Hachion

Salesforce Developer Course Content

Course Description:

Salesforce is a powerful cloud computing technology that provides SaaS - Software as a Service and PaaS - Platform as a Service. Salesforce is a leading CRM platform that bridges the gap between companies and customers by allowing business enterprises to manage marketing, e-commerce, sales, clients and services in one place. In simple words, Salesforce is a cloud-based CRM solution that provides service and enterprise application suite which offers data analytics, customer service, marketing automation, and app development.

Salesforce Developers are the IT professionals who build salesforce applications over several PaaS platforms. Within the Salesforce cloud-based platform salesforce developers build mobile and enterprise applications, accomplish integrations with third-party systems and customize salesforce software to meet clients' requirements as well as customer needs. Salesforce has a huge demand in the global market. To learn salesforce, there is no requirement of deep programming skills but basic understanding knowledge of HTML and java or any programming language will be added advantage to learn Apex, Visual force.

Hachion's Salesforce Developer course is prepared by professional masters with the structured modules. Beginners can easily learn the salesforce course to gain in-depth knowledge and may have a strong career as a Salesforce developer. By end of the course training, you will gain deep knowledge of salesforce technology and tools and further you can appear for the Salesforce developer certification exam to validate your skills and knowledge and get certified as a salesforce developer expert.

Course Content:

Fundamentals of Salesforce

- CRM basics and cloud computing
- Salesforce.com overview and foundation
- Salesforce platform, Salesforce terminology, and introduction to force.com platform
- Multi-tenancy and cloud
- Salesforce metadata and APIs
- Salesforce architecture
- The functionality of a central CRM object in Salesforce schema
- Determining declarative custom restrictions and programmatic custom use cases in a given scenario
- Declarative vs. Programmatic customizations
- Identify common scenarios for using the AppExchange extension organization
- OOP programming and database connections in Python

Data Modeling and Management

- Identify appropriate data models in a given scenario
- Capabilities of different relationship types and the impact of each record access, reporting, and user interface

- Identify considerations for changing the field type. Determine considerations and select the appropriate field type based on many requirements
- Describe the features and considerations of the schema builder
- Illustrate options and considerations for importing/exporting data
- Describe the features and use cases of external objects under Salesforce
- Trust, data model, field and relationship fields
- Pattern generator, junction object, and business logic
- Salesforce data management, data import wizard and data loader
- Export data, Apex class variables, and methods
- Class constructor, Apex, and access modifier
- Apex development tools, development process, and governor boundaries

Backing up and sharing data

- Describe the available features and ways to limit and extend objects, field access, and records
- Identify appropriate release solutions based on many business needs
- Salesforce application development tools
- Application development strategy using AppExchange
- Search and navigate Salesforce documents

Business logic & process automation

Describe the function and use cases of:

- Record type
- Formula fields
- Of roll-up summary fields
- Validation rules
- Approval process
- Workflow, visual workflow and Lightning process builder

Working with Chatter

- Chatter features
- Chatter use cases

Analyze data using reports and dashboards

- Report type and dashboard
- Identify the features available when creating a report

Salesforce development

- Describe development considerations in a multi-tenant environment
- Describe how the Salesforce platform features map to MVC patterns, sharing, ‘this’ keyword, inheritance, and interfaces
- System classes/methods, field API names, and standard objects

- Relationships in Apex; basic SOQL syntax, functions, variable binding, keywords, relational queries, and SOQL-for-loops, SOQL Debugging with debug logs, and anonymous blocking
- SOSL and SOSL
- Debugging tools
- Apex testing, testing classes, and unit testing
- System testing class methods
- Test classes, test data, unit testing using private variables and methods

Introduction to Apex Code

- Language constructs declaring variables & constants in Apex and assigning values using the various expressions
- Collection overview (lists, collections, and mappings), calling Apex, classes, interfaces, and objects
- Use and application of Apex control flow instructions
- Usage of basic SOSL, DML statements, and SOQL
- Apex design, dynamic and batch Apex
- Debug Apex, manage debug logs, and anonymous scripts
- Developer console, Force.com IDE, and workbench
- SaveResult classes, DML (Data Manipulation Language)
- DML & loop, DML options, Database, and transaction control
- SOQL and governor restrictions
- Apex triggers and execution order
- Trigger context variables; create and display triggers
- Recognize trigger events, helper class patterns, and batch triggers
- Handle recursion in triggers, using addError() and Apex trigger best practices to limit memory operations
- Exceptions, exception methods, and system-defined exceptions
- Handle user-defined or custom exception and intercept various exception types

Apex Triggers

- Governor limits, start/stop use, test Apex
- Bulkified code, test framework, create/load test data
- Use system.runAs to test configuration file security

Visualforce

- Describe how to write a Visualforce controller using JavaScript / HTML / CSS
- Use standard Visualforce controllers and custom Apex controllers/controller extensions
- Visualforce page, build method, development tool, tags, and their binding types & syntax
- Understand the MVC pattern
- Static resources in Visualforce
- Salesforce Lightning overview
- Lightning component framework structure and its benefits

Salesforce Lightning

- Salesforce Lightning tools and technology collections

- What is a Visualforce controller - standard controller, list controller?
- Custom controllers; custom list controllers, methods, extensions, and execution order
- Speed optimization of modern user interface speed optimization
- Salesforce Lightning experience and create reusable components to customize it
- Salesforce1 mobile application

Salesforce1 Lightning Process Builder

- Salesforce1 Introduction and its action layout
- Lightning process builder
- Use a visual layout to create a process and establish the entire process instead of using multiple workflow rules
- Collaborating for different users in the user interface

Sign-up of Salesforce account

- Two free accounts, one-lifetime free developer account, one 30-day trial production account, a registered lifetime developer account, and an account created at https://developer.force.com

Salesforce domain registration

- Create custom Salesforce domain
- Define the organization's custom domain name on https:// .lightning.force.com and test the domain
- Event-driven Lightning framework & Aura framework
- Deploy the domain using the developer console to create a lightning component

Use HTML and CSS styles

- Style Lightning components using HTML and CSS
- See the components in the Lightning App
- Use the developer console to create a CSS file
- Link to the lightning component and upload an external style sheet to a static resource

Attributes of Lightning component

- Understand Apex class's attributes and member variables
- Typed fields for component instances
- Use expression syntax to reference from component tags
- Use attributes dynamic components
- Add attributes to a component or application using the aura:attribute> tag

Lightning functions

- Client controller, functions, and Binding components
- Pass values and JavaScript-based operations
- Application attributes and reference components in the application

Component Composition

Learn to create:

- Simple components such as c:LC and c:LC2
- Fine-grained components in larger components by assembling each component
- Wrapper component with simple components

Conditional Statements

- Conditional If-else statement, isTrue expression, aura:if, component instantiating in its body or in the Else attribute

Use of Value Providers

- Use value providers to access data, combine related values, component v (view) and c (control) value providers

Out-of-the-box components

- Use out-of-the-box Salesforce1 components
- Lightning experience and its applications
- Different namespaces components, Force, Force Community, Aura, Force Chatter, etc.
- Style Lightning system, user interface, and namespace component

Generic UI & UI namespace

- Universal user interface, use of aura: components, UI components
- Handle common user interfaces and component extensions & designs

Dynamic update of UI components

- Dynamic update of UI components
- Trigger JavaScript controller actions for events in Lightning component
- Event declarations with aura:event tag

Handling events

- JavaScript and Java Swing for event-driven programming
- Component and application events
- Write JavaScript controller actions handlers

Server-side controller

- Create a server-side controller in Apex, Aura annotation
- Annotation that allow server-side and client access to controlled methods
- Invoke server-side controller operations from the client controller

Server-side controller hands-on

- A real-world example of fetching data from an opportunity object via server-side controller deployment
- Create the client controller to call the server-side controller and pass the returned value to the component

Create a Lightning application

- Create a contact management app on the Lightning Framework
- Upload CSS files and upload them to static resources for reference in the app

- Create a contact-controller Apex class
- Retrieve data from contacts and return to the contact list
- Design user interfaces using HTML & CSS classes provided in the style sheet (bootstrap)

Salesforce Lightning Project

- Create a component to call the client controller function and process the data
- Retrieve the contact details as per the ID
- Create a component to handle the event
- Create a client controller to bind the user event
- Call the Apex controller method
- Add fields & queries to the components
- And displays them on the screen