Lightning Aura Component

Lightning Component è un framework UI per lo sviluppo di applicazioni a pagina singola per dispositivi mobili e desktop.
L'Interfaccia Lightning include LightningComponentFramework e strumenti per gli sviluppatori.
Lightning semplifica la creazione di applicazioni reattive per qualsiasi dispositivo.

Lightning Experience

Lightning Experience nasce al fine di reinventare l'ambiente desktop per supportare meglio i processi aziendali.
Il risultato è un'interfaccia intuitiva e intelligente che aiuta i team di vendita e di assistenza a lavorare in modo più naturale e produttivo.
Quando si parla di Lightning Experience, si parla di pagine di Salesforce ottimizzate per il team di vendita e assistenza.

Lightning Experience: Opportunities

Per Opportunity Workspace, è stato preso il processo di vendita ed è stato inserito in uno spazio di lavoro ottimizzato per l'azione, progettato per aiutare i rappresentanti a

lavorare più velocemente e in modo più intelligente.
Questo Workspace, oltre a visualizzare sia le proprie opportunità di vendita che altre, è stato realizzato per favorire il processo di vendita e quindi velocizzare il lavoro ed è una funzione disponibile solo in Lightning Experience.

Lightning components

Gli elementi che compongono Lightning Experience sono componenti Lightning, sono riutilizzabili nell'intera interfaccia,quindi è facile includere solo ciò che è necessario tramite il **Lightning App Builder.**
Gli amministratori possono aggiungere, rimuovere e riordinare i componenti sulla home page, o comporre le pagine con facilità attraverso il drag'n'drop.

Lightning Experience: Developers

I Developers, hanno la possibilità di creare componenti personalizzati di Lightning utilizzando i blocchi predefiniti inclusi nel framework.

Salesforce Lightning Design System

Salesforce Lightning Design System è una libreria di stili completa, in modo che gli sviluppatori non debbano scrivere il proprio codice HTML o CSS.

Lightning Experience: Data Service

Lightning Data Service consolida il codice di accesso ai dati in componenti riutilizzabili, eliminando la necessità di scrivere il codice del controller, velocizzando la programmazione e riducendo il tempo circa le prestazioni e test di qualità.

Domain

È necessario distribuire il dominio personalizzato nell'organizzazione se si desidera utilizzare i componenti di Lightning tabs e Lightning pages come app autonome, azioni e sostituzioni di azioni, modelli personalizzati di Lightning o altrove nell'organizzazione.

Aura Components

Il suo vantaggio è un set di componenti pronti all'uso,un'architettura basata sugli eventi e un framework ottimizzato per le prestazioni.

Questo vantaggio è dato dai componenti Aura, che sono dei componenti riutilizzabili e

componibili, in quanto un componente può contenere altri componenti, nonché HTML, CSS, JavaScript o qualsiasi altro codice abilitato per il Web.
Ciò consente di creare App con interfacce utente sofisticate.
I Dettagli dell'implementazione di un componente sono incapsulati.
Ciò consente al consumatore di un componente di concentrarsi sulla creazione della propria app, mentre l'autore del componente può innovare e apportare modifiche.

Events

La programmazione basata su eventi viene utilizzata in molti linguaggi e framework, come JavaScript e Java Swing.
L'idea è di scrivere controller che rispondano agli eventi dell'interfaccia .
Gli eventi vengono generati dalle azioni del controller JavaScript che in genere vengono attivate da un utente che interagisce con l'interfaccia utente.

Esistono due tipi di eventi nel framework:

- **Component events**
  Sono gestiti dal componente stesso o da un componente che crea un'istanza o contiene il componente.
- **Application events**
  Sono gestiti da tutti i componenti che ascoltano l'evento.

Developer Console

La Developer Console è uno strumento pratico e integrato che è possibile utilizzare per creare nuovi e modificare i componenti Aura esistenti e altri pacchetti.
Quando si crea un' applicazione Lightning, si visualizza nella console a destra una tabella che permette la creazione di ulteriori risorse ad essa collegate:

- Application/Component
- Controller
- Helper
- Style
- Documentation
- Renderer
- Design
- SVG

Component

Le risorse del componente contengono markup.
Il markup può contenere testo o riferimenti ad altri componenti e dichiara anche i metadati relativi al componente.
Il nome di un componente deve rispettare le seguenti proprietà:
• Deve iniziare con una lettera.
• Deve contenere solo caratteri alfanumerici o di sottolineatura.
• Deve essere univoco nello spazio dei nomi.
• Non è possibile includere spazi bianchi.
• Non può finire con un trattino basso.
• Non può contenere due trattini bassi consecutivi.

Un componente ha due tipi di ID:

- ID locale.
  Un ID locale è un ID che vale solo per il componente, spesso è univoco ma non necessariamente.

```
<lightning:button aura:id="button1" label="button1"/>
```

- ID globale.
  Ogni componente ha un unico **globalId**, che è l'ID univoco di runtime generato dall'istanza del componente.
  Un ID globale può essere utile per distinguere tra più istanze di un componente o per scopi di debug.
  Si consideri il seguente esempio per creare un id globale:

  ```
  <div id="{!globalId + '_footer'}"></div>
  ```

Create Aura Components

Il pannello New Lightning Bundle nella Developer Console offre una scelta di configurazioni dei componenti, quando si crea un componente Aura, o un pacchetto di applicazioni.
Le configurazioni aggiungono le interfacce necessarie per supportare l'utilizzo del componente nel contesto desiderato.

- **Lightning Tab**
  Crea un componente da utilizzare come elemento di navigazione in Lightning Experience o nelle app mobili Salesforce.

- **Lightning Page**
  Crea un componente da utilizzare nelle pagine Lightning o Lightning App Builder.
- **Lightning Record Page**
  Crea un componente da utilizzare in una home page del record in Lightning Experience.
- **Lightning Communities Page**
  Crea un componente disponibile per il trascinamento della selezione nel Community Builder.
- **Lightning QuickAction**
  Crea un componente che può essere utilizzato con un'azione rapida Lightning.

HTML components

Un tag HTML viene trattato come un componente di prima classe dal framework. Ogni tag HTML è tradotto in un**<Aura: html>** component, consentendogli di godere degli

stessi diritti e privilegi di qualsiasi altro componente.

<aura:html tag="div" />

Controller

Controller lato Client-Side con metodi che rispondano a eventi come interazioni dell'utente o eventi del ciclo di vita

Controller lato Server-Side con azioni eseguite sul server ed eseguendo l'elaborazione come le interazioni col database.

Helper

Classe progettata per avere codice condiviso in esso.

Quindi, potresti avere il codice condiviso dal tuo renderer e dal tuo controller o da più funzioni del controller.

Style

Classi che descrivono come devono essere visualizzati gli elementi HTML: Cascading StyleSheets (CSS)

STYLE Resource

La risorsa Style viene utilizzata per definire proprietà CSS personalizzate.

La parola chiave "**THIS**" raggruppa tutte le classi utilizzate nel proprio componente.

Per definire nuove classi, o per sovrascrivere le proprietà per una esistente, è necessario specificare il nome:

CSS components

È possibile aggiungere stile ai propri componenti utilizzando/aggiungendo il CSS a un gruppo di componenti facendo clic sul pulsante **STYLE** nella barra laterale della Console per gli sviluppatori.

Errori nella Developer Console

Gli errori di convalida vengono trattati come errori e impediscono il salvataggio delle modifiche.

I messaggi di errore spiegano gli errori.

A seconda dello strumento in uso, questi errori vengono presentati in diversi modi.

Ad esempio, la Console per gli sviluppatori mostra un avviso per il primo errore riscontrato (1) ed elenca tutti gli errori di convalida rilevati nella scheda **Problemi** (2).

Lightning Application

Le applicazioni lightning personalizzate sono generalmente strutturate come un'applicazione principale contenente una gerarchia di componenti lightning.

<aura:application>

Un'app è uno speciale componente di livello superiore il cui markup si trova in una risorsa.app.
Il markup è simile all'HTML e può contenere componenti e un insieme di tag HTML supportati.
<aura: application> di livello superiore, contiene attributi di sistema opzionali.
Questi attributi di sistema comunicano al framework come configurare l'app.

- Access
  Indica se l'app può essere estesa da un'altra app al di fuori di uno spazio dei nomi.
  I valori possibili sono public (predefinito) e global.
- Controller
  La classe del controller lato server per l'app.
  Il formato è namespace.myController.
- Description
  Una breve descrizione dell'app.
- Extends
  L'app da estendere, se applicabile.

Ad esempio:
extends="namespace:yourApp".

- Extensible
  Indica se l'app è estensibile da un'altra app.
  Il valore predefinito è falso.
- Implements
  Un elenco di interfacce separate da virgole implementate dall'app.
  Include un attributo body definito nel tag:Il corpo dell'app.
  Nel markup, questo è tutto nel corpo del tag.

<aura:component>
Qualsiasi markup non incluso in uno dei tag ammessi in un componente viene considerato parte del corpo e impostato nell'attributo body.
L'attributo body ha type Aura.Component [].
Può essere un array di un tipo di componente o un array vuoto, ma comunque è sempre un array.
In un componente, utilizzando "v" è possible accedere alla raccolta di attributi.

Ad esempio, {! v.body} restituisce il corpo del componente.

Component Attributes

Gli attributi dei componenti sono come variabili membro su una classe in Apex. Sono campi tipizzati impostati su un'istanza specifica di un componente e possono essere consultati all'interno del markup del componente utilizzando una specifica sintassi.
Gli attributi consentono di rendere i componenti più dinamici.
Utilizzando il tag **<aura: attribute>** è possibile aggiungere un attributo al componente o all'app.
Hanno i seguenti attributi :

- Access
  Indica se l'attributo può essere utilizzato al di fuori del proprio spazio dei nomi.
  I valori possibili sono public (predefinito), global o privat.
- Name
  E Required e Indica Il nome dell'attributo.
  Ad esempio, se si imposta <aura: attribute name = "isTrue" type = "Boolean" /> su un componente

denominato aura: newCmp, è possibile impostare questo attributo quando si crea un'istanza del componente; per esempio,<aura:newCmp isTrue="false" />.

- type
  E **Required e indica** Il tipo dell'attributo.Per un elenco di tipi base supportati.

- Basic Types
  Boolean, Date, DateTime, Decimal, Double,
  Integer, Long, String
  - Function Type
    Un attributo può avere un tipo corrispondente
    a una funzione JavaScript.
  - Object Types
    può avere un tipo corrispondente
    a una Superclass Salesforce Standard
  - Types
    Un attributo può avere un tipo corrispondente a un oggetto standard o personalizzato in Salesforce.
  - Collection Types

type[] (Array), List, Map, Set
- Custom Apex Class Types
  Custom Class definite in Salesforce
- Default
- Il valore predefinito per l'attributo, che può
  essere sovrascritto secondo necessità.
  Quando si imposta un valore predefinito, le espressioni utilizzano il $Label $Locale, e $Browser i fornitori di valore globale sono supportati.
  In alternativa, per impostare un valore predefinito dinamico, utilizzare un evento init.

- Required
  Determina se l'attributo è obbligatorio. L'impostazione predefinita è false.
- Description
  Un riepilogo dell'attributo e del suo utilizzo.
  Descrive un attributo utilizzabile nelle app, nelle interfacce, nei componenti o negli eventi.