Sfdc Telugu

createLead 1

To create a Lead in Salesforce using Apex, you can follow these steps:

- Define a method in an Apex class to create the Lead.
- Use the insert DML statement to insert the Lead record into the Salesforce database.

Apex code:

```
public class LeadCreator {

    // Method to create a Lead
    public static void createLead(String firstName, String lastName, String company,
                                  String email, String phone) {
        try {
            // Create a new Lead record
            Lead newLead = new Lead();
            newLead.FirstName = firstName;
            newLead.LastName = lastName;
            newLead.Company = company;
            newLead.Email = email;
            newLead.Phone = phone;

            // Insert the new Lead record
            insert newLead;

            System.debug('Lead created successfully with Id: ' + newLead.Id);
        } catch(Exception e) {
            System.debug('Error creating lead: ' + e.getMessage());
        }
    }
}
```

In this code:

- We have a method named createLead that accepts parameters such as firstName, lastName, company, email, and phone.
- Inside the method, we create a new instance of the Lead standard object and assign values to its fields using the parameters passed to the method.
- We then use the insert statement to insert the new Lead record into the database.
- Any errors that occur during the lead creation process are caught in a try-catch block, and the error message is logged to the debug logs.
- To use this Apex class, you would need to call the createLead method from somewhere in your Salesforce code, passing in the necessary parameters.

For example:

```
// Call the createLead method to create a new lead
LeadCreator.createLead('Sai', 'Ritesh', 'SFDC TELUGU',
                       'sfdctelugu@gmail.com', '7997891996');
```

- This would create a new lead record in Salesforce with the specified details.
- Adjust the field names and data types according to your Salesforce org's Lead object structure.

Sfdc Telugu

updateLead 2

To update a Lead in Salesforce using Apex, you can follow these steps:

- Define a method in an Apex class to update the Lead.
- Use the update DML statement to update the Lead record in the Salesforce database.

Apex code:

```
public class LeadUpdater {

    // Method to update a Lead
    public static void updateLead(String leadId, String newFirstName, String
                                  newLastName, String newCompany, String
                                  newEmail, String newPhone) {
        try {
            // Retrieve the Lead record to be updated
            Lead leadToUpdate = [SELECT Id, FirstName, LastName, Company,
                                 Email, Phone FROM Lead WHERE Id = :leadId LIMIT 1];

            // Update the Lead record fields
            leadToUpdate.FirstName = newFirstName;
            leadToUpdate.LastName = newLastName;
            leadToUpdate.Company = newCompany;
            leadToUpdate.Email = newEmail;
            leadToUpdate.Phone = newPhone;

            // Update the Lead record in the database
            update leadToUpdate;

            System.debug('Lead updated successfully with Id: ' + leadToUpdate.Id);
        } catch(Exception e) {
            System.debug('Error updating lead: ' + e.getMessage());
        }
    }
}
```

In this code:

- We have a method named updateLead that accepts parameters such as leadId, newFirstName, newLastName, newCompany, newEmail, and newPhone.
- Inside the method, we query the Lead record based on the provided leadId to retrieve it from the database.
- We then update the Lead record's fields with the new values provided as parameters.
- Finally, we use the update statement to update the Lead record in the database.
- Any errors that occur during the lead update process are caught in a try-catch block, and the error message is logged to the debug logs.
- To use this Apex class, you would need to call the updateLead method from somewhere in your Salesforce code, passing in the necessary parameters.

For example:

```
// Call the updateLead method to update an existing lead
LeadUpdater.updateLead('LeadId', 'NewSai', 'NewRitesh',
                       'salesforce.com', 'ritesh@sfdctelugu.in', '7997891996');
```

- Replace '**LeadId**' with the Id of the lead you want to update and adjust the field names and data types according to your Salesforce org's Lead object structure.

Sfdc Telugu

deleteLead 3

To delete a Lead in Salesforce using Apex, you can follow these steps:

- Define a method in an Apex class to delete the Lead.
- Use the delete DML statement to delete the Lead record from the Salesforce database.

Apex code:

```
public class LeadDeleter {

    // Method to delete a Lead
    public static void deleteLead(String leadId) {
        try {
            // Retrieve the Lead record to be deleted
            Lead leadToDelete = [SELECT Id FROM Lead WHERE Id = :leadId LIMIT 1];

            // Delete the Lead record from the database
            delete leadToDelete;

            System.debug('Lead deleted successfully with Id: ' + leadId);
        } catch(Exception e) {
            System.debug('Error deleting lead: ' + e.getMessage());
        }
    }
}
```

In this code:

- We have a method named **deleteLead** that accepts a parameter **leadId**, which represents the Id of the Lead record to be deleted.
- Inside the method, we query the Lead record based on the provided **leadId** to retrieve it from the database.
- We then use the delete statement to delete the Lead record from the database.
- Any errors that occur during the lead deletion process are caught in a try-catch block, and the error message is logged to the debug logs.
- To use this Apex class, you would need to call the **deleteLead** method from somewhere in your Salesforce code, passing in the Lead Id of the lead you want to delete.

For example:

```
// Call the deleteLead method to delete an existing lead
LeadDeleter.deleteLead('LeadId');
```

- Replace '**LeadId**' with the Id of the lead you want to delete.
- Adjust the field names and data types according to your Salesforce org's Lead object structure.

Sfdc Telugu

assignLeadOwner 4

To assign a new owner to a Lead in Salesforce using Apex, you can follow these steps:

- Define a method in an Apex class to assign the new owner to the Lead.
- Use the OwnerId field of the Lead object to set the new owner's Id.
- Use the update DML statement to update the Lead record in the Salesforce database.

Apex code:

```
public class LeadOwnerAssigner {

    // Method to assign a new owner to a Lead
    public static void assignLeadOwner(String leadId, String newOwnerId) {
        try {
            // Retrieve the Lead record to be updated
            Lead leadToAssign = [SELECT Id FROM Lead WHERE Id = :leadId LIMIT 1];

            // Set the new owner's Id for the Lead
            leadToAssign.OwnerId = newOwnerId;

            // Update the Lead record in the database
            update leadToAssign;

            System.debug('Lead owner assigned successfully. Lead Id: ' + leadId + ',
                         New Owner Id: ' + newOwnerId);
        } catch(Exception e) {
            System.debug('Error assigning lead owner: ' + e.getMessage());
        }
    }
}
```

In this code:

- We have a method named **assignLeadOwner** that accepts parameters **leadId** and **newOwnerId**, representing the Id of the Lead record and the Id of the new owner, respectively.
- Inside the method, we query the Lead record based on the provided leadId to retrieve it from the database.
- We then set the **OwnerId** field of the Lead to the provided **newOwnerId**.
- Finally, we use the update statement to update the Lead record in the database with the new owner.
- To use this Apex class, you would need to call the **assignLeadOwner** method from somewhere in your Salesforce code, passing in the Lead Id and the Id of the new owner.

For example:

```
// Call the assignLeadOwner method to assign a new owner to an existing lead
LeadOwnerAssigner.assignLeadOwner('LeadId', 'NewOwnerId');
```

- Replace '**LeadId**' with the Id of the lead you want to assign a new owner to, and '**NewOwnerId**' with the Id of the new owner.
- Adjust the field names and data types according to your Salesforce org's Lead object structure.

Sfdc Telugu

convertLead 5

- To convert a Lead in Salesforce using Apex, you can leverage the Database.convertLead() method provided by Salesforce. This method allows you to convert a Lead into an Account, Contact, and optionally, an Opportunity.

Apex code:

```
public class LeadConverter {

    // Method to convert a Lead
    public static void convertLead(String leadId) {
        try {
            // Retrieve the Lead record to be converted
            Lead leadToConvert = [SELECT Id FROM Lead WHERE Id = :leadId LIMIT 1];

            // Create an instance of Database.LeadConvert
            Database.LeadConvert leadConvert = new Database.LeadConvert();
            leadConvert.setLeadId(leadToConvert.Id);

            // Optionally, specify whether to create an Opportunity during conversion
            // leadConvert.setDoNotCreateOpportunity(false);

            // Optionally, specify an Opportunity name during conversion
            // leadConvert.setOpportunityName('New Opportunity');

            // Convert the Lead record
            Database.LeadConvertResult leadConvertResult = Database.convertLead(leadConvert);

            // Check if the Lead conversion was successful
            if (leadConvertResult.isSuccess()) {
                System.debug('Lead converted successfully. Converted Account Id: '
                             + leadConvertResult.getAccountId() +
                             ', Converted Contact Id: ' + leadConvertResult.getContactId());
            } else {
                System.debug('Failed to convert lead. Error: ' +
                             leadConvertResult.getErrors()[0].getMessage());
            }
        } catch(Exception e) {
            System.debug('Error converting lead: ' + e.getMessage());
        }
    }
}
```

In this code:

- We have a method named convertLead that accepts a parameter leadId, representing the Id of the Lead record to be converted.
- Inside the method, we query the Lead record based on the provided leadId to retrieve it from the database.
- We create an instance of Database.LeadConvert and set the Lead Id to be converted using setLeadId() method.
- Optionally, you can set other properties such as setDoNotCreateOpportunity() and setOpportunityName() to control the conversion process.
- We then use the Database.convertLead() method to convert the Lead record.
- Finally, we check the result of the conversion and log the outcome.
- To use this Apex class, you would need to call the convertLead method from somewhere in your Salesforce code, passing in the Lead Id of the lead you want to convert.

For example:

```
// Call the convertLead method to convert an existing lead
LeadConverter.convertLead('LeadId');
```

- Replace '**LeadId**' with the Id of the lead you want to convert.
- Adjust the field names and data types according to your Salesforce org's Lead object structure.

Limited TIME Offer

SFDC TELUGU STORE ACCESS FOR LIFE TIME

స్టోర్ లో ఇప్పుడున్న ప్రతి **PDF** లతో పాటు కొత్తగా **UPLOAD** చేసే ప్రతి **PDF** మీ యొక్క మెయిల్ కి లైఫ్ టైం వస్తూనే ఉంటాయి.

ONE TIME SUBSCRIPTION

~~RS.4,499~~

JUST RS.2,249

WWW.SFDCTELUGU.IN

ప్రస్తుతం, **SFDC STORE**లో ₹**1500** విలువగల **PDF** లు ఉన్నాయి.

Buy Now

SFDC STORE

www.sfdctelugu.in/store

VISA MasterCard PayPal RuPay G Pay paytm PhonePe BHIM
OLAMONEY JioMoney freecharge MobiKwik NEFT IMPS NET BANKING BANK TRANSFER

https://www.youtube.com/@sfdctelugu