irrespective of the location. Currently, there are no bootstrap files in the Laravel app.

Step 3: Download the files from http://getbootstrap.com.
Step 4: Place the minified CSS file at public/css/.
Although the Master Blade template uses the yield directive, it is replaced by the content in the sections directive of child templates.

Additional information:
The **yield** directive is used inside master templates.

To know more about bootstrap, refer https://www.taniarascia.com/what-is-bootstrap-and-how-do-i-use-it/

Show Slide 15



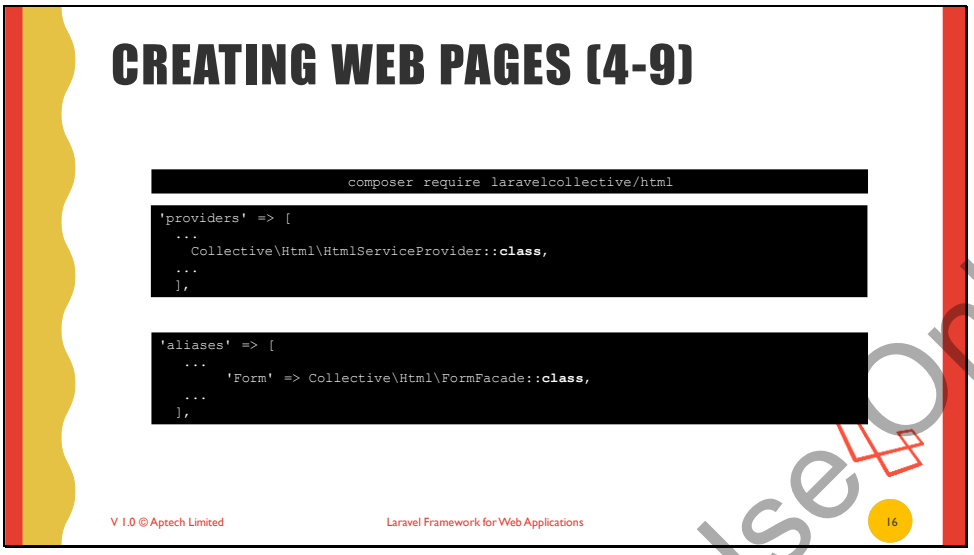
Instruction(s) to the trainer:
Using slide 15, explain how to use Forms in Web pages. Use this slide to give detailed explanation about the features and usefulness of forms in developing a Web application. Forms are used to link a static home page to other pages. Forms can be created using the form builder class, which is pre-installed in Laravel versions up to 4.x. Higher versions of Laravel are shipped with Advanced Blade directives. A repository called LaravelCollective is created by the open source community in order to help developers use the Form builder class.

To know more about forms, refer https://usersnap.com/blog/designing-web-forms-examples/
To gain more information on Laravel Collective, refer https://www.tutcodex.com/laravel-collective-html-form-builder/

Ask:
What is LaravelCollective?

Activity:
Ask the students to prepare a document providing all necessary information about Forms in a Web application.

Show Slide 16



Instruction(s) to the trainer:

Using slide 16, explain the steps involved to add the repository:

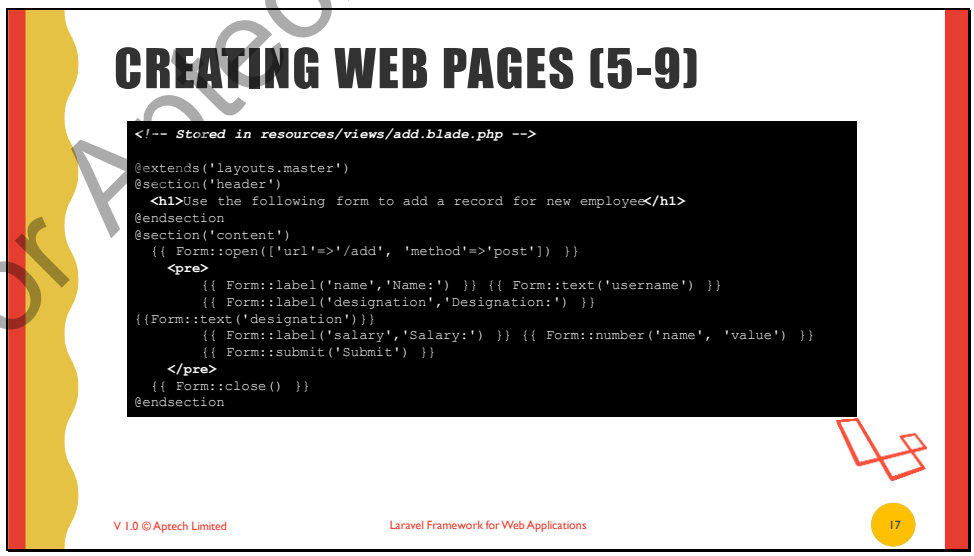
Step 1: To install the package, enter the command as shown in the first Code Snippet.

Step 2: To add a new provider, add HTMLServiceProvider to the list of providers in the config/app.php file as shown in second Code Snippet.

Step 3: To use classes anywhere, create an alias for Form in the config/app.php file as shown in the third Code Snippet.

Step 4: Create a blank form using the Form class. The Form::open() method writes a form. Form::close() denotes the end of the form.

Show Slide 17



Instruction(s) to the trainer:

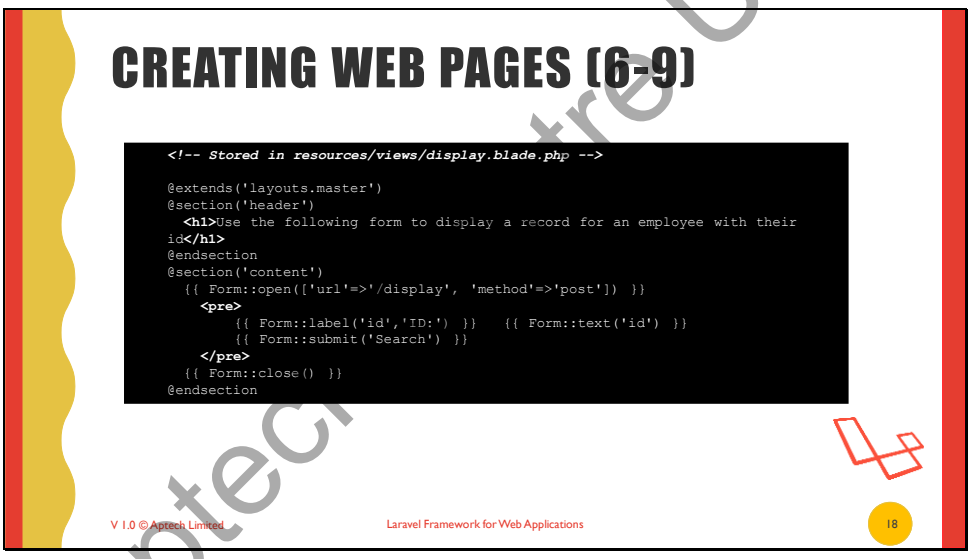
Using slide 17, explain creating Views for Forms.

Consider the following requirements for the Web application:

Home page
Add Employee Data (/add)
Displaying Employee Data (/display)
Modifying Employee Data (/edit)
Deleting Employee Data (delete)

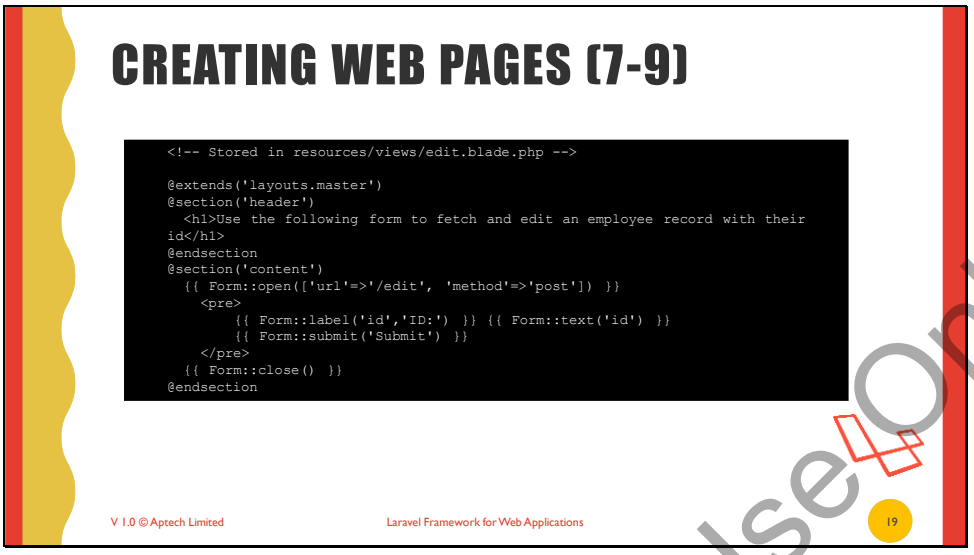
Tell them that slide 17 shows the code snippet to add a new employee. This has to be populated in **add.php**.

Show Slide 18



Instruction(s) to the trainer:
Using slide 18, show the code to be populated in **display.php.** This will display employee data. Explain the code line by line to the students.

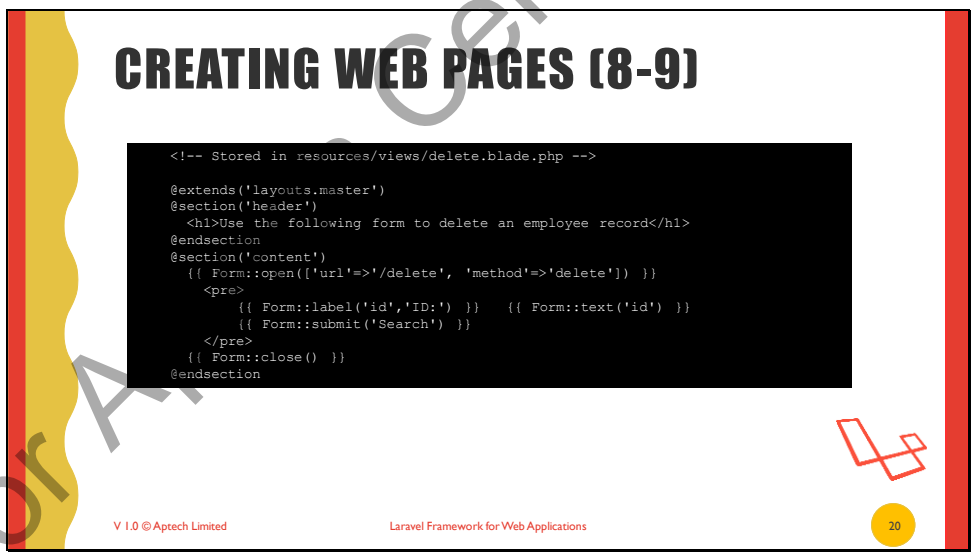
Show Slide 19



Instruction(s) to the trainer:

Using slide 19, show the code to be populated in **edit.php.** This is required to modify employee data. Explain the code line by line to the students.

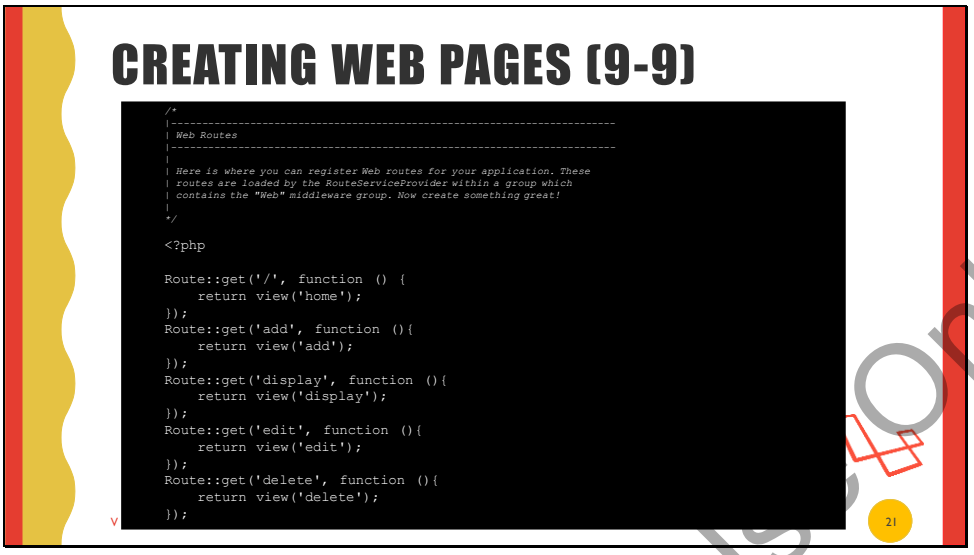
Show Slide 20



Instruction(s) to the trainer:

Using slide 20, show the code to be populated in **delete.php.** This is required to delete employee data. Explain the code line by line to the students.

Show Slide 21

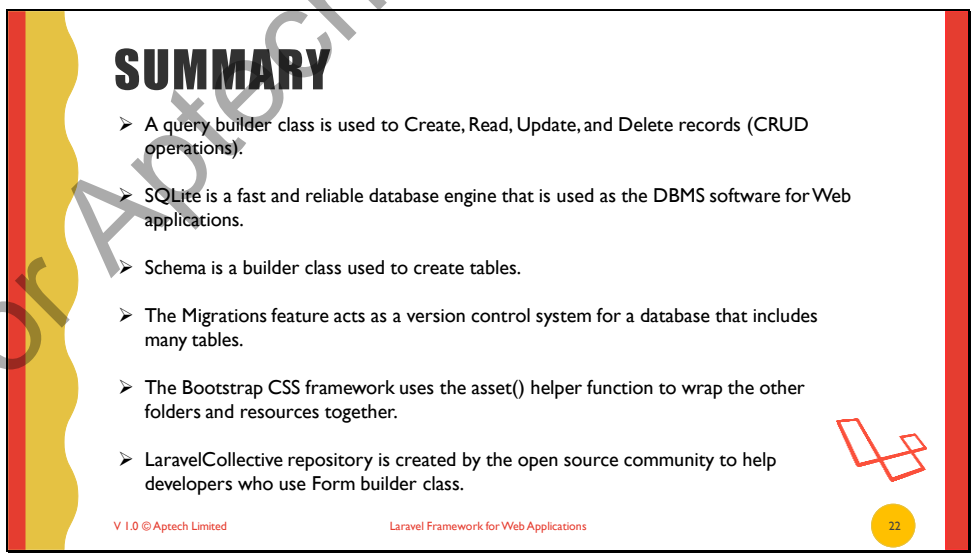


Instruction(s) to the trainer:

Using slide 21, explain adding temporary routes in order to verify that forms are created properly. Edit the **routes\Web.php** file and write the routes as shown in Code Snippet in this slide.

Activity:
Ask the students to write assignment on concept of routing in Laravel.

Show Slide 22



Instruction(s) to the trainer:

Using slide 22, summarize the key concepts taught in this session.

4.2 Post Class Activities for Faculty

You should familiarize yourself with the topics of the next session.

Tips: You can also check the Articles/Blogs/Expert Videos uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session. You can also connect to online tutors on the OnlineVarsity site to ask queries related to the sessions.

Session 5: The Eloquent ORM

5.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. You should revisit topics of the previous session for a brief review. Here, you can ask students to recall the key topics from previous session. Prepare a question or two that will be a key point to relate the current session objectives.

5.1.1 Teaching Skills

To teach this session, you should be well-versed with the functionalities offered by Eloquent Object Relational Mapping (ORM). You should be able to explain database operations using Eloquent ORM. Also, you should be able to tell them about database testing using Routes and Eloquent ORM.

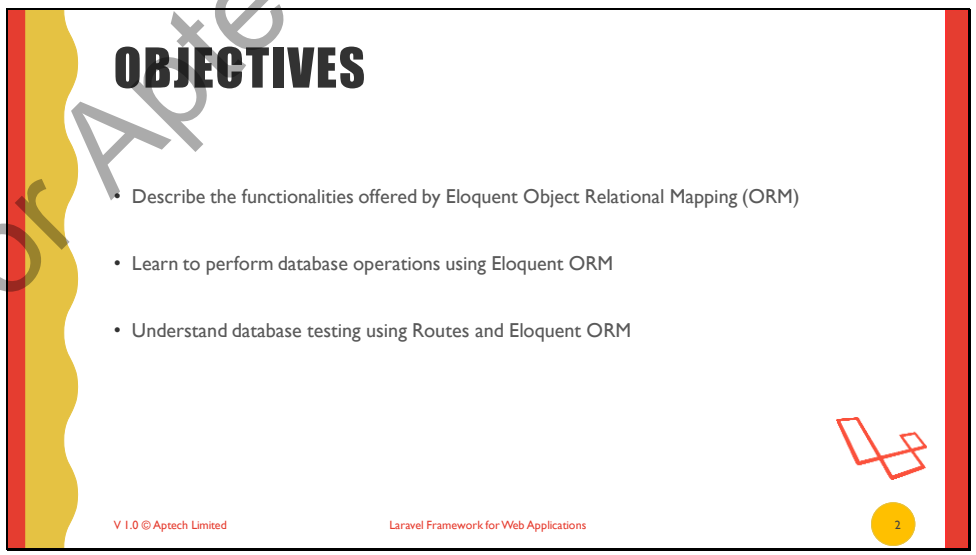
For teaching in the class, you are expected to use slides and LCD projectors.

Tips:
It is recommended that you test the understanding of the students by asking questions in between the class.

In-Class Activities
Follow the order given here during In-Class activities:

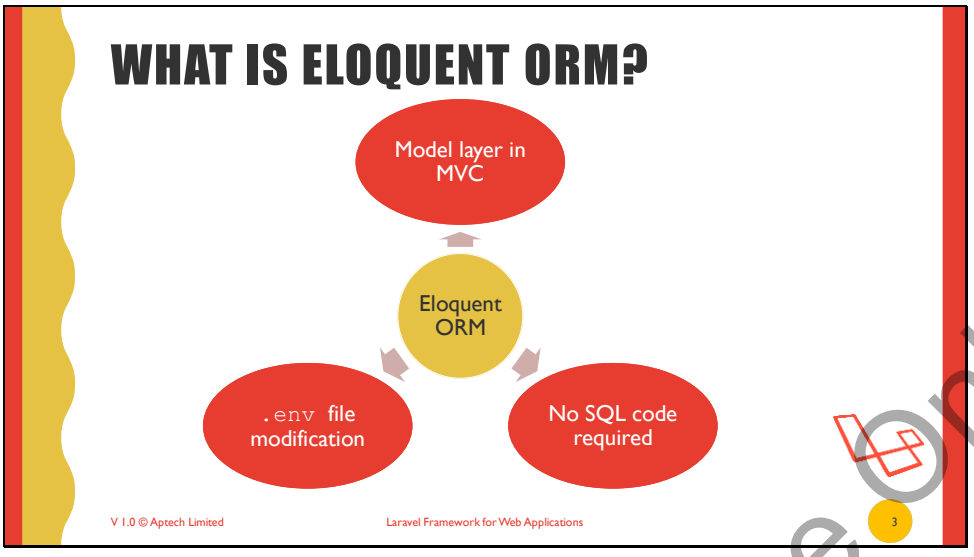
Overview of the Session
Give the students an overview of the current session in the form of session objectives. Show the students slide 2 of the presentation.

Show Slide 2



Instruction(s) to the trainer:
Using slide 2, give a brief introduction about this session to the students. Explain each objective in detail.

Show Slide 3



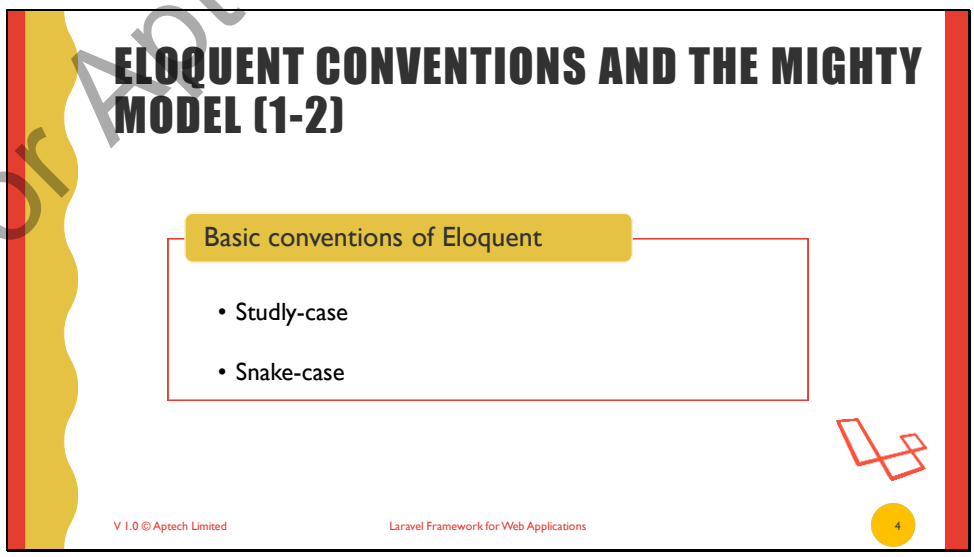
Instruction(s) to the trainer:
Using slide 3, explain Eloquent ORM to the students in detail. ORM stands for **Object Relationship Model**. Elaborate on the features of Eloquent ORM given in the slide. Explain that Eloquent ORM acts as a model layer in the MVC design pattern. In Laravel, the Eloquent ORM is useful for creating models that help a user to work with database without writing a single line of SQL code. In addition, while migrating to other databases, it only requires modification in the **.env** file.

For more information on Eloquent ORM, refer https://laravel.com/docs/5.0/eloquent#introduction.

Activity:

Ask the students to prepare a document illustrating the features of Laravel Eloquent.

Show Slide 4



Instruction(s) to the trainer:
Using slide 4, explain the basic conventions of Eloquent. Elaborate on the points given in the slide. Explain that Eloquent model name is studly-cased version of the table name that it corresponds to. For example, the Eloquent model name for the **employees** table is **Employee**. With the name Employee, Eloquent by default performs database operations on a table, which has the snake-case plural version of its name. In a snake-case, the text is formatted with small letters. The snake case for Employee is employee, and its plural is employees. Note, the Employees table is created in session 4.

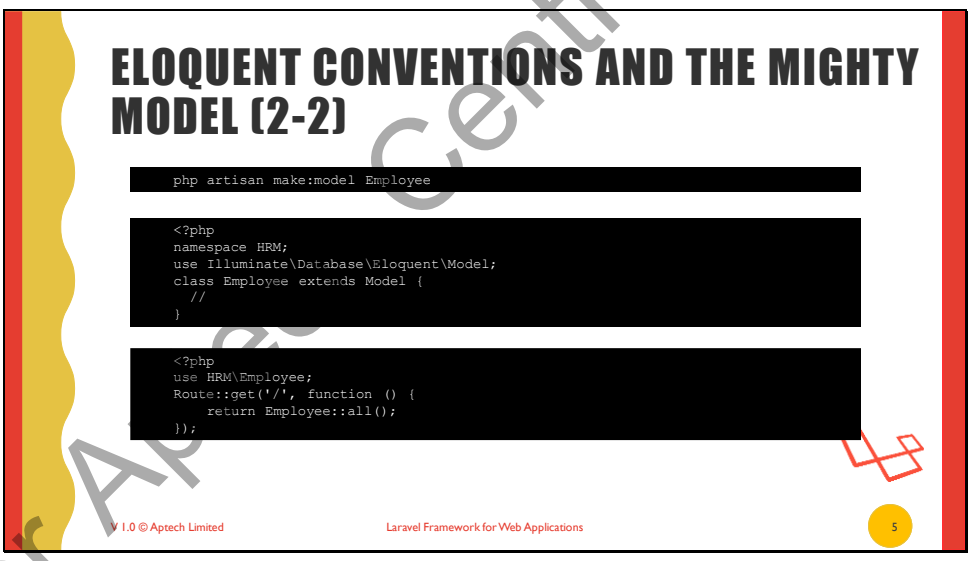
For more information, you can refer following links:

https://stillat.com/blog/2017/11/27/laravel-55-string-helper-function-studly

https://laravel.com/docs/5.8/helpers

Activity:
Ask the students to create a document on Laravel conventions. They should be able to explain the same in class.

Show Slide 5

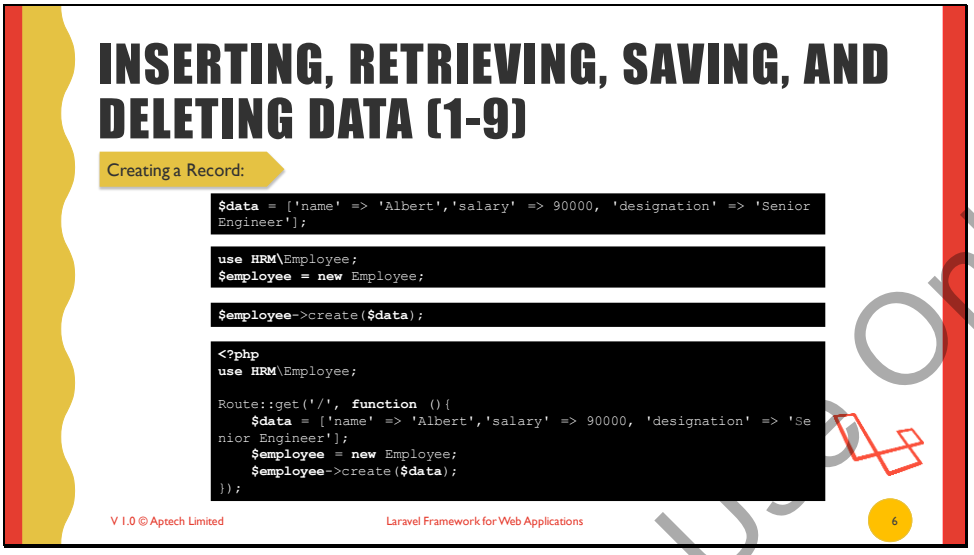


Instruction(s) to the trainer:
Using slide 5, explain creating an Eloquent model. First, execute the command as shown in the first code snippet in the slide. The app/Employee.php file corresponding to the Employee model is created in the app directory and it includes the code as shown in the second code snippet. To retrieve all records in a database (which is yet to be created), use the web.php routes as shown in the third code snippet.

Ask:
What is studly case and snake case?

For more information, you can refer:
https://laravel.com/docs/5.8/eloquent

Show Slide 6



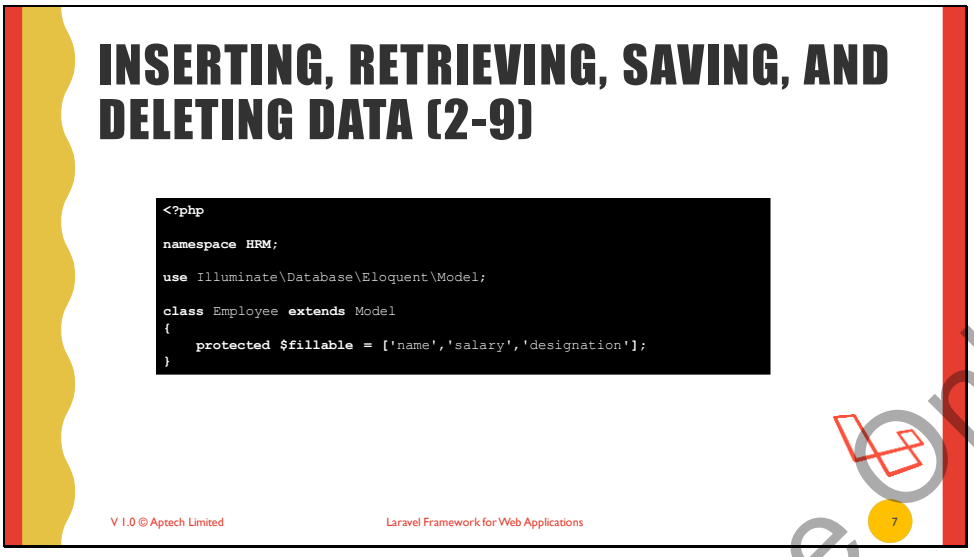
Instruction(s) to the trainer:
Using slides 6 and 7, explain the steps to create record in a model as follows:

Step 1: To create an associative array, execute the command as shown in the first code snippet.
Step 2: To reference the Employee model, use the **use** statement and then, create an instance of the Employee model as shown in the second code snippet. In this code, **HRM** is the application name, which must be modified if the application name is different.
Step 3: To insert the data array, use the create method of the Employee instance as shown in the third code snippet.
Step 4: For test purpose, delete the content in the Web routes **Web.php** file.
Step 5: Create a new route for the / path.
Step 6: Enter the code from the previous code snippet inside the closure function. The fourth code snippet shows the final route file.
Step 7: To view the result, browse to the Web application at **http://hrm.local/**. A **MassAssignmentException** occurs.

Additional information:

A **Mass Assignment** is a case where the attributes of a Model are blindly updated. For example, if the **$data** array comes when a user submits a form then, the user can update any value in the same database. This method is detrimental because a hacker can gain access and update the fields.
To prevent any such malicious attempt, it is necessary to include the **$fillable** keyword array. Doing so informs the application about the fields that can be updated by a user.

Show Slide 7

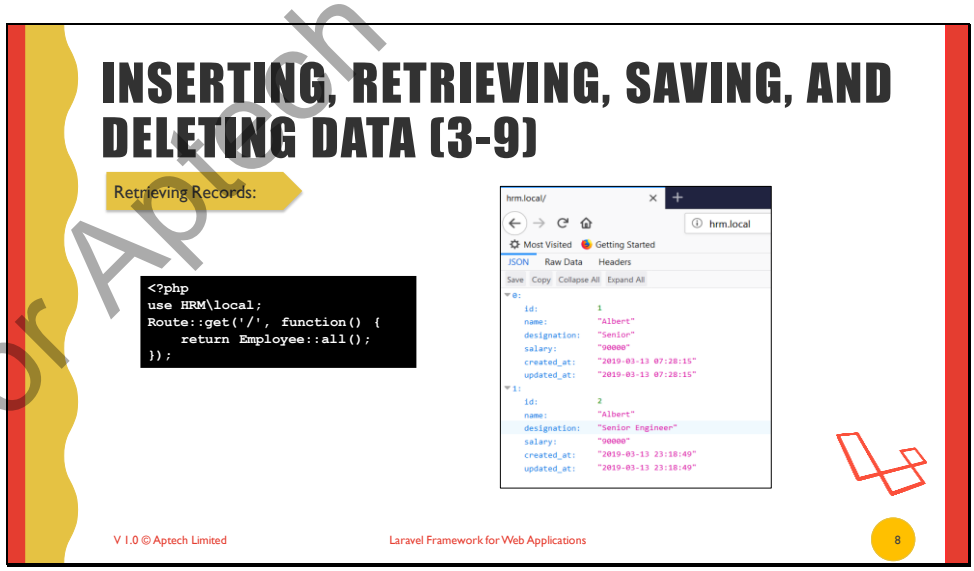


Instruction(s) to the trainer:
Using slide 7, explain steps to modify model and view the result:

Step 8: To modify the model, open the **Employee** model in **app/Employee.php** and enter the code as shown in code snippet in the slide. The **$fillable** array signifies that only the fields, name, salary, and designation can be updated manually.

Step 9: To view the result, browse to the Web application. A blank page appears because there is no return statement inside the closure function.

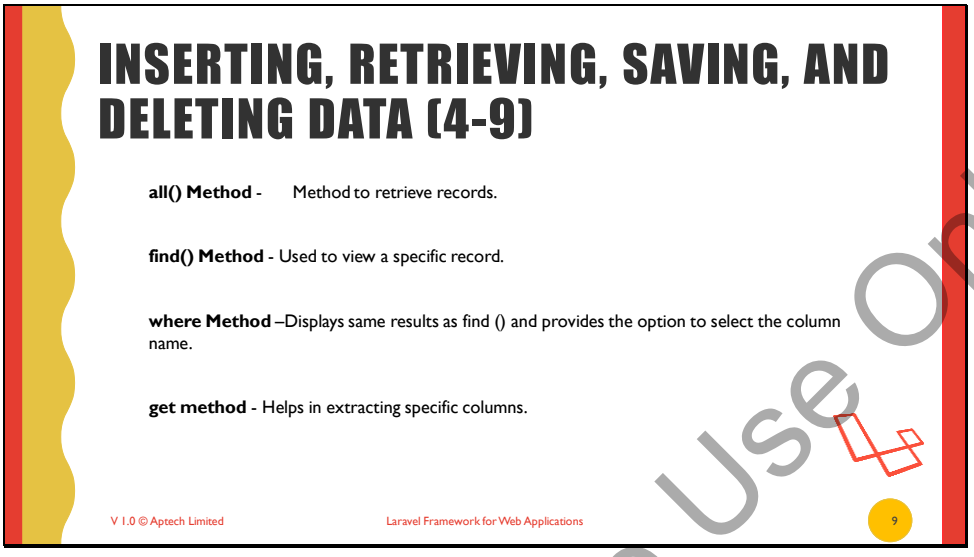
Show Slide 8



Instruction(s) to the trainer:
Using slide 8, students can be introduced to retrieving records. To know if the new record is successfully added, perform the following steps:

Step 1: To reference to the method, replace the routes file content with the code as shown in this slide and save the file.
Step 2: To view the page, browse to the application as shown in the image.

Show Slide 9



Instruction(s) to the trainer:
Using slide 9, give a brief idea of all different methods mentioned in the slide.

all() Method - When the Employee::all() method is used, it returns a collection object and Laravel returns as a json object. The output in each instance will be different.

find() Method - To view a specific record, use the find() method; the records are sorted according to the filtered criteria. Replace the route file with the code as shown in Code Snippet below and browse to the Web application.

```
<?php
use HRM\local;
Route::get('/', function() {
return Employee::find(2);
    });
```

The record with primary key value two is displayed. Note, the primary key is also the column id because the bigIncrements method was used to define the id column during database migration.

Where Method – Code snippet for **where** method is as shown:

```
Employee::where('id', '=', 1)->first();
```

get Method – Code snippet for **get** method is as shown:

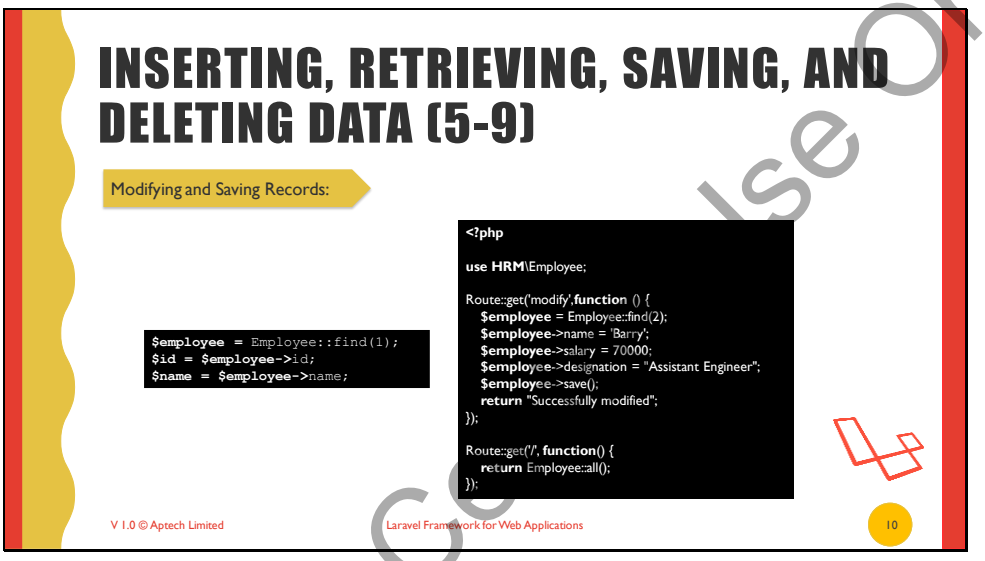
```
Employee::get('id');
```

Additional information:
Methods of a Model class return the Collection object type. This collection object has multiple methods which return another collection object. This feature of the Eloquent Model is called chaining. Clauses are chained so that filters are created to filter records rather than writing WHERE clauses.

Activity:

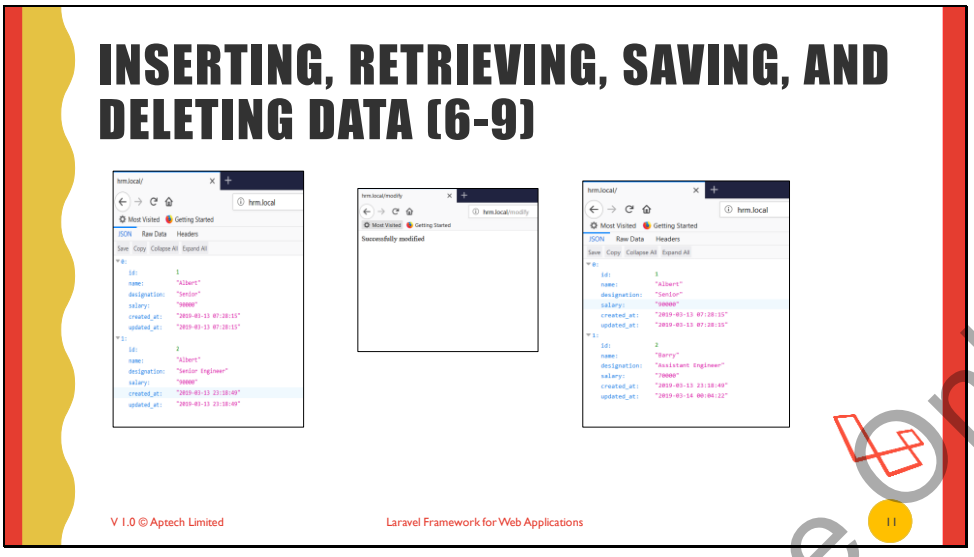
Ask the students to prepare a document explaining all(), find(), where, and get methods.

Show Slide 10



Instruction(s) to the trainer:
Using slide 10, explain process of modifying and saving records. To access the values after retrieving a record, enter the code as shown on left. The values can also be changed using basic assignments, and the changes can be reflected in the database using the save command. Rewrite the routes file as shown on right. `Creates two routes, for '/', and 'modify'. The '/' route displays all the entries within the table and as soon as you visit the second route, the record with the id of two is updated. Going back to '/' will help see the changes. The output is displayed in slide 11.

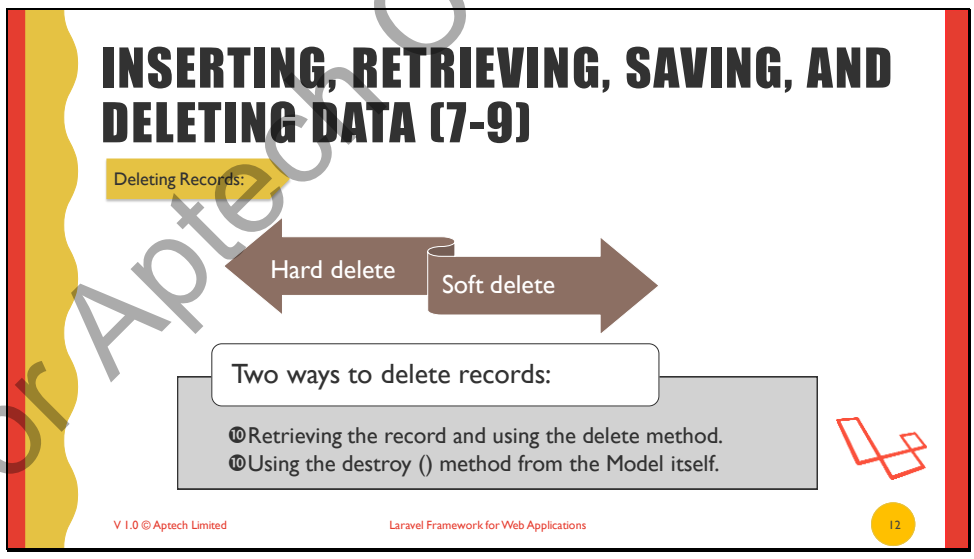
Show Slide 11



Instruction(s) to the trainer:
Using slide 11, explain the images displayed. The first image shows all entries for **'/' Route,** second image shows output for updating records, and the finally the third image the output for updated records.

Ask:
Which method is used to select specific columns?

Show Slide 12



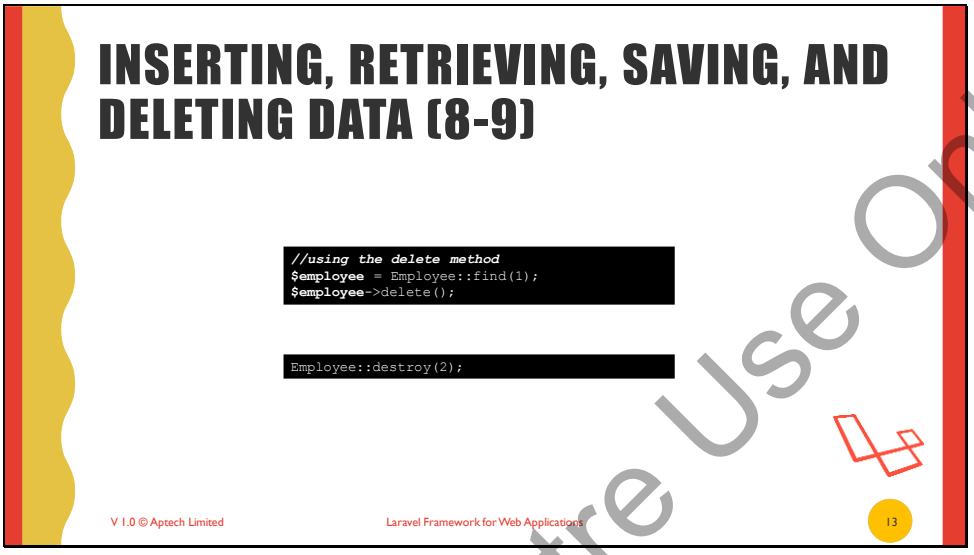
Instruction(s) to the trainer:

Using slide 12, explain the ways of deleting records.
Explain to the students the two ways by which Eloquent deletes records:
Hard delete - Hard delete leads to permanent removal of data and cannot be retrieved.

Soft delete - Soft delete does not remove the data from the database; instead a **deleted_at** timestamp is added.

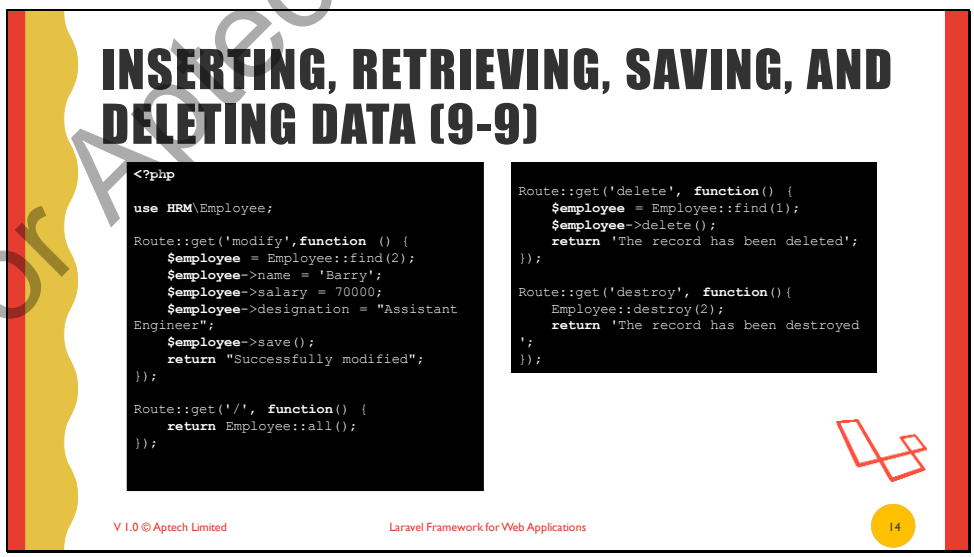
The two ways to delete records is shown in this slide. The two methods are described in detail in slides 13 and 14.

Show Slide 13



Instruction(s) to the trainer:
Using slide 13, explain the code snippets shown in the slide. The first code snippet shows **delete** method. To delete the record with id=1, use the **delete** method as shown. The second code snippet shows using the **destroy ()** method. To destroy the record with id=2, use the **destroy** method as shown in the code snippet.

Show Slide 14



Instruction(s) to the trainer:
Using slide 14, explain that the slide displays **Web.php** route file after deleting the record.

Following are the mentioned steps:

- To view all records, browse to **hrm.local**.
- To delete the record with id=1, browse to **hrm.local/delete**.
- To view the deleted data, browse to **hrm.local**.
- To test the destroy method, browse to **hrm.local/destroy.**
- To view the deletion, browse to **hrm.local**.

Explain **soft delete** to the students. To soft delete a record, the model and migration is required to be modified. When models are soft deleted, they aren't removed from the database. Instead, a **deleted_at** flag is set on the model and inserted into the database. Now, to check whether a model is deleted or not, we can check **deleted_at** value. If it is non-null, it can be concluded that the model has been soft deleted.

Show Slide 15

Instruction(s) to the trainer:
Using slide 15 explain the concept of query scopes. Explain that sometimes chaining becomes complex and requires considerable time and effort. For example, to find a specific record, multiple WHERE and get methods need to be chained. If the query needs to be used again, then the code needs to be typed again. The re-type action may lead to typos and error in code. Query scopes can be used in such a situation.

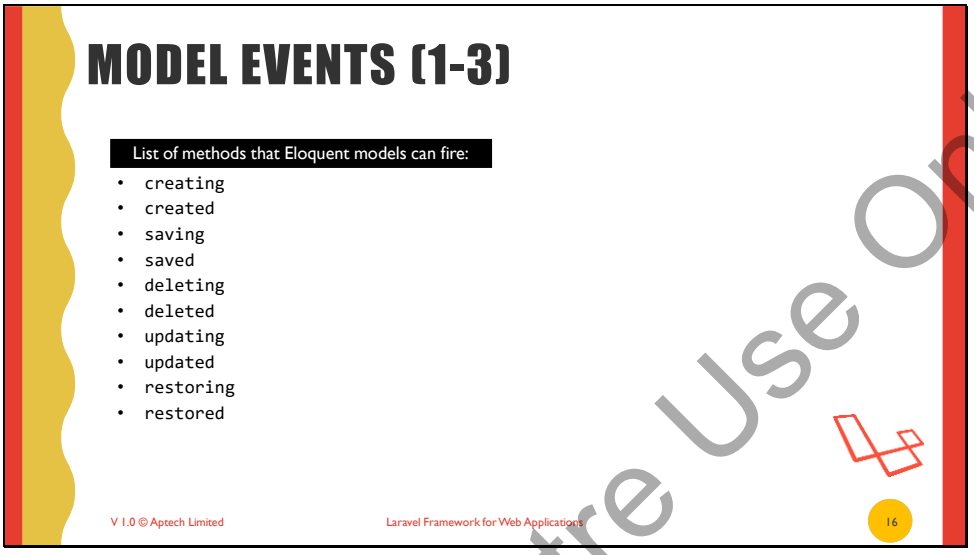
Give an example to the students using query scopes. For example, to find an employee with designation 'Senior Engineer' and salary above 80,000, consider the chained query as shown in the first code snippet in the slide. To use the same query multiple times, define the function in the model as shown in second code snippet. Eloquent automatically provides the chained query as an argument to the $query variable. The highPaidHighRank query scope then makes further chains from that point; thereby allowing the query scope to be used as a chained method in itself.

The final Employee model appears as shown in the final code snippet in the slide.

Ask:
Explain the need for query scopes?

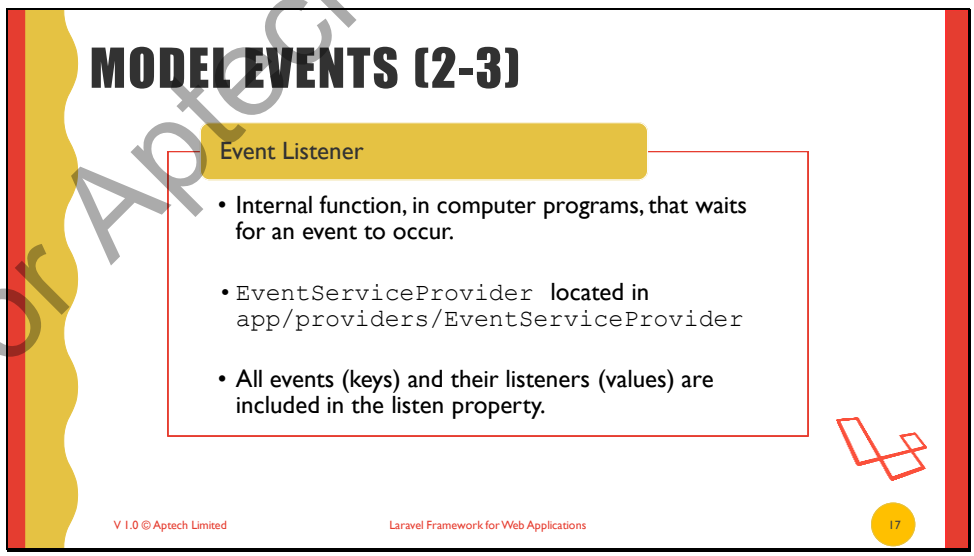
For more information on query scopes, refer https://laraveldaily.com/query-scopes-convenient-way-to-isolate-often-used-conditions/

Show Slide 16



Instruction(s) to the trainer:
Using slide 16, explain the list of methods that Eloquent models can fire. Eloquent fires numerous events at different points, such as when a model is being saved or deleted.

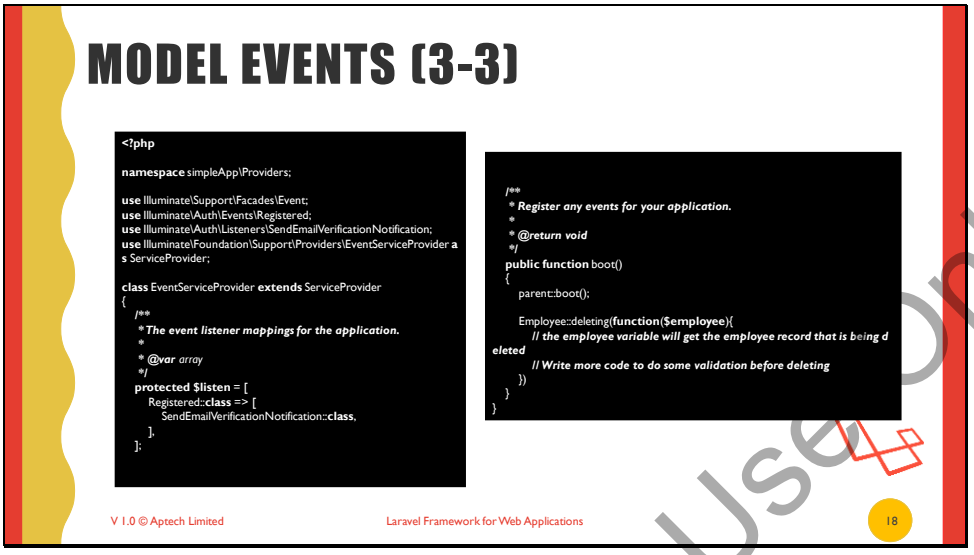
Show Slide 17



Instruction(s) to the trainer:
Using slide 17, explain the Event Listener function. Give examples of event listeners such as user clicking the mouse, network activity, or pressing a key on the keyboard.

For more information on event listeners, visit https://vegibit.com/8-steps-to-success-with-laravel-events/

Show Slide 18



Instruction(s) to the trainer:
Using slide 18, explain the code for deleting event. For this, edit the EventServiceProvider class as shown in the code snippet.

Show Slide 19

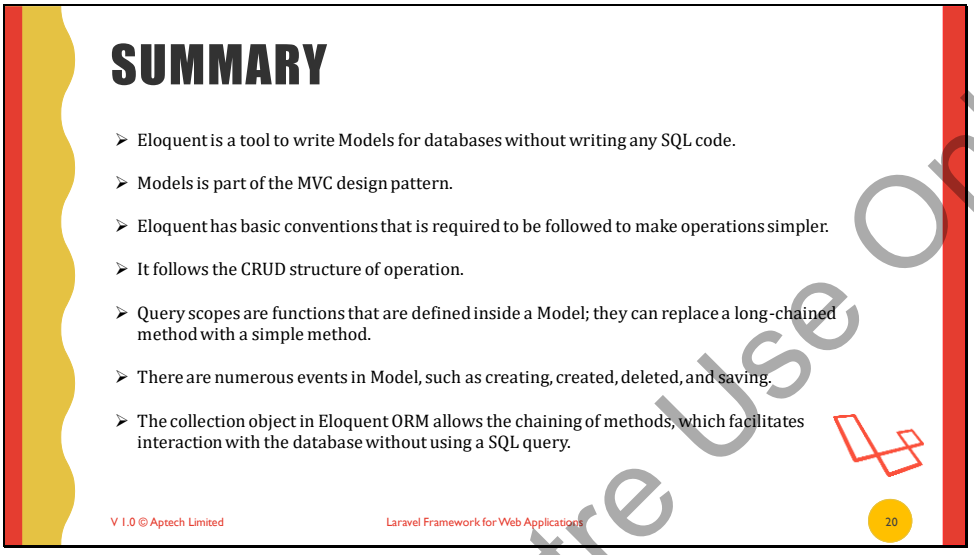
Instruction(s) to the trainer:
Using slide 19, explain in brief, relationships and collections in Laravel.
Relationships - Eloquent helps define the relationship between tables using the common fields. This is possible when multiple tables share the same fields.
Collections - The collection object in Eloquent ORM allows the chaining of methods, which facilitates interaction with the database without using a SQL query. Methods, such as **all**, **find**, **get**, and **where** are part of collections.

For more information on relationships and collections, refer:

https://laravel.com/docs/5.8/eloquent-relationships

https://laravel.com/docs/5.8/eloquent-collections

Show Slide 20



Instruction(s) to the trainer:

Using slide 20, summarize the key concepts from this session.

5.2 Post Class Activities for Faculty

You should familiarize yourself with the topics of the next session.

Tips: You can also check the Articles/Blogs/Expert Videos uploaded on the OnlineVarsity site to gain additional information related to the topics covered in the next session. You can also connect to online tutors on the OnlineVarsity site to ask queries related to the sessions.

Session 6: Implementing CRUD Operations

6.1 Pre-Class Activities

Before you commence the session, you should familiarize yourself with the topics of this session in-depth. You should revisit topics of the previous session for a brief review. Here, you can ask students to recall the key topics from previous session. Prepare a question or two that will be a key point to relate the current session objectives.

6.1.1 Teaching Skills

To teach this session, you should be well-versed with the CRUD operations. You should be able to explain different aspects of MVC design patterns. Also, you should be able to tell them about implementing Session, Request, and Response, and explain them the concept of paginations. In addition, you should be able to describe the importance of authentication.

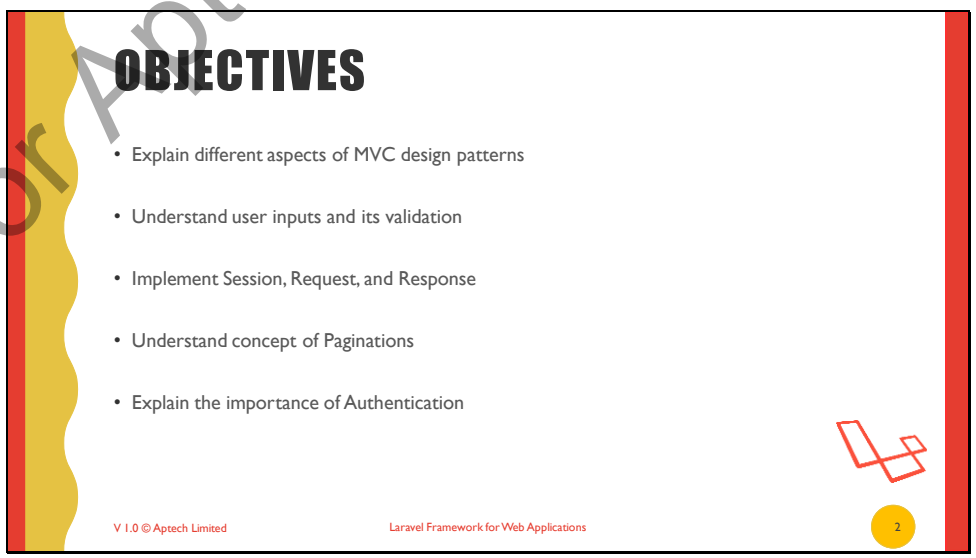
For teaching in the class, you are expected to use slides and LCD projectors.

Tips:
It is recommended that you test the understanding of the students by asking questions in between the class.

In-Class Activities
Follow the order given here during In-Class activities:

Overview of the Session
Give the students an overview of the current session in the form of session objectives. Show the students slide 2 of the presentation.

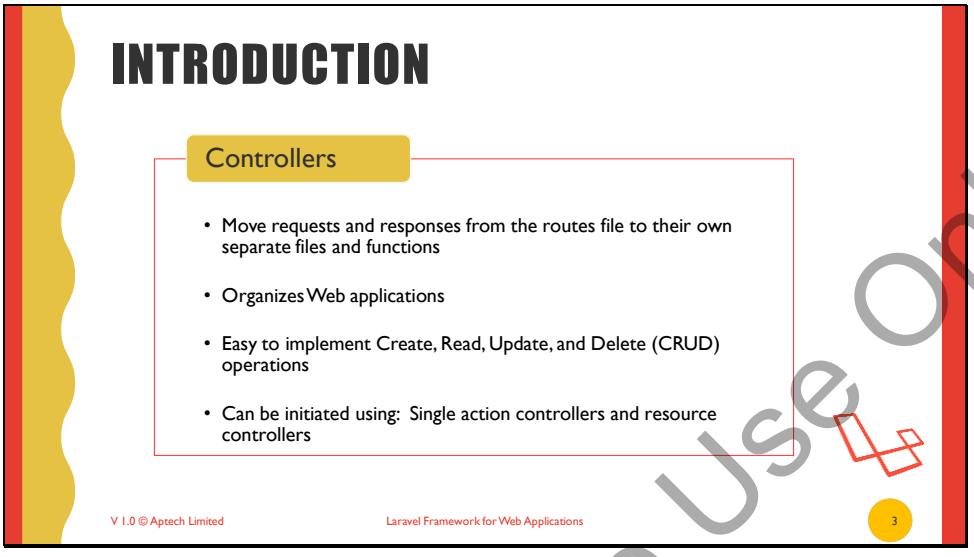
Show Slide 2



Instruction(s) to the trainer:

Using slide 2, give a brief introduction about this session.

Show Slide 3



Instruction(s) to the trainer:

Using slide 3, introduce the concept of controllers in the MVC design pattern to the students. Explain that Controllers are classes that are predefined in Laravel. In cases where more features or functions are needed for a Web application, many closure functions are included in route files to handle HTTP requests. Thus, the application logic required to execute these increased number of functions leads to additional lines of code in the route file as well. Controllers help in organizing these codes by grouping requests that handle the codes in a single route and class. Also, multiple routes with multiple functions can be mapped to a single Controller class. They are stored in **app/Http/Controllers** directory stores Controllers.

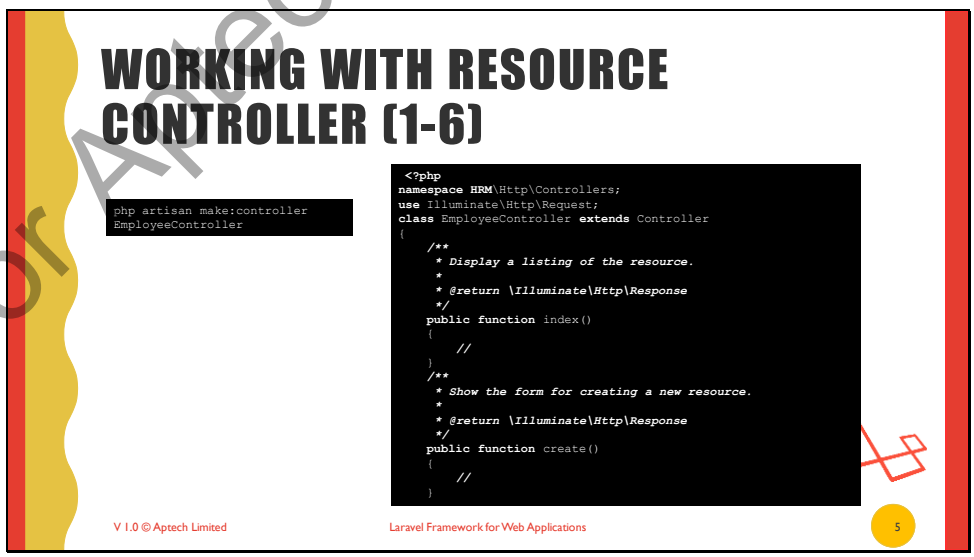
For more information on controllers, refer:
https://laravel.com/docs/5.8/controllers.

Show Slide 4

Instruction(s) to the trainer:
Using slide 4, explain creation of **Employee** Controller. Explain the code snippets in detail to the students. The first code snippet creates a basic controller called **EmployeeController.php** in **hrm/app/Http/Controllers**. The controller will include the code as shown in the second code snippet. The third code snippet shows the code in vagrant ssh console to create a resource controller with the same name. Delete the basic controller, if created.

In-class Question:
Q. Name the directory that stores controllers?
Ans: The **app/Http/Controllers** directory stores Controllers.

Show Slide 5

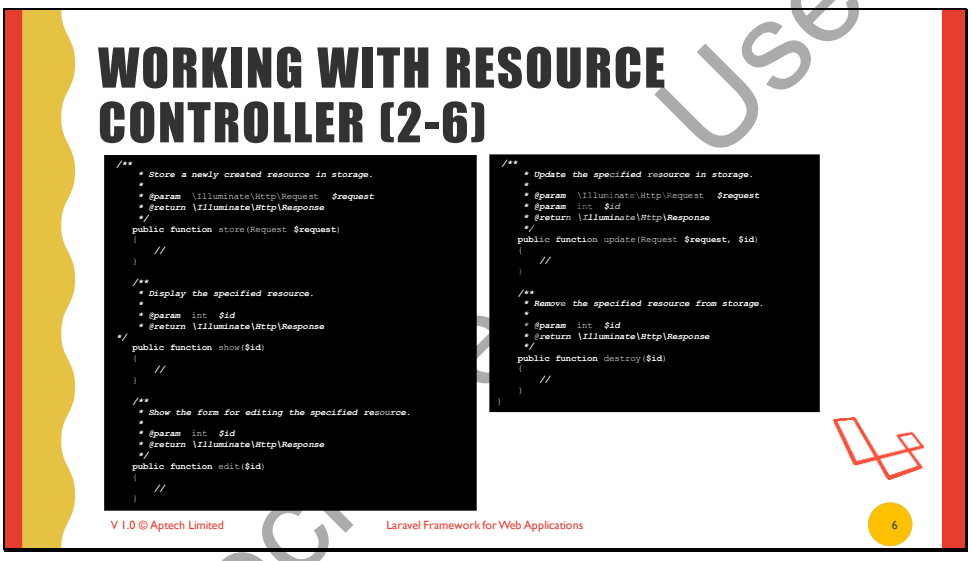


Instruction(s) to the trainer:
Using slides 5 and 6, explain working of resource controller.

CRUD is used almost in each Web application but Laravel makes this process easy by using the resource controllers. The resource controllers take advantage of some routing techniques and make the task much easier. Use the artisan command to preconfigure the resource controller. It creates a Controller framework that is commonly used for CRUD operations. To create a Resource Controller, append the --resource flag after the original command as shown in the first code snippet of slide 5. The second code snippet in this slide and in slide 6 (continued from slide 5) shows the source code of the Controller that displays the **CRUD** functions.

Activity:
Create two groups among participants and inform them they have 10 minutes to gather information about artisan commands. Now call participants randomly from both the groups to discuss about specific artisan command.

Show Slide 6



Instruction(s) to the trainer:
Using slide 6, show the source code of the Controller that displays the CRUD functions (continued from slide 5). It includes the following functions:

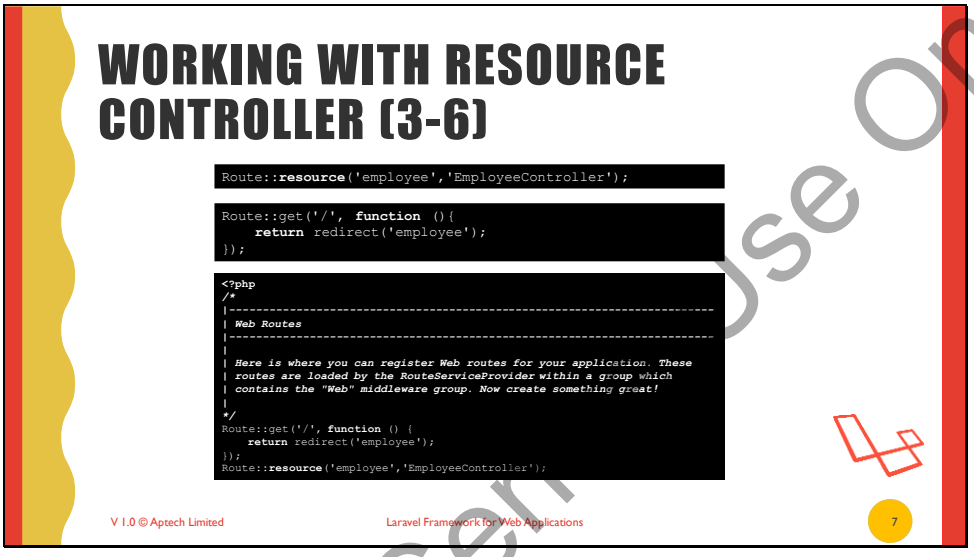
create()
destroy()
edit()
index()
show()
store()
update()

The action performed by each function easily relates to the name assigned to them. For example, the destroy () functions deletes records, the edit () function provides a form to edit records, and so on. The code snippet is self explanatory as it includes the comments that explain the code. The functions can be configured to automate declaration tasks, which help in reducing the development time.

For example, the destroy function is shown in the code that also includes the 'Remove the specified resource from storage' comment, which describes the type of parameter that can be accepted as well as the type of object that the function must return.

Activity:
Inform students they have to prepare a document on the various CRUD operation related functions and explain each of them in detail.

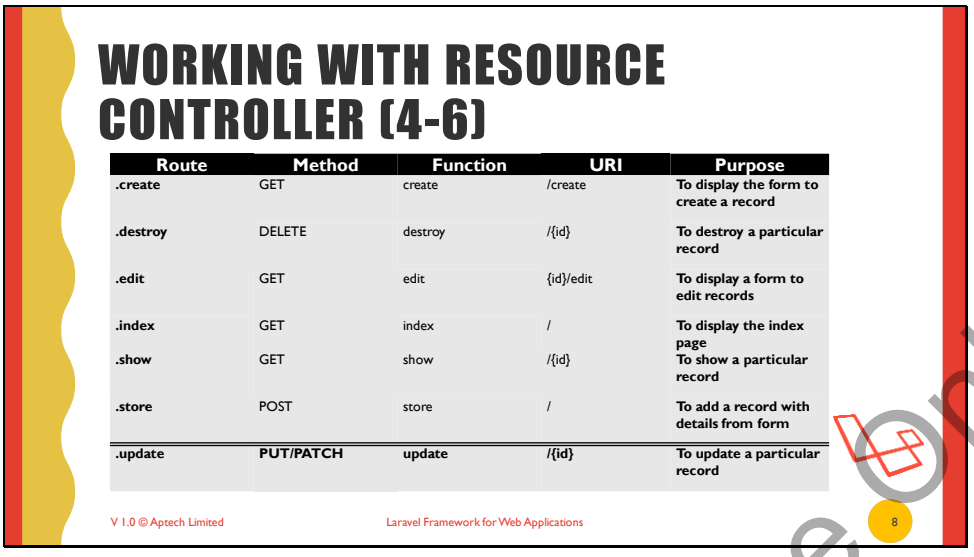
Show Slide 7



Instruction(s) to the trainer:
Using slide 7, explain another important feature in Laravel - **resource routing**. This feature automatically assigns routes to all functions that have been declared in the Resource Controller with a single line of code. The first codes snippet shows how to register the resourceful route for the **http://hrm.local/employee path**. The second code snippet shows the redirect function to redirect to the **employee** directory from the root path (/). The third code snippet shows the final route **routes/Web.php** file where two routers are added and previously defined routers are removed.

Additional information:
Resourceful routes work well in subdirectories. The show and destroy methods cannot be used in the Web root directory path (/) else it may create problems. For this reason, the employee subdirectory will be used to implement CRUD operations on the Employee database.

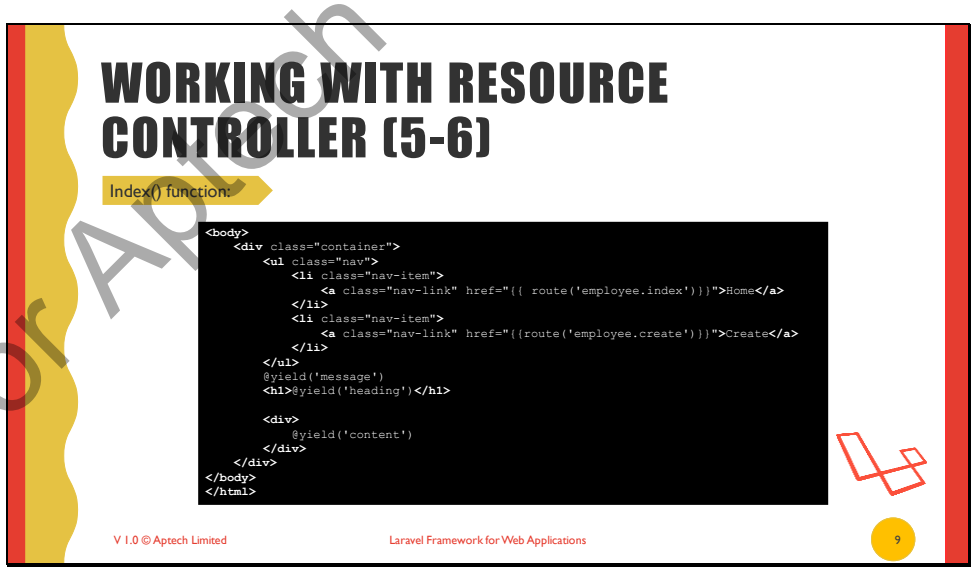
Show Slide 8



Instruction(s) to the trainer:
Using slide 8, show the list of routes that correspond to the defined path (**employees**) and are associated with the respective functions in the **EmployeeController** Controller. The list also appears using the artisan command - **php artisan route:list.**

For more information on route:list command, refer:
https://stillat.com/blog/2016/12/07/laravel-artisan-route-command-the-routelist-command

Show Slide 9



Instruction(s) to the trainer:
Using slides 9 to 13, display the implementation of a few CRUD operations. To use the **Employee** model, add the namespace of the model by typing use **HRM\Employee.**
The following code snippet shows the first few lines of the **EmployeeController** code.

Code Snippet:

```
<?php
namespace HRM\Http\Controllers;
use Illuminate\Http\Request;
use HRM\Employee;
```

Slide 9 shows the code snippet that demonstrates steps for using the index() function.
The steps are as follows:

Step 1: Modify the **master** blade template.
Step 2: Include a few **bootstrap 4** classes.

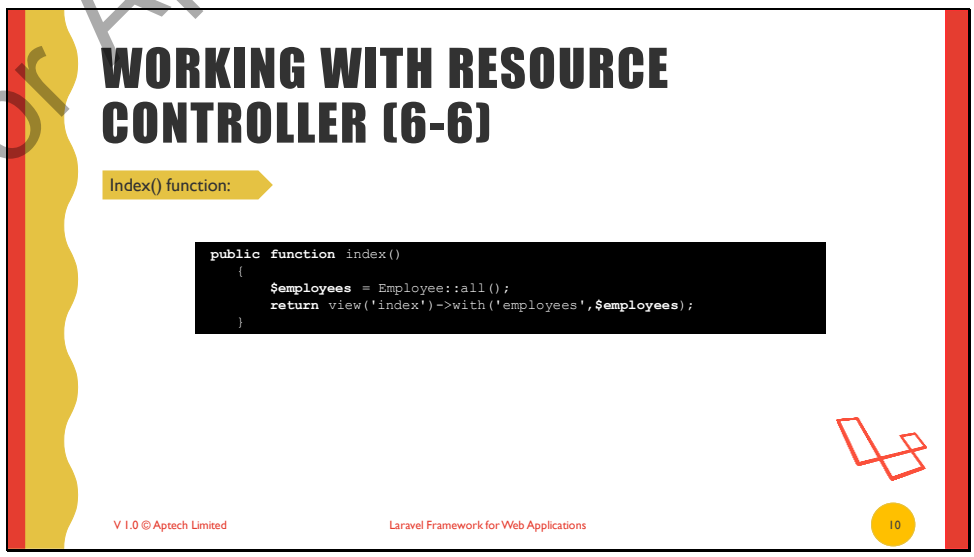
Show the demonstration on this topic from student guide.

Explain them that the code uses **Bootstrap 4** for styling purposes. For example, a navigation bar is added to help a user return to the Home page from multiple routes and views the Web application may use. The template also includes four extensible sections that are defined by the following constructs:

@yield('content'): This is a placeholder for the actual code to perform CRUD operations.
@yield('heading'): This is a placeholder for different headings of different views.
@yield('title'): This is a placeholder for different titles of different child views.
@yield('message'): This is a placeholder for displaying messages to show the result of a CRUD operation.

The route function is also used. It is a new helper function that returns a URL for the given route. The index() function corresponds with the first route **http://hrm.local/employee**. The function is used to list resources; the kind of information that is included in the Contents page.

Show Slide 10



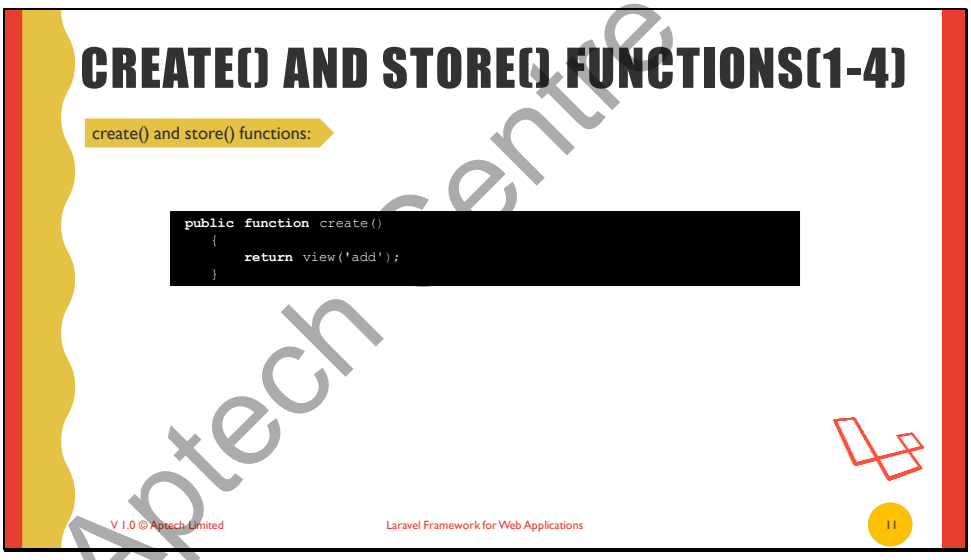
Instruction(s) to the trainer:
Using slide 10, show the code snippet with the index() function. The code snippet in the slide makes use of the index() function to display minor details of all employees present in the database. The $employee variable will be passed to all the records that are in the employees table using the Employee::all() method. The variable will then be passed to the view index created with variable $employee.

Thus, as can be seen, index() function will create an index on a particular column which it refers to. Thus, it becomes easier to make queries for that column, the **employees** column in this case.

Ask:
Q: What is the purpose of index() function in Laravel?

For more information on resource controllers, refer:
https://scotch.io/tutorials/simple-laravel-crud-with-resource-controllers
https://www.sitepoint.com/crud-create-read-update-delete-laravel-app/

Show Slide 11



Instruction(s) to the trainer:
Using slides 11 to 13, explain the **create()** and **store()** functions. The **create()** and **store()** functions are bundled together to form the create function in the CRUD operations. The function of create() and store() functions are as follows:

create() - displays a form for the user to enter values for a new employee record.
store() - includes the code to create a new record in the database.

To perform actions, the **create()** function can be populated with code as shown in the code snippet in this slide. The create() function returns the View with the name add. This can be changed as per requirement. Laravel puts no conventions on the name of views used. This does not work for the home view as the view will be overwritten if authentication is implemented.

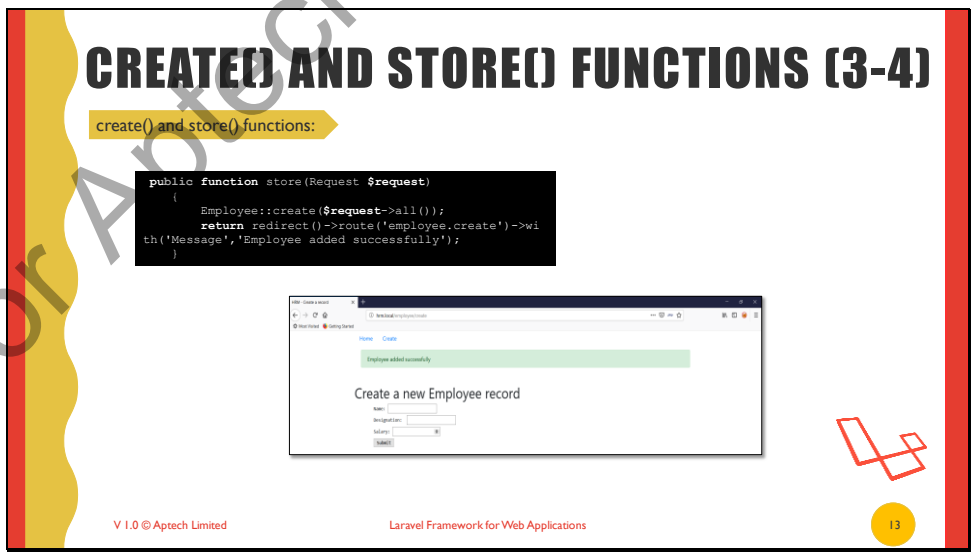
For more information on CRUD operations, refer:
https://www.5balloons.info/tutorial-simple-crud-operations-in-laravel-5-5/

Show Slide 12

Instruction(s) to the trainer:
Using slide 12, explain the code snippet to modify the **add** View. The View contains a form, which will send a post request to the **employee.store** route. The 'message' section prints the status message, which will be sent from the **store** function.

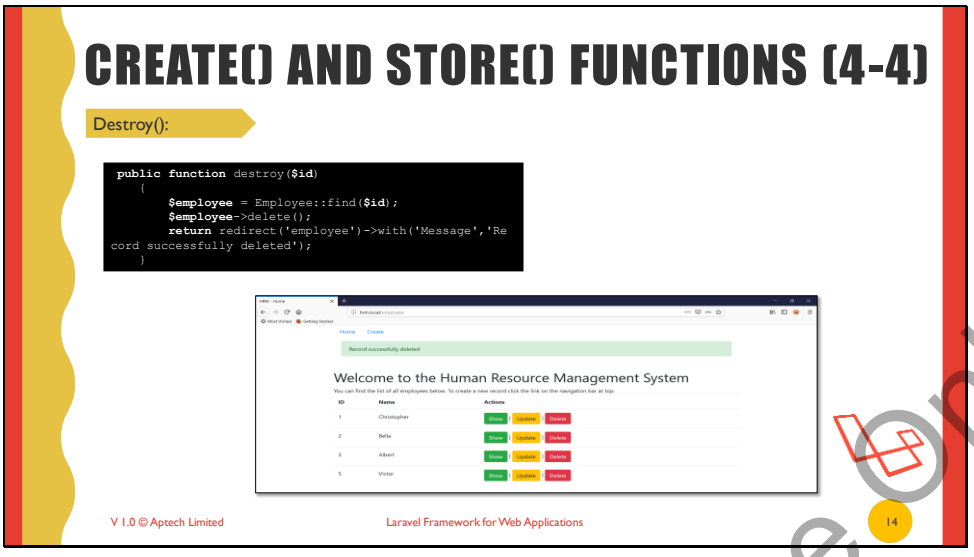
Ask:
Explain the features of create() and store() functions in Laravel.

Show Slide 13



Instruction(s) to the trainer:
Using slide 13, explain the code snippet to be added to store function of the controller. The image in the slide shows the form where the entries can be added. The form should get displayed by clicking the **create** link on the NAV bar.

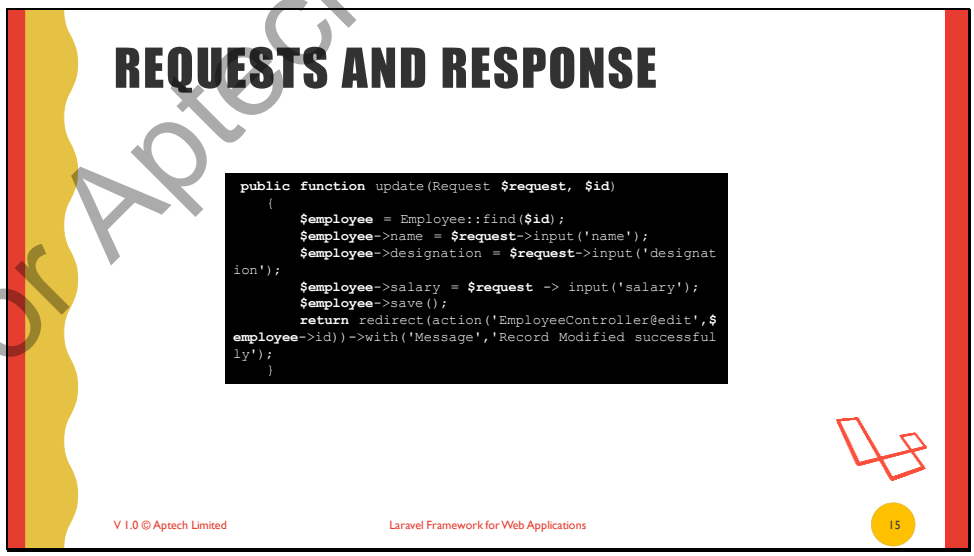
Show Slide 14



Instruction(s) to the trainer:
Using slide 14, explain the code snippet to delete a record. The destroy() function is used for this purpose. The code redirects the user back to the home page, so no additional views need to be created. The image in the slide shows the delete page.

Activity:
Similar to the functions discussed in the slides, ask the students to research more on other basic functions such as edit(), update(), show() and keep themselves updated on these functions. This will enhance their knowledge of all the basic CRUD functions required to create a Web application.

Show Slide 15

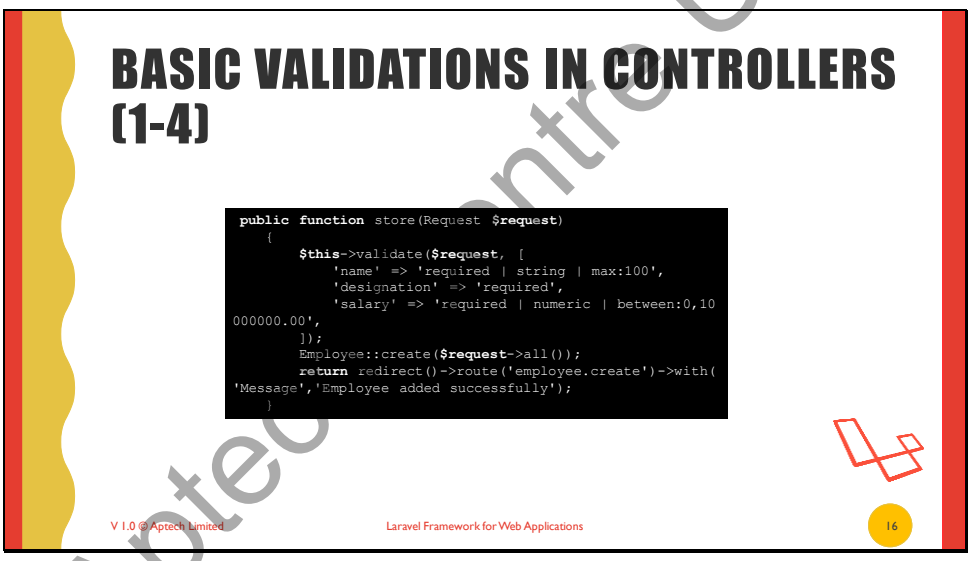


Instruction(s) to the trainer:
Using slide 15, explain requests and response in Laravel. Controller class has parameters with type **Request**. In the code snippet, the **$request** parameter is an object of the **Request** class.

The **$request** object has all data relating to request, such as HTTP method and input parameters. In the Code Snippet, the **$request** object is used to retrieve values that have been passed through forms. The **$request->input()** method is used to retrieve values that are passed through a POST or GET request, using their parameter name. Similarly, the **$request->path()** retrieves the URL that is making the request. The $request->method()retrieves the method used. The **Response** object returns a response object to the requesting client. All the return statements used in the routes and Controllers return a **Response** object. Laravel automatically converts strings, views, redirects, and many more to a valid request object. This saves a lot of time for the developers as they don't have to deal with creating different responses depending on different computations.

In-Class Questions:
Q: What is the function of **$request->input()** method?
Ans: The **$request->input()** method is used to retrieve values that are passed through a POST or GET request, using their parameter name.

Show Slide 16



Instruction(s) to the trainer:
Using slide 16, show the code snippet to modify the **store()** method to add some validations. Laravel offers many ways to validate user input sent to servers. The incoming data is validated in the **store()** and **update()** functions of the **EmployeeController** class. The request object **Illuminate\HTTP\Request** provides a validate method to validate the input. If the input is as per the validation rules then the code executes normally, else, an exception is thrown. In addition, the corresponding error message is stored in an array named **errors**. The errors can then be handled accordingly.

In the code shown, the **validate** method takes in the incoming request through the **$request** variable as its first argument, and an array containing validators as its second argument.

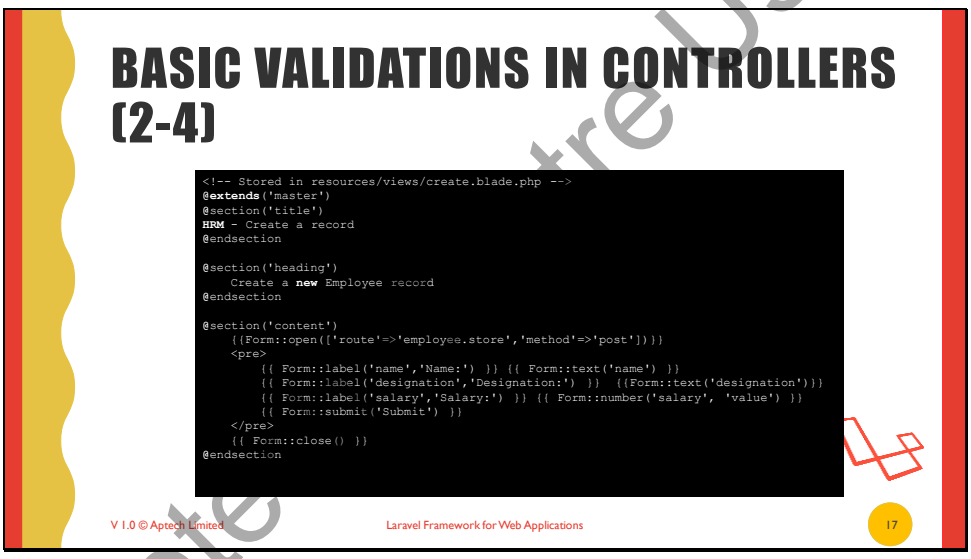
The associative array contains key value pairs. The keys should contain the same name as the column that they are validating. The value should contain validating keywords separated by the pipe | symbol.

Following describes the validators.

between:x,y: Validates if the value is between x, and y.
max:x: Validates if the string is less than x characters in length.
numeric: Validates if the value entered is a numeric value.
required: Validates if any value was passed for that column.
string: Validates if the input passed is a string value.

For more information on request validation, refer:
https://medium.com/@kamerk22/the-smart-way-to-handle-request-validation-in-laravel-5e8886279271

Show Slides 17 to 19



Instruction(s) to the trainer:

Using slides 17, 18, and 19, explain that the code snippets serve as an example of modifications done to the create view to display the errors.
If a new record is created with invalid values, the Laravel Web application redirects to the requesting page without modifying any values. Laravel compiles all invalidations as errors and stores them in the **$errors** array along with a descriptive message.

BASIC VALIDATIONS IN CONTROLLERS (3-4)

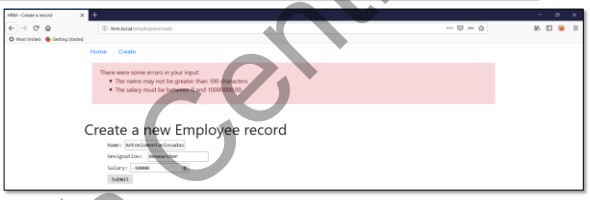
```
@section('message')
@if ($message = Session::get('Message'))
    <div class="alert alert-success">{{ $message}}</div>
@endif

@if ($errors->any())
    <div class="alert alert-danger">There were some errors in your input:
        <ul>
            @foreach ($errors->all() as $error)
                <li>{{$error}}</li>
            @endforeach
        </ul>
    </div>
@endif
@endsection
```

V 1.0 © Aptech Limited Laravel Framework for Web Applications 18

BASIC VALIDATIONS IN CONTROLLERS (4-4)

```
@if ($errors->any())
    <div class="alert alert-danger">There were some errors in your input:
        <ul>
            @foreach ($errors->all() as $error)
                <li>{{$error}}</li>
            @endforeach
        </ul>
    </div>
@endif
```



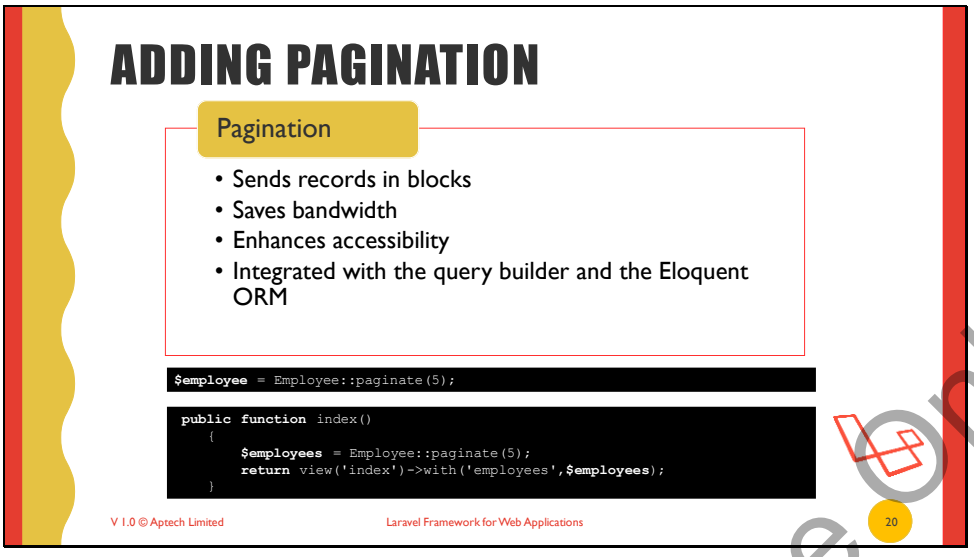
V 1.0 © Aptech Limited Laravel Framework for Web Applications 19

Instruction(s) to the trainer:

Using slide 19, explain the code added in the **'message'** section. The image in the slide displays the validation errors.

In-Class Questions:
Q: Name the validator to check if the value is between x and y.
Ans: between:x,y.

Show Slide 20

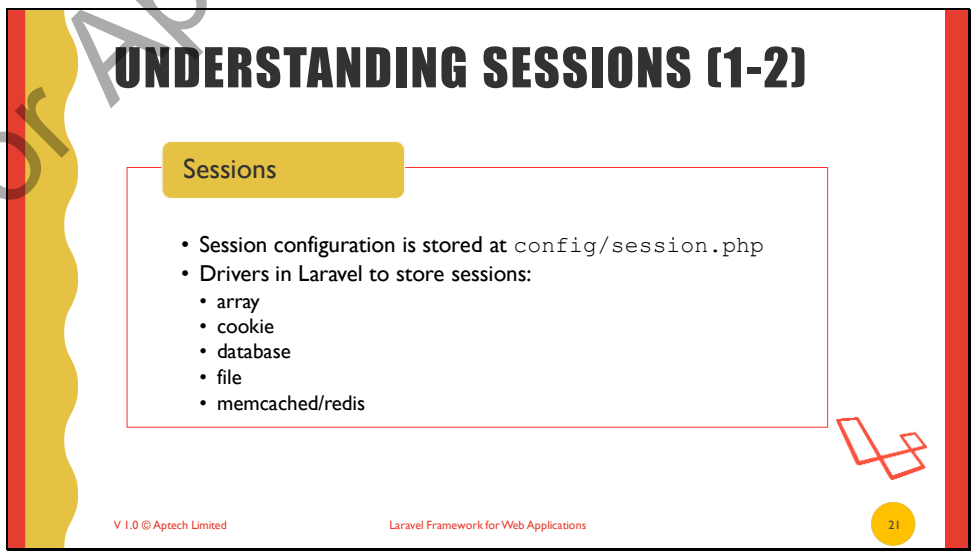


Instruction(s) to the trainer:
Using slide 20, explain the concept of pagination in Laravel. Explain that pagination is a method in which records are sent in blocks. Consider a database includes 10000 records. It is difficult to show them on a single page. In addition, sending large response increases the server load and bandwidth. These blocks of records can be organized in pages to save bandwidth and enhance accessibility.

The paginate method of a Model is the simplest way to paginate. The first code snippet in slide 20 shows how to paginate a Model. The second code snippet shows an example of modifying the index() method of the Controller. As can be seen in the code, five records are sent in an instance.

For more information on pagination, refer:
https://laraveldaily.com/back-basics-laravel-pagination/

Show Slide 21



Instruction(s) to the trainer:
Using slide 21, explain the concept of sessions in Laravel. The sessions provide a way to store information about the user across multiple requests.

As shown in the slide, the session configuration is stored at **config/session.php**. Laravel has multiple drivers to define how the session will be stored. The description of each driver is as follows:

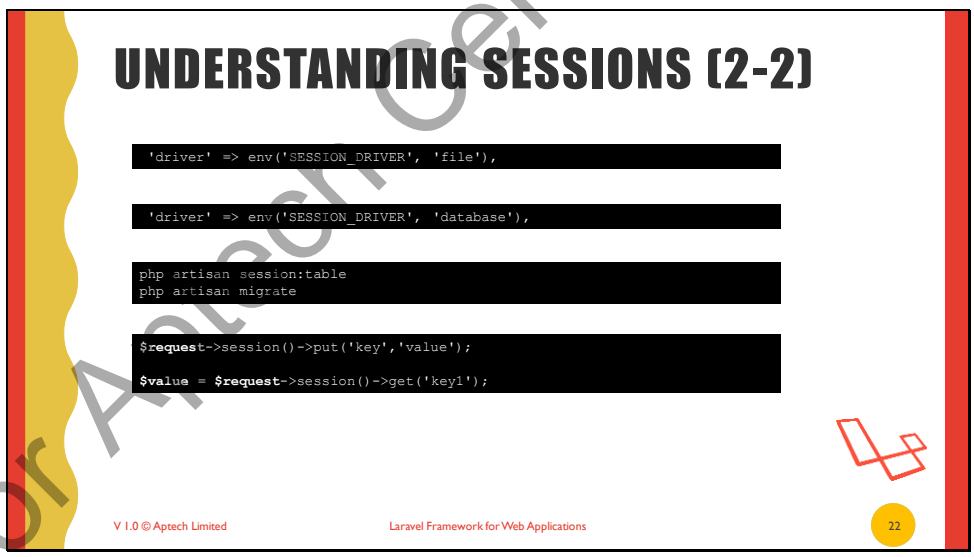
array: sessions are not persistent; they are stored in temporary PHP arrays.
cookie: sessions are stored in encrypted cookies.
database: sessions are stored in the configured database.
file: sessions are stored in storage/framework/sessions.
memcached/redis: sessions are stored in cloud-based services.

By default, Laravel uses the file driver.

Activity:
Prepare a document on sessions in Web applications.

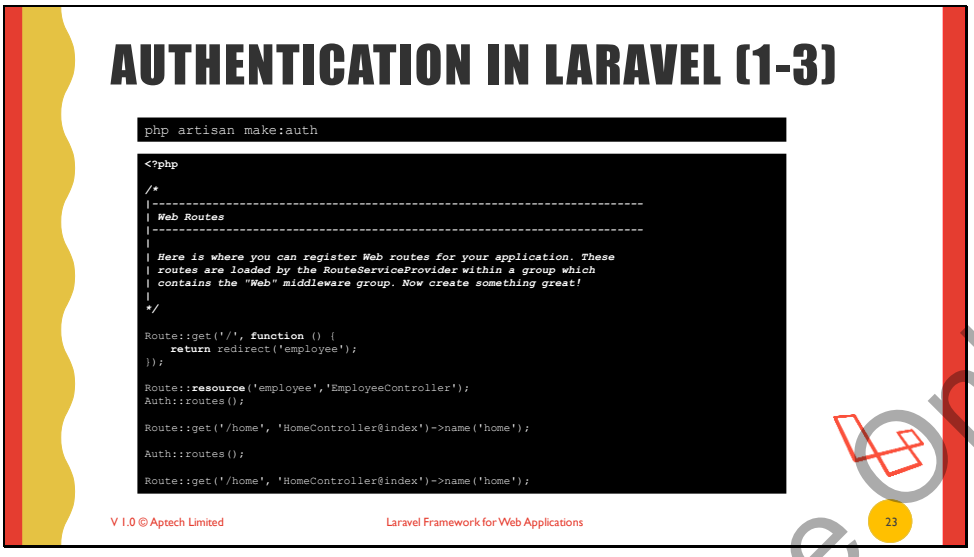
For more on sessions, refer:
https://codeboxr.com/working-with-laravel-session/

Show Slide 22



Instruction(s) to the trainer:
Using slide 22, explain the code snippets to configure sessions to be stored in a database instead of a file. To configure sessions, modify the **config/sessions.php** file. The first code snippet shows how to locate the file. The second code snippet is an example of modifying the file. To use the session, a table is required in the database and the third code snippet shows how to create a new session table in the SQLite database. Session variables can be modified using the **Session** class or the **Request** class. Session variables are defined in a **key:value** pair and can be defined as shown in fourth code snippet.

Show Slide 23

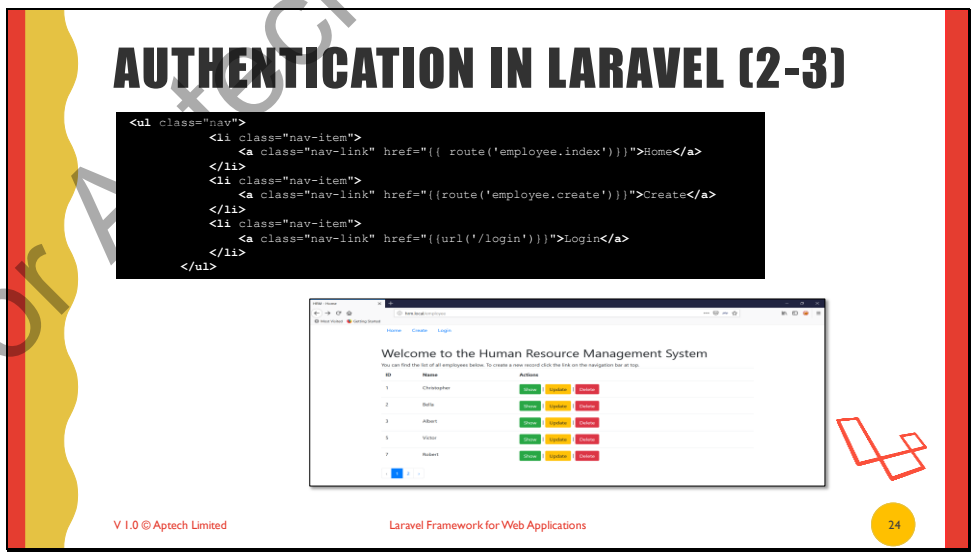


Instruction(s) to the trainer:
Using slide 23, explain the authentication feature in Laravel. In Laravel, authentication can be initiated as shown in first code snippet. It generates multiple Views and adds three authentication routes to the route file **routes/web.php**. The routes file appears as shown in the second code snippet.

Ask:
Explain the concept of authentication in a Web application.

Accept the responses from the students and discuss the concept in detail.

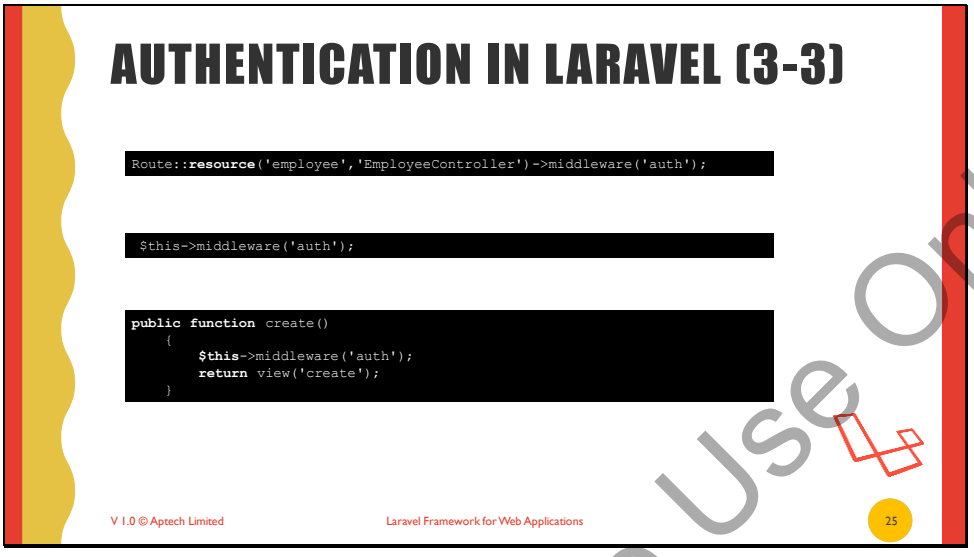
Show Slide 24



Instruction(s) to the trainer:
Using slide 24, explain that to add the login link to the Web pages, another link can be added in the navbar section of the master layout.

The code snippet shows the modifications. The image shows the login option on the Web page. Clicking the Login option displays a prebuilt login page and an option to register.

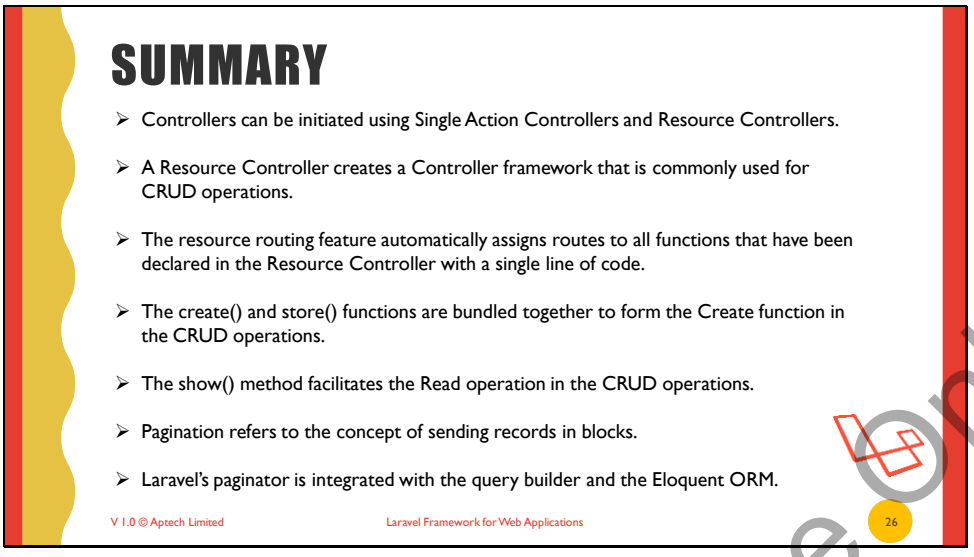
Show Slide 25



Instruction(s) to the trainer:
Using slide 25, explain that to make the routes accessible only by the registered users, Laravel provides an auth middleware. It can be directly in the Web routes file, routes/web.php, where the middleware method can be used on the resource route. This protects all routes defined by the resource route. The first code snippet shows an example of the same. Now, whenever a user visits the page without logging in, the user will be redirected automatically to the login page. To individually protect routes, edit the **EmployeeController** class, and add the code shown in the second code snippet. The final code for the **create()** method appears as shown in the third code snippet. Thus, now the code redirects all unauthenticated users to the login page.

To know more about authentication in Laravel, refer:
https://laravel.com/docs/5.8/authentication
https://www.tutorialspoint.com/laravel/laravel_authentication.

Show Slide 26



Instruction(s) to the trainer:

Using slide 26, summarize the key concepts from this session.

6.2 Post Class Activities for Faculty

You can ask additional questions to the students to check their understanding.

Tips: You can also ask students to check the Articles/Blogs/Expert Videos uploaded on the Onlinevarsity site to gain additional information related to the topics covered in this session.