Learning Force.com Application Development

Starting with an introduction to the Force.com model, key features, and advantages of the Force.com platform, this book delves into the design and development of applications on the Force.com platform. Particular emphasis is given to building the data model and user interfaces of the Force.com application, designing the Force.com application for multiple users, and protecting data related to your application.

This book also covers the most important features of the Force.com platform such as preserving data quality with validation rules, automating the business process by using workflows and approval processes, data management operations and tools, reports and dashboards, e-mail service related features of the Force.com platform, and deploying a Force.com application. Through the course of the book, you will learn not only declarative (point-click) developments, but also programmatic developments such as building custom pages using Visualforce and custom coding using Apex. By the end of the book, you will have learned about the Force.com tools including the Force.com IDE and standard Visualforce components that are used to develop build Visualforce pages.

Who this book is written for
If you are a developer who wants to learn how to develop and deploy applications from the Salesforce.com platform, then this book is for you. No prior knowledge of Salesforce is necessary.

What you will learn from this book
- Design, develop, customize, and deploy applications on the Force.com platform
- Build a data model and standard user interfaces on the Force.com platform
- Develop Force.com applications for multiple users and protect your organization’s data
- Implement the application’s business processes using Validation Rules, Approval Processes, and Workflows
- Use Data Management operations and tools on the Force.com application
- Make custom pages using Visualforce and custom controllers using Apex
- Create reports, dashboards, and e-mail services on the Force.com platform
- Construct Force.com sites that allow the application to be extended out into the broader population of users through the use of a public website
- Employ the tools available on the Force.com platform


$ 59.99 US
£ 39.99 UK


7.50 x 9.25
235 mm x 191 mm

Chamil Madusanka

Learning Force.com Application Development

Use the Force.com platform to design and develop real-world, cutting-edge cloud applications

Chamil Madusanka
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 2 'Building the Data Model'
- A synopsis of the book’s content
- More information on Learning Force.com Application Development

About the Author

Chamil Madusanka is a Salesforce.com-certified Force.com developer. He has been working on Force.com projects since 2011. He works as a developer on many custom applications built on Force.com and has also trained end users and new Salesforce developers at his current company (attune Lanka (Pvt) Ltd.) and former company (Sabre Technologies (Pvt) Ltd). He has authored Visualforce Developer's Guide, Packt Publishing.

Chamil won the Salesforce New Year Resolution 2013 challenge, which was rolled out by Salesforce. He is an active member of the Force.com community and contributes through various channels. He is passionate about Force.com and shares his knowledge of Force.com technologies through his blog (http://salesforceworld.blogspot.com/). He is a super-contributor on the Force.com discussion board and shares his knowledge and experience on Force.com by providing effective answers to developer questions. He is the initiator and organizer of the Sri Lanka Salesforce Platform Developer User Group. His contribution to the Sri Lanka Salesforce community has led to an increase in Salesforce competency in Sri Lanka.

He completed his BSc in computer science from the University of Colombo, School of Computing, Sri Lanka (UCSC). His areas of interest include cloud computing, semantic web technologies, and Ontology-based systems. Hailing from Polonnaruwa, an ancient city in Sri Lanka, he currently resides in Gampaha in the Western province of Sri Lanka. His interests include reading technology books and technology blog posts, and playing cricket. Chamil can be reached via twitter at @chamilmadusanka, Skype at chamilmadusanka, and e-mail at chamil.madusanka@gmail.com.
Learning Force.com Application Development

*Learning Force.com Application Development* is a hands-on guide aimed at developing Force.com applications on the Force.com platform. As you read through the content, you will notice that this book focuses on a single real-world example. This book builds upon this example to help you understand and use the tools and features of the Force.com platform.

Cloud computing has made significant changes to the IT/software development industry. Cloud platforms are one of the important directions of cloud computing. Cloud platforms allow the developers to develop apps and run them on the cloud, including platforms for building on-demand applications and platforms as a service (PaaS). Salesforce.com has introduced the first on-demand platform, called Force.com.

**What This Book Covers**

*Chapter 1, Getting Started with Force.com*, introduces the Force.com platform and explains the design and development aspects of the Force.com platform. The sample application scenario and ERD are introduced at the end of this chapter.

*Chapter 2, Building the Data Model*, introduces the data model of the Force.com platform and explains about creating custom objects, custom fields, and the various data types of the Force.com platform.

*Chapter 3, Building the User Interface*, explains how to create custom applications, custom tabs, and customized page layouts. The overview of Visualforce will be there at the end of the chapter and that overview will link to *Chapter 8, Building Custom Pages with Visualforce.*

*Chapter 4, Designing Apps for Multiple Users and Protecting Data*, explains the security aspects of the Force.com platform. That means design considerations for Force.com applications, which are accommodating multiple users and the security framework of the Force.com platform.

*Chapter 5, Implementing Business Processes*, describes the capabilities of data validation rules and the ways of automating business processes with workflows and approval processes. The troubleshooting of automated processes are explained at the end of the chapter.

*Chapter 6, Data Management on the Force.com Platform*, describes typical data management operations and data management tools. There is a section to talk about the record IDs and a consideration of the object relationships in data management.
Chapter 7, Custom Coding with Apex, introduces custom coding on the Force.com platform. Apex controllers and Apex triggers will be explained with examples of the sample application. The Force.com platform query language and data manipulation language will be described with syntaxes and examples.

Chapter 8, Building Custom Pages with Visualforce, introduces Visualforce, the architecture of Visualforce, the advantages of Visualforce and the use of Visualforce. There are sections to describe Visualforce pages and Visualforce controllers with examples of the sample application. Finally, there is an explanation section on Visualforce custom components.

Chapter 9, Analytics as a Service with the Force.com Platform, introduces the reports and dashboard of the Force.com platform and analytical features of the Force.com platform.

Chapter 10, E-mail Services with the Force.com Platform, introduces the e-mail feature of the Force.com platform that allows the extension of the application out to a broader population of users, through the use of e-mails.

Chapter 11, Building Public Websites with Force.com Sites, introduces the Force.com sites that allow extending the application out into the broader population of users, through the use of the public website.

Chapter 12, Deploying the Force.com Application, introduces deployment methodologies that allow for the distribution of Force.com applications to other end users.

Appendix, Force.com Tools, explains Force.com tools by looking at the Force.com IDE.
Building the Data Model

In this chapter, the data model of the Force.com platform will be introduced, which explains about creating custom objects, custom fields, and various data types of the Force.com platform.

This chapter covers the following topics:

- Creating objects
- Creating custom fields

Before we start building the data model on the Force.com platform, we need to log into our organization using the login credentials specified in Chapter 1, Getting Started with Force.com. You will see the Force.com home page (or landing page) for developers and administrators after you have logged on:
Creating objects

We have already introduced our leave management application, which we will continue looking at in future chapters. In the previous chapter, we introduced the objects as one of the building blocks of the Force.com platform. An object is a container used to store particular object data in the Force.com platform. By creating an object, we can create user interfaces to add/edit/view the records of a particular object. With the creation of an object, the Force.com platform provides us the related user interfaces to add, edit, and view records of the particular object.

Time saving feature
Create an object and gain a bunch of UI (standard pages) for various tasks such as add, edit, view, list view, and lookup view.

The objects of Salesforce.com have the following features:

- Configurable
- Relational
- Reportable
- Searchable
- Securable

There are two types of objects on the Force.com platform:

1. Standard Objects
2. Custom objects

Standard objects

The Force.com platform comes with a set of standard objects that are part of Salesforce Customer Relationship Management (Salesforce CRM). Standard objects are created by Salesforce.com. A standard object has a set of standard functionalities and properties, which are pre-defined and are Salesforce CRM-oriented. However, we can do customizations up to some extent and that customization is much less than custom objects. Let's consider the following example:

- Custom fields can be added to standard objects (custom fields will be described in the next few sections)
- Modify the existing pick list value
- Page layouts can be added
• Standard fields cannot be deleted
• Standard objects cannot be deleted

All the standard objects are categorized under the **Customize** link. The following screenshot shows the place where we can find the standard objects:

![Screenshot showing standard objects under customize](image)

The standard objects are under the customize category.

You can find the standard objects by navigating to **Setup** | **Customize**.

**Custom objects**

Custom objects are created by the developer according to the requirements of the application. The custom objects are more important building blocks because they can be created according to our requirements of the application. A custom object is more customizable than a standard object. A set of fields that are created in a custom object creation are called standard fields. All the other fields that are created by the developer according to the requirements are called custom fields. We will discuss more about fields in the next couple of sections.
All the custom objects are categorized under the **Create** link. The user can create custom objects through schema builder or the old fashioned way by clicking on the **New Custom Object** button. The following screenshot shows the place where we can find the custom objects.

You can find the custom objects by navigating to **Setup** | **Create** | **Objects**.

![Custom Objects](image)

The custom objects are categorized under the Create category

The design of the data model behind an app is typically the biggest factor in its success or failure.

Let's consider our leave management application and defined objects. According to the design, there are five custom objects for our leave management application. These five custom objects are used to track the date of the leave management application:

- Employee
- Holiday Calendar
- Leave Type
- Leave Category
- Leave Configuration
That's enough talk about objects for now. Let's go define one!

Although we have to create all of the preceding objects for the leave management app, let's create the Employee object as the example of creating a custom object. Use the following steps to create the Employee object:

1. Go to **Setup** | **Build** | **Create** | **Objects**.
2. Click on the **New Custom Object** button. You will see the following screenshot:

![Custom Object Definition](image)

The custom object definition create/edit page—upper section

The employee's object-related information is captured in the upper section of the object definition page.
The preceding screenshot shows the upper section of the object definition page. Let's see the information that is captured in this section. For ease of explanation, the screenshot of the object definition page has been divided into an upper section and a lower section. The content of the sections are as follows:

- **Label**: This is the label of the object definition page. This label is seen on page layouts and reports. Here we have used `Employee` as the label of this object.

- **Plural Label**: This label is used in the Salesforce object tabs. In this case, we have used `Employees` as the plural label.

- **Starts with a vowel sound**: Check whether the label `Starts with a vowel sound` should be preceded by "an" instead of "a".

- **Object Name**: The object name (otherwise API name) must be a unique name because it is used to refer to the object via the Force.com API. When the object name is accessed from the API, it will be appended with `__c` to the object name. For example, in the preceding screenshot, we have used `Employee` as the object name, and the API name will be `Employee__c`. This is automatically done by Force.com. This `__c` is the key to identify a custom object from a standard object. The standard objects don't have any `__c` suffix in their API name.

  The object name must be unique, should begin with a letter, should not include spaces, should not end with an underscore, and should not contain two consecutive underscores.

- **Context-Sensitive Help Setting**: There are two options to select the user help page. You can get the standard Salesforce help page by selecting the **Open the standard Salesforce.com Help & Training window** option or you can select a custom Visualforce page by selecting the **Open a window using a Visualforce page** option. From the **Open a window using a Visualforce page**, you can provide your own help page rather than using a standard Salesforce.com help window.
The lower section of the object definition page, including the record name label and format, the deployment status, and other value-added optional features will be captured. This is shown in the following screenshot of the lower section of the object definition page.

![The custom object definition create/edit page—lower section](image)

The lower section of the object definition page contains the following fields:

- **Record Name**: The record name is used everywhere the particular object records appear (such as page layouts, search results, list views, and related lists). We can specify the record name as our desire but the API name of the record can always be accessed as the name.
• **Data Type:** This field specifies the data type of **Record Name**. There are two options (text or autonumber) to select the data type of record name. If you choose the **Text** type, the end user must enter the record name for the particular object record. If you choose the **Auto Number** option, then you have to specify the following two additional fields:

  ° **Display Format:** A display format allows you to control the format of the **Auto Number** field. A display format consists of the substitution variables described in the following table, plus any other characters you wish to include as a prefix or suffix. The following table shows the substitution variables in display format:

<table>
<thead>
<tr>
<th>Display Format</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>{0}</td>
<td>Required</td>
<td>This is the sequence number. One or more zeros enclosed in curly braces represent the sequence number itself. The number of zeros in the curly braces dictates the minimum number of digits that will be displayed. If the actual number has fewer digits than this, it will be padded with leading zeros. The maximum is 10 digits.</td>
</tr>
<tr>
<td>{YY}</td>
<td>Optional</td>
<td>This is the year. Two or four Y characters enclosed in curly braces represent the year of the record creation date. You can display 2 digits (for example, &quot;04&quot;) or all 4 digits (for example, &quot;2004&quot;) of the year.</td>
</tr>
<tr>
<td>{YYYY}</td>
<td>Optional</td>
<td>This is the year. Two or four Y characters enclosed in curly braces represent the year of the record creation date. You can display 2 digits (for example, &quot;04&quot;) or all 4 digits (for example, &quot;2004&quot;) of the year.</td>
</tr>
<tr>
<td>{MM}</td>
<td>Optional</td>
<td>This is the month. Two M characters enclosed in curly braces represent the numeric month (for example, &quot;01&quot; for January, &quot;02&quot; for February) of the record creation date.</td>
</tr>
<tr>
<td>{DD}</td>
<td>Optional</td>
<td>This is the day. Two D characters enclosed in curly braces represent the numeric day of the month (for example, &quot;01&quot; to &quot;31&quot; are valid days in January) of the record creation date.</td>
</tr>
</tbody>
</table>

If you modify the display format of an existing field, only records created after the change will be affected.

° **Starting Number:** You can specify the starting number of the autonumber format. It will start the counting with that starting number. For example, if you use the display format of the Employee object as EMP-{0} and the starting number as 1, the Name field of Employee records will be populated with the value of EMP-1, EMP-2, EMP-3, so on.

• **Allow Reports:** This checkbox allows you to make the data of the particular object available for reporting.
• **Allow Activities**: This checkbox allows you to associate tasks and events related to the particular object records and the related lists for the activities will be added to the page layout.

• **Track Field History**: This checkbox allows you to track the records of the particular object when it changes the field value of the record. It will keep the old value, the new value, and the user who made the changes.

• **Deployment Status**: This is the option to deploy a particular object. The object will not be visible to any users (except to the users who have a system admin profile) until it is deployed.

When we are creating an object, there are two special options to select as **Object Creation Options**. They are available only when the custom object is first created and not available in modifying the custom object. The following options are available as **Object Creation Options**:

• **Add Notes and Attachments related list to default page layout**: This checkbox allows you to add the notes and attachment-related list to the particular object's page layout.

• **Launch New Custom Tab Wizard after saving this custom object**: This checkbox allows you to launch the new custom tab creation wizard after saving the particular object's definition. In this scenario, we don't select this checkbox. Tab creating will be discussed in *Chapter 3, Building the User Interface*.

After filling in the preceding details, click on the **Save** button. Now, we have created the *Employee* object and now we can create custom fields according to our requirements. Before that, let's get an idea about standard fields that are automatically created at custom object creation. The following are the standard fields that are system generated upon object creation:

**ID**: ID field stores the key, which uniquely locates each record. This unique key is system assigned and you cannot update this value. The ID field exists in every standard and custom object. But this field is not listed in the object definition page. The ID field is equivalent to the primary key of a relational database. Each key (ID value) has the following two versions:

• **15 digit**: This version is used in the Salesforce user interfaces and it contains case sensitive and base-62 strings.

• **18 digit**: This version is used in all API calls and is created by adding a 3-digit suffix to the 15-digit version. 18-character IDs are the case-safe version of Salesforce ID and it is used to safely compare the uniqueness by case-insensitive applications.
Building the Data Model

You cannot delete standard fields from the object.

• **Name:** As we discussed in the object creation, there are two data type options: Text and Auto Number. The following table shows the difference between text and the autonumber data type options in the name field.

<table>
<thead>
<tr>
<th>Text</th>
<th>Auto Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>User assigned value</td>
<td>System generated value</td>
</tr>
<tr>
<td>Not necessarily unique</td>
<td>Unique</td>
</tr>
<tr>
<td>Modifiable</td>
<td>Not modifiable</td>
</tr>
<tr>
<td>Required field</td>
<td>Required field</td>
</tr>
</tbody>
</table>

• **Owner:** This is a reference to the user or group of users (queue). The owner is granted additional privileges. Upon the record creation, the record creator will be assigned as the owner of the particular record but it can be changed later.

• **CreatedBy:** This is a reference to the user who created the particular record.

• **LastModifiedBy:** This is a reference to the user who last modified the particular record.

• **CreatedDate:** This is a DateTime field that contains the date/time that the record was created.

• **ModifiedDate:** This is a DateTime field that contains the date/time when the record was last modified.

The values of **CreatedBy**, **LastModifiedBy**, **CreatedDate**, and **ModifiedDate** are systematically assigned. These fields are also called as audit fields.
You will learn more about the preceding data types in the next couple of sections. The following screenshot shows the Employee object details:

Object definition details and fields—Employee Object

In the same way as with the Employee object creation, create the Holiday Calendar, Leave, Leave Type, Leave Category, and Leave Configuration objects. According to the data type, we will create custom fields for these objects in the next couple of sections.
Creating custom fields

We have successfully created the Employee object with the standard fields. According to the requirements, we need more fields to store various types of data values. In the preceding sections, you have learned that a Salesforce object is analogous to a database table, in the same way as the Salesforce standard/custom field is analogous to a database table column. Normally, a Database Management System (DBMS) has a set of data types for the database table columns. The Force.com platform also provides a set of data types to define custom fields of Salesforce custom objects and selected standard objects. The following two figures show us a few data types in the Force.com platform:

The set of data types of Force.com platform — set 1
As you can see, we have a set of data types to create custom fields for a custom object or a standard object.

Let's explore the data types of the Force.com platform under the seven categories. This categorization will be helpful for you to understand the usage of these data types. The following are the categories:

- Text data types
- Numeric data types
- Calendar data types
- Formatted text data types
Building the Data Model

- Calculation data types
- Limited option data types
- Relationship data types

Text data types
There are five text data types, which are used to create text type fields. The fields of text data types are used to store alphanumeric values. The following table depicts data types with their properties:

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>• This allows the users to enter any combination of letters and numbers</td>
</tr>
<tr>
<td></td>
<td>• It allows maximum 255 characters</td>
</tr>
<tr>
<td></td>
<td>• A single line field is allowed</td>
</tr>
<tr>
<td>Text Area</td>
<td>• This allows the users to enter any combination of letters and numbers</td>
</tr>
<tr>
<td></td>
<td>• It allows maximum 255 characters</td>
</tr>
<tr>
<td></td>
<td>• A multiple line field is allowed</td>
</tr>
<tr>
<td>Text Area (Long)</td>
<td>• This allows the users to enter any combination of letters and numbers</td>
</tr>
<tr>
<td></td>
<td>• It allows maximum 32,768 characters</td>
</tr>
<tr>
<td></td>
<td>• Here, multiple lines are allowed</td>
</tr>
<tr>
<td>Text Area (Rich)</td>
<td>• This allows users to enter formatted text, add images, and links</td>
</tr>
<tr>
<td></td>
<td>• It allows maximum 32,768 characters</td>
</tr>
<tr>
<td></td>
<td>• Here, multiple lines are allowed</td>
</tr>
<tr>
<td>Text (Encrypted)</td>
<td>• It allows users to enter any combination of letters and numbers and stores them in encrypted form</td>
</tr>
</tbody>
</table>

The value of an encrypted field is only visible to users that have the "View Encrypted Data" permission.
Let's see the creation of the text data type field of the `Employee` object. Use the following steps to create a text data type of a custom object:

1. Navigate to Setup | Create | Objects.
2. Click on Employee.
3. In the Custom Fields & Relationships related list, click on New.
4. Select the field data type (for this example, we select the Text data type) and click on Next.
5. Then you will get the following screen to fill in the field details:

```
<table>
<thead>
<tr>
<th>Field Label</th>
<th>First Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>Help Text</td>
<td></td>
</tr>
</tbody>
</table>

Required [ ] Always require a value in this field in order to save a record
Unique [ ] Do not allow duplicate values
- Treat "ABC" and "abc" as duplicate values (case insensitive)
- Treat "ABC" and "abc" as different values (case sensitive)
External ID [ ] Set this field as the unique record identifier from an external system
Default Value

Enter text field type details
```
Following are the details to be filled:

- **Field Label**: This is the label to identify the particular custom field. Enter a label to be used on displays, page layouts, reports, and list views (*First Name*).

- **Length**: This is the maximum length of the particular text field. Refer to the preceding table for length limits of different text data types (255).

- **Field Name**: Normally, this will be autopopulated. The Field Name is an internal reference and is used for integration purposes such as custom links and the API. Be careful when changing the Field Name as it may affect existing integrations.

- **Description**: This is an option field. This will be helpful to understand the purpose of the custom field.

- **Help Text**: This is an optional field. This text displays on detail and edit pages when users hover over the Info icon next to this field.

- **Required**: If you tick this checkbox, this field always requires a value in order to save the record.

- **Unique**: If you tick this checkbox, duplicate values are not allowed here. There are two options to specify the case sensitivity.

- **External ID**: You can use this checkbox to set the particular field as the unique record identifier from an external system.

- **Default Value**: You can specify a default value here by entering any value or using a formula.

Click on **Next**. Then you will get step 3 of the custom field creation wizard with the title, Establish field-level security. For now, accept the defaults. We will discuss more about security of the Force.com platform in *Chapter 4, Designing Apps for Multiple Users and Protecting Data*.

Click on **Next**. Then you will get the last step of the text data type field creation with the Add to page layouts title. You can specify the page layout by adding the newly created custom field to the particular object's page layout.
Click on **Save**. Now, we have created the First Name custom field (Text type) in the Employee object.

<table>
<thead>
<tr>
<th>Field Information</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Field Label</td>
<td>First Name</td>
<td></td>
</tr>
<tr>
<td>Field Name</td>
<td>First_Name</td>
<td></td>
</tr>
<tr>
<td>API Name</td>
<td>First_Name__c</td>
<td></td>
</tr>
<tr>
<td>Data Type</td>
<td>Text</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Help Text</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Created By</td>
<td>Chaml Manusanka, 11/28/2013 7:28 AM</td>
<td></td>
</tr>
<tr>
<td>Modified By</td>
<td>Chaml Manusanka, 11/28/2013 7:28 AM</td>
<td></td>
</tr>
</tbody>
</table>

**General Options**

- **Required** ☑
- **Unique** 
- **Case Sensitive** 
- **External ID** 
- **Default Value** 

**Text Options**

- **Length** 255

**Validation Rules**

Employee custom field—First Name
Building the Data Model

Numeric field data types

There are three numeric data types on the Force.com platform, which are listed in the following table:

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Properties</th>
</tr>
</thead>
</table>
| Number    | • This allows users to enter the number  
• The leading zeroes are removed  
• The decimal places can be defined  
• The maximum length is 18 digits and 18 digits are shared among the length and the number of decimal places |
| Currency  | • This allows users to enter a currency amount  
• It automatically formats the field as a currency amount  
• If you have enabled the multicurrency in your organization, your currency fields are automatically converted into your corporate currency in the view mode  
• The decimal places can be defined  
• The maximum length is 18 digits and the sum of the length and decimal places must be an integer less than or equal to 18 |
| Percent   | • This allows users to enter a percentage number  
• It automatically adds the percentage sign to the number |

Let's see the creation of a numeric data type field on the Leave Configuration object. Use the following steps to create a text data type of a custom object:

1. Navigate to Setup | Create | Objects.
2. Click on Leave Configuration.
3. In the Custom Fields & Relationships related list, click on New.
4. Select the field data type (this time Number) and click on Next.
5. Then you will get the following screen to fill in the field details:

Enter the numeric field type details

Field Label: Number of Leave per Year

Please enter the length of the number and the number of decimal places. For example, a number with a length of 8 and 2 decimal places can accept values up to "12345678.90".

Length: 2

Decimal Places: 0

Field Name: Number_of_Leave_per_Year

Description:

Help Text:

Required: [Always require a value in this field in order to save a record]

Unique: [Do not allow duplicate values]

External ID: [Set this field as the unique record identifier from an external system]

Default Value: Show Formula Editor

Use formula syntax, e.g., Text in double quotes "hello", Number: 25, Percent as decimal: 0.1, Date expression: Today() + 7

Enter the numeric field type details
Building the Data Model

Fill in the following details:

- **Field Label**: This is the label to identify the particular custom field. Enter a label (Number of Leave per Year) to be used on displays, page layouts, reports, and list views.
- **Length**: You can define the maximum length of the particular number field. You can define the length up to 18.
- **Decimal Places**: You can define the decimal places to the particular number field. The sum of the length and decimal places must be an integer less than or equal to 18.
- **Field Name**: Normally, this will be autopopulated. **Field Name** is an internal reference and is used for integration purposes such as custom links and the API. Be careful when changing **Field Name** as it may affect existing integrations.
- **Description**: This is an optional field. This will be helpful to understand the particular custom field.
- **Help Text**: This is an optional field. This text displays on detail and edit pages when users hover over the Info icon next to this field.
- **Required**: If you tick this checkbox, this field always requires a value in order to save the record.
- **Unique**: If you tick this checkbox, duplicate values are not allowed here. There are two options to specify the case sensitivity. This field is not in currency and percent data types.
- **External ID**: You can use this checkbox to set the particular field as the unique record identifier from an external system.
- **Default Value**: You can specify a default value here by entering any value or using a formula.
Click on Next. Then you will get step 3 of the custom field creation wizard with the title, Establish field-level security. For now, accept the defaults. We will discuss more about the security of the Force.com platform in Chapter 4, Designing Apps for Multiple Users and Protecting Data.

Click on Next. Then you will get the last step of the number data type field creation with the Add to page layouts title. You can specify the page layout by adding the newly created custom field to the particular object’s page layout.

Click on Save. Now, we have created the Number of Leave per Year custom field in the Leave Configuration object.

Calendar data types
There are two types of calendar data types in the Force.com platform. The fields of calendar data types are used to store date or date/time values.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Properties</th>
</tr>
</thead>
</table>
| Date        | • This allows users to enter a date value of pick a date from the popup calendar  
              • This type doesn't contain the information about time                  |
| Date/Time   | • This allows users to enter a date and time, or pick a date from the popup calendar  
              • When users click on a date from the popup, that selected date and the current time are entered into the Date/Time field |

Let’s see the creation of a numeric data type field of the Holiday Calendar object. Use the following steps to create a date data type of a custom object:

1. Navigate to Setup | Create | Objects.
2. Click on Holiday Calendar.
3. In the Custom Fields & Relationships related list, click on New.
4. Select the field data type (this time Date) and click on Next.
5. Then you will get the following screen to fill in the field details:

![Image of the screen to fill in field details]

Enter Date field type details

Fill in the **Field Label** as Holiday, and the **Field Name** will be autopopulated. We mark this field as required because this field always needs a date value in order to save a holiday calendar record. In such a scenario, when a user is trying to save the particular record without specifying a required field, an error will be fired.

Click on **Next**. Then you will get step 3 of the custom field creation wizard with the title **Establish field-level security**. For now, accept the defaults. We will discuss more about the security of the Force.com platform in *Chapter 4, Designing Apps for Multiple Users and Protecting Data*.

Click on **Next**. Then you will get the last step of the date type field creation with the title, **Add to page layouts.** You can specify the page layout by adding the newly created custom field to the particular object's page layout.
Chapter 2

Click on Save. Now, we have created the Holiday custom field in the Holiday Calendar object.

Formatted text data types
We can categorize Email, Phone, and URL field types as the formatted text data types.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Properties</th>
</tr>
</thead>
</table>
| Email     | • This allows users to enter an e-mail address  
• This field type has in-built validation to ensure a proper e-mail format  
• If this field is specified for a contact or lead, users can choose the address when clicking on send an e-mail |
| Phone     | • This allows users to enter any phone number  
• It automatically formats it as a phone number |
| URL       | • This allows users to enter any valid website address  
• When users click on the field, the URL will open in a separate browser window  
• This field type has in-built validation to ensure a proper URL format |

Custom e-mail addresses cannot be used for mass e-mails.

Let's see the creation of a formatted data type field of the Employee object. Use the following steps to create an Email data type of a custom object:

1. Navigate to Setup | Create | Objects.
2. Click on Employee.
3. In the Custom Fields & Relationships related list, click on New.
4. Select the field data type (this time Email) and click on Next.
5. Then you will get the following screen to fill in the field details:

![New Custom Field](image)

Enter Email data type details

Fill in the **Field Label** as Employee, and the **Field Name** will be autopopulated.

Click on **Next**. Then you will get step 3 of the custom field creation wizard with the title of **Establish field-level security**. For now, accept the defaults. We will discuss more about the security of the Force.com platform in Chapter 4, *Designing Apps for Multiple Users and Protecting Data*.

Click on **Next**. Then you will get the last step of the **Email** data type field creation with the title of **Add to page layouts**. You can specify the page layout by adding the newly created custom field to the particular object's page layout.
Click on **Save**. Now, we have created the **Email** custom field in the **Employee** object.

### Calculation data types

We can categorize the **Auto Number**, **Formula**, and **Roll-Up Summary** field types as the calculation data types in the Force.com platform.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto Number</td>
<td>• This data type is a system-generated sequence number that uses a display format you define&lt;br&gt;• The number is automatically incremented for each new record</td>
</tr>
<tr>
<td>Formula</td>
<td>• This data type is a read-only field that derives its value from a formula expression you define&lt;br&gt;• The formula field is updated when any of the source fields change</td>
</tr>
<tr>
<td>Roll-Up Summary</td>
<td>• This data type is a read-only field that displays the sum, minimum, or maximum value of a field in a related list or the record count of all records listed in a related list&lt;br&gt;• You cannot create this type of field on an object that is the master of a master-detail relationship. The master-detail relationship will be explained in the <strong>Relationship data types</strong> section</td>
</tr>
</tbody>
</table>

Let's create a formula field called **Age** for the **Employee** object. This field will return the employee's age that is calculated using the birth date and the current date. The following are the steps to create a formula field:

1. Navigate to **Setup | Create | Objects**.
2. Click on **Employee**.
3. In the **Custom Fields & Relationships** related list, click on **New**.
4. Select the field data type (this time **Formula**) and click on **Next**.
5. Then you will get the following screen to fill in the field details:

Choose the output type of the formula filed

Fill in the Field Label as Age, and the Field Name will be autopopulated.

Select the return type of formula field. For this field, we have selected Number as the output type. There are seven options to select the output type, which are as follows:

- Checkbox
- Currency
- Date
- Date/Time

- Number
- Percent
- Text
• Number
• Percent
• Text

Define the decimal places for the field. (For this example, we don't need decimal places. So we set the value to 0.)

Click on Next and you will get the page to enter the formula you need to execute in this field.

Enter formula of the formula field
Building the Data Model

You can enter your formula in the formula editor. There are **Simple Formula** and **Advanced Formula** editors. You can use them according to your requirements. We have used the advanced editor in this field creation. The Force.com platform provides a facility to check your formula before you save.

> A formula is more like an equation that performs a calculation at runtime. Depending on the context, the formula can use data and operations to calculate the new value of some other type. You can build your formula using the set of in-built functions. There are four types of functions. They are, **Date & Time**, **Logical**, **Math**, and **Text**.

Click on **Next**. Then you will get step 3 of the custom field creation wizard with the title, Establish field-level security. For now, accept the defaults. We will discuss more about the security of the Force.com platform in *Chapter 4, Designing Apps for Multiple Users and Protecting Data*.

Click on **Next**. Then you will get the last step of the formula field creation with the title, **Add to page layouts**. You can specify the page layout by adding the newly created custom field to the particular object's page layout.

Click on **Save**. Now, we have created the **Age** formula field in the **Employee** object. When you are viewing the particular employee record, the age will be calculated and viewed on the page.

**Limited option data types**

We can identify **Checkbox**, **Picklist**, and **Picklist (Multi-Select)** as limited option data types, which are explained in the following table:

<table>
<thead>
<tr>
<th>Date Type</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>PickList</td>
<td>• This allows users to select a value from a predefined list</td>
</tr>
<tr>
<td></td>
<td>• A single value can be selected</td>
</tr>
<tr>
<td></td>
<td>• A maximum of 1,000 picklist values can be accommodated in a Picklist field</td>
</tr>
<tr>
<td></td>
<td>• This type of field can be used to prevent spelling mistakes and grammatical errors by users</td>
</tr>
</tbody>
</table>
You can define dependent picklists in the Force.com platform. A dependent picklist has a controlling picklist. The controlling picklist controls the whole set of values in the dependent picklist. In other words, the selected values of the controlling picklist can change the values of the dependent picklist. There are two types of picklists, which are as follows:

- Standard picklist: This can only be the controlling picklist
- Custom picklist: This can be the controlling picklist as well as the dependent picklist

Let’s create a Picklist field called Year for the Holiday Calendar object. This field contains the year values. The following are the steps to create a picklist field:

1. Navigate to Setup | Create | Objects.
2. Click on Holiday Calendar.
3. In the Custom Fields & Relationships related list, click on New.
4. Select the field data type (this time Picklist) and click on Next.
5. Then you will get the following screen to fill in the field details:

Enter the details and values of the picklist field

Fill in the **Field Label** as **Year**, and the **Field Name** will be autopopulated.

Enter the year values to the picklist values' text area. According to the requirements, you can select that first value as the default value.

Click on **Next**. Then you will get step 3 of the custom field creation wizard with the title, **Establish field-level security**. For now, accept the defaults. We will discuss more about security of the Force.com platform in *Chapter 4, Designing Apps for Multiple Users and Protecting Data*.

Click on **Next**. Then you will get the last step of the picklist data type field creation with the title, **Add to page layouts**. You can specify the page layout by adding the newly created custom field to the particular object's page layout.

Click on **Save**. Now, we have created the **Year** picklist field in the **Holiday Calendar** object.
Relationship data types

In Chapter 1, Getting Started with Force.com, we have already discussed the ER diagram of the leave management application. There are a few relationships in the leave management app. The relationships play an important role in the Force.com platform. The Force.com platform provides two relationship data types to link two objects, which are listed in the following table:

<table>
<thead>
<tr>
<th>Data type</th>
<th>Properties</th>
</tr>
</thead>
</table>
| Lookup Relationship| • This creates a relationship that links one object to another  
• The relationship field allows users to click on a lookup icon to select a value from a popup list  
• The other object is the source of the values in the list  
• The lookup field value can be mandatory or optional. It is up to the developer to decide according to the design  
• The relationship dependency between two objects can be defined as **Clear the value of this field. You can't choose this option if you make this field required** or **Don't allow deletion of the lookup record that's part of a lookup relationship**. Therefore, updating and deleting the child record depends on the particular relationship field definition  
• Two objects can have their own owners and sharing rules  
• There are maximum of 25 lookup relationships per child |
| Master-Detail Relationship | • This creates a special type of parent-child relationship between two objects  
• The relationship field allows users to click on a lookup icon to select a value from a popup list  
• The other object is the source of the values in the list  
• The relationship field is required on all detail records  
• The ownership and sharing of detail records is determined by the master record  
• When a user deletes the master record, all the detail records are deleted (cascade deleting)  
• You can create roll-up summary fields on the master record to summarize the detail records  
• A detail object can have a maximum of two master objects and both masters are required in the detail record |
Both relationship types look almost the same on a page layout; both include a lookup field for the one side and a related list for the many sides. You will learn more about page layout in Chapter 3, Building the User Interface.

Special relationship types
Besides the lookup and master-detail relationships, there are two special relationship types in the Force.com platform. These types are created using master-detail and lookup relationships, which are as follows:

<table>
<thead>
<tr>
<th>Self-Relationship</th>
<th>Many-to-Many Relationship</th>
</tr>
</thead>
<tbody>
<tr>
<td>• This involves a lookup relationship to the same object. For example, the Employee object has a lookup field called Manager and Manager is an employee. Therefore, the Employee object has a lookup relationship to the Employee object.</td>
<td>• This allows for the relationship of two objects in a many-to-many fashion.</td>
</tr>
<tr>
<td>• The user object has a special type of lookup relationship called Hierarchical Relationship.</td>
<td>• When modeling a many-to-many relationship, you use a junction object to connect the two objects you want to relate to each other.</td>
</tr>
<tr>
<td>• For example, a hierarchical relationship allows developers to create a Manager field on the User object to relate to another user.</td>
<td>• A junction object is a custom object with two relationships.</td>
</tr>
<tr>
<td></td>
<td>• For example, in our sample, the leave management app, a Leave Category can have many leave types and a Leave Type can have many leave categories. Therefore, we have created the junction object called Leave Configuration to apply the many-to-many relationship between Leave Type and Leave Category.</td>
</tr>
</tbody>
</table>

When creating a junction object, consider the following:

- Name the object with a label that indicates its purpose
- Use the autonumber data type to the Name field
The junction object takes the initially created master as the primary master and uses the look and feel of the primary master. It also inherits the sharing model from the primary master. If any master record is deleted, the junction record is deleted.

Creating a lookup relationship field

Let's create a lookup relationship field called Leave Category for the Employee object. The following are the steps to create a lookup relationship field:

1. Navigate to Setup | Create | Objects.
2. Click on Employee.
3. In the Custom Fields & Relationships related list, click on New.
4. Select the field data type (this time Lookup Relationship) and click on Next.
5. Then you will see the page for step 2 (choose the related object). You have to select the other object to which the Employee object is related. Click on Next.
6. Then you will get the following screen to fill in the field details:

   ![Field details](image)

Enter the details of the lookup field
Fill in the **Field Label** as *Leave Category*, and the **Field Name** will be autopopulated.

- **Child Relationship Name**: The Child Relationship Name is an internal reference and is used for integration purposes. Be careful when changing the Child Relationship Name as it may affect existing integrations. This is an autopopulated field, but you can change/modify it.

- **Required**: If you tick this checkbox, this field always requires a value in order to save the record. This option was released in Summer '12. When the lookup field is optional, you can choose one of the following options:
  - **Clear the value of this field**: If the referenced lookup record is deleted, this default option clears the value of the dependent lookup field. You can't choose this option if you make this field required.
  - **Don't allow deletion of the lookup record that's part of a lookup relationship**: This option is used to restrict the deletion of the dependent lookup records by deleting the referenced lookup record.

- **Lookup Filter**: Optionally, you can create a filter to limit the records available to users in the lookup field. It helps to improve the productivity and the data quality of the lookup field. It can be made mandatory and fields can be compared to static values or to other fields. You can define informational messages to the lookup window.

Click on **Next**. Then you will get step 4 of the custom field creation wizard with the title, **Establish field-level security**. For now, accept the defaults. We will discuss more about security of the Force.com platform in *Chapter 4, Designing Apps for Multiple Users and Protecting Data*.

Click on **Next**. Then you will get step 5 of the lookup relationship field creation with the title, **Add to page layouts**. You can specify the page layout by adding the newly created custom field to the particular object's page layout.

Click on **Next**. Then you will get the last step called Add custom-related lists. Here, you can specify the title that the related list will have in all of the layouts associated with the parent. Also, you can select the page layouts that should include this field. The field will not appear on any pages if you do not select a layout in step 5.

Click on **Save**. Now, we have created the *Leave Category* lookup field for the Employee object.
You can convert relationships in the Force.com platform as follows:

- **Master-Detail Relationship** to a lookup relationship only if the master object doesn't have any roll-up summary field
- **Lookup Relationship** to a master-detail relationship only if the lookup field in all records contains a value

Creating a master-detail relationship field

Let's create a master-detail relationship field called Leave Type for the Leave Configuration object. When you are creating a master-detail relationship field, you can follow the same steps of creating a lookup relationship field. The only difference is the content of step 3. The following screenshot shows step 3 of creating a master-detail field:

![Step 3 of creating a master-detail relationship field](image)

Enter the details of master-detail relationship field
If the Master-Detail relationship is between the standard and the custom object, the custom object will always be the detail record.

A master-detail can be multiple levels deep, parent-child-grand-children.

If the master is a standard object: two multiple levels deep.

If the master is a custom object: three multiple levels deep.

Fill in the Field Label as Leave Type, and the Field Name will be autopopulated.

- **Child Relationship Name**: The Child Relationship Name is an internal reference and is used for integration purposes. Be careful when changing the Child Relationship Name as it may affect existing integrations. This is an autopopulated field, but you can change and modify it.

- **Sharing Setting**: This setting is to select the minimum access level required on the Master record to create, edit, or delete related detail records. There are two options to select the minimum access level.
  - **Read Only**: This allows users with at least read access to the master record to create, edit, or delete related detail records.
  - **Read/Write**: This allows users with at least read/write access to the master record to create, edit, or delete related detail records.

- **Allow reparenting**: If you tick this checkbox, the child records can be reparented to other parent records after they are created.

- **Lookup Filter**: Alternatively, you can create a filter to limit the records available to users in the lookup field. It helps to improve the productivity and the data quality of the lookup field. It can be made mandatory and the fields can be compared to static values or to other fields. You can define informational messages to the lookup window.

Click **Next**. Then you will get the 4th step of the custom field creation wizard with the title of **Establish field-level security for reference field**. For now, accept the defaults. We will discuss more about the security of the Force.com platform in Chapter 4, Designing Apps for Multiple Users and Protecting Data.

Click on **Next**. Then you will get step 5 of the master-detail relationship field creation with the title, **Add to page layouts**. You can specify the page layout by adding the newly created custom field to the particular object's page layout.

Click on **Next**. Then you will get the last step called **Add custom related lists**. Here, you can specify the title that the related list will have in all of the layouts associated with the parent. You can also select the page layouts that should include this field. The field will not appear on any pages if you do not select a layout in step 5.
Click on **Save**. Now, we have created the **Leave Type** master-detail field for the **Leave Configuration** object.

Now, we have finished creating custom objects and custom fields. You can complete the data model (according to the ER diagram in *Chapter 1, Getting Started with Force.com*) of the leave management application using the knowledge we gain in this chapter. Use the following details of custom fields and particular data type to complete the data model of the leave management application:

1. **Employee Object**

<table>
<thead>
<tr>
<th>Field Label</th>
<th>API Name</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>Age__c</td>
<td>Formula (Number)</td>
</tr>
<tr>
<td>Birthday</td>
<td>Birthday__c</td>
<td>Date</td>
</tr>
<tr>
<td>Email</td>
<td>Email__c</td>
<td>Email</td>
</tr>
<tr>
<td>First Name</td>
<td>First_Name__c</td>
<td>Text(255)</td>
</tr>
<tr>
<td>IsActive</td>
<td>IsActive__c</td>
<td>Checkbox</td>
</tr>
<tr>
<td>Last Name</td>
<td>Last_Name__c</td>
<td>Text(255)</td>
</tr>
<tr>
<td>Leave Category</td>
<td>Leave_Category__c</td>
<td>Lookup(Leave Category)</td>
</tr>
<tr>
<td>Manager</td>
<td>Manager__c</td>
<td>Lookup(Employee)</td>
</tr>
<tr>
<td>Social Security Number</td>
<td>Social_Security_Number__c</td>
<td>Text(10) (External ID)</td>
</tr>
<tr>
<td>User</td>
<td>User__c</td>
<td>Lookup(User)</td>
</tr>
</tbody>
</table>

2. **Holiday Calendar**

<table>
<thead>
<tr>
<th>Field Label</th>
<th>API Name</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Holiday</td>
<td>Holiday__c</td>
<td>Date</td>
</tr>
<tr>
<td>Holiday Reason</td>
<td>Holiday_Reason__c</td>
<td>Text Area(255)</td>
</tr>
<tr>
<td>Year</td>
<td>Year__c</td>
<td>Picklist</td>
</tr>
</tbody>
</table>

3. **Leave Category**

<table>
<thead>
<tr>
<th>Field Label</th>
<th>API Name</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsActive</td>
<td>IsActive__c</td>
<td>Checkbox</td>
</tr>
</tbody>
</table>

4. **Leave Type**

<table>
<thead>
<tr>
<th>Field Label</th>
<th>API Name</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsActive</td>
<td>IsActive__c</td>
<td>Checkbox</td>
</tr>
</tbody>
</table>
### Building the Data Model

#### 5. Leave Configuration

<table>
<thead>
<tr>
<th>Field Label</th>
<th>API Name</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsActive</td>
<td>IsActive__c</td>
<td>Checkbox</td>
</tr>
<tr>
<td>Leave Category</td>
<td>Leave_Category__c</td>
<td>Master-Detail(Leave Category)</td>
</tr>
<tr>
<td>Leave Type</td>
<td>Leave_Type__c</td>
<td>Master-Detail(Leave Type)</td>
</tr>
<tr>
<td>Number of Leave per Year</td>
<td>Number_of_Leave_per_Year__c</td>
<td>Number(2, 0)</td>
</tr>
</tbody>
</table>

#### 6. Leave

<table>
<thead>
<tr>
<th>Field Label</th>
<th>API Name</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee</td>
<td>Employee__c</td>
<td>Master-Detail(Employee)</td>
</tr>
<tr>
<td>From Date</td>
<td>From_Date__c</td>
<td>Date</td>
</tr>
<tr>
<td>Leave Type</td>
<td>Leave_Type__c</td>
<td>Lookup(Leave Type)</td>
</tr>
<tr>
<td>Number of Days</td>
<td>Number_of_Days__c</td>
<td>Number(3, 1)</td>
</tr>
<tr>
<td>Status</td>
<td>Status__c</td>
<td>Picklist</td>
</tr>
<tr>
<td>To Date</td>
<td>To_Date__c</td>
<td>Date</td>
</tr>
</tbody>
</table>

### Summary

In this chapter, we got familiar with the data model of the Force.com platform. You have learned about standard and custom objects and have created custom objects according to our sample leave management application. We have discussed various kinds of data types available in the Force.com platform and you learned to create those various data type custom fields. Finally, you learnt the object relationship that is an important topic for building data models on the Force.com platform.
Where to buy this book

You can buy Learning Force.com Application Development from the Packt Publishing website.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.