



# Advanced Developer

Kentico 9 Solution Book

# Contents

Kentico API.....	3
Event Handlers .....	5
Exercise 1 .....	5
Exercise 2 .....	6
K# Macros .....	8

# Kentico API

Add a **Send email** button to the *MyPage.aspx* file, and create an on-click handler for the button.

```
<asp:Button ID="btnSendEmail" runat="server" Text="Send email" OnClick="btnSendEmail_Click" />
```

Type the following source code to the **MyPage.aspx.cs** class:

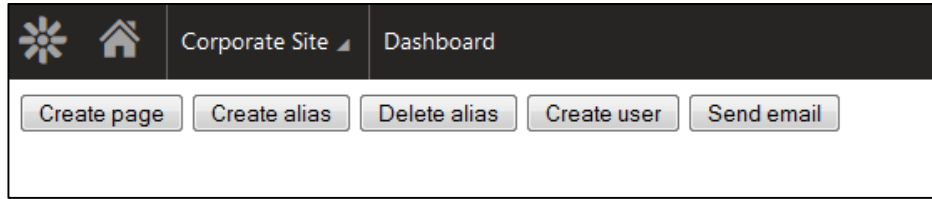
```
using System;
using System.Data;
using CMS.DataEngine; // for DataQuery and its operators
using CMS.EmailEngine; // for email objects
using CMS.Helpers; // for validation helper
protected void btnSendEmail_Click(object sender, EventArgs e)
{
    // Gets John's email address from the database
    UserInfo userInfo = UserInfoProvider.GetUsers()
        .Columns("Email")
        .Where("UserName", QueryOperator.Equals, "John")
        .FirstObject;

    // Check if the userInfo is not null and therefore found
    if (userInfo != null)
    {
        string userEmail = userInfo.Email;

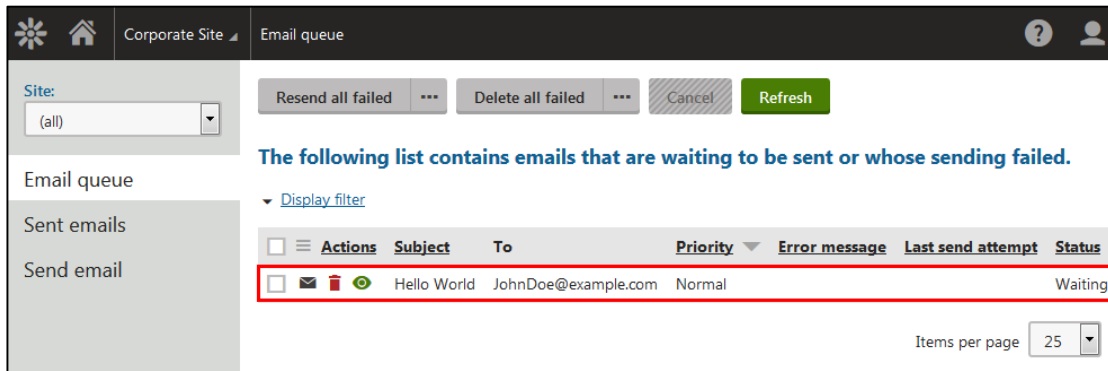
        if (ValidationHelper.IsEmail(userEmail))
        {
            // Create an EmailMessage instance and set its required properties
            EmailMessage newEmail = new EmailMessage()
            {
                From = "administrator@example.com",
                Recipients = userEmail,
                Subject = "Hello World",
                PlainTextBody = "Hello World",
            };

            // Send the email
            EmailSender.SendEmail(newEmail);
        }
    }
}
```

The system now adds the button to the ASPX page you created before. After clicking the **Send email** button, the system will send an email to John.



Log in to the administration interface, open the **My app** application, and click the **Send email** button. Then, open the **Email queue** application to see your email in the queue.



# Event Handlers

## Exercise 1

Type the following code to the **MyHandler.cs** class:

```
using System;
using CMS.Base;
using CMS.SiteProvider; // for current site context
using CMS.DataEngine; // for ObjectEventArgs needed in the event handler
using CMS.Membership; // user objects
using CMS.EmailEngine; // email objects

[MyHandlerEvents]
public partial class CMSModuleLoader
{
    private class MyHandlerEventsAttribute : CMSLoaderAttribute
    {
        public override void Init()
        {
            // Assign the handler to the Insert event for user objects
            UserInfo.TYPEINFO.Events.Insert.After += User_Insert_After;
        }

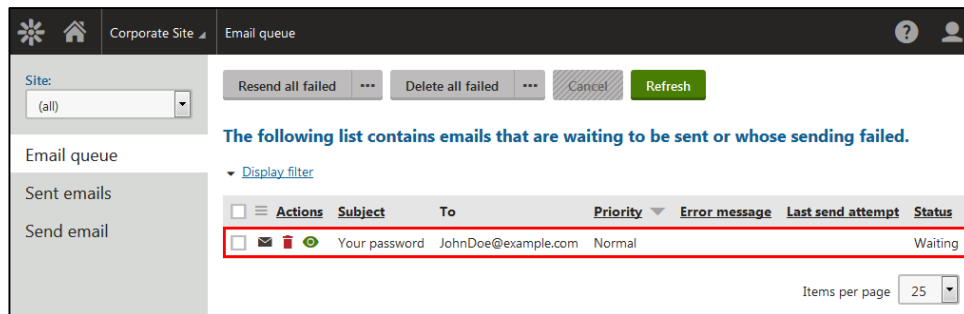
        private void User_Insert_After(object sender, ObjectEventArgs e)
        {
            // Get the new user object
            UserInfo user = (UserInfo)e.Object;

            // Generate the new password and assign it to the user
            // (Set the site name because of site password requirements)
            string password = UserInfoProvider.GenerateNewPassword(SiteContext.CurrentSiteName);
            UserInfoProvider.SetPassword(user, password);

            // Create and send an email with the password
            EmailMessage newEmail = new EmailMessage();
            newEmail.From = "administrator@example.com";
            newEmail.Recipients = user.Email;
            newEmail.Subject = "Your password";
            newEmail.PlainTextBody = "Your password is: " + password;
            EmailSender.SendEmail(newEmail);
        }
    }
}
```

Now, when a user is created, the system automatically generates their password and sends them an email with the password.

Log into the administration interface and use the **My app** application to create a user, or use the **Users** application to create a user manually. (If you use the **My app** application, don't forget to delete John in the **Users** application first.) Then, open the **Email queue** application to see the email.



## Exercise 2

Type the following code to the **MyHandler.cs** class:

```
using System;
using CMS.Base;
using CMS.DocumentEngine;
using CMS.SiteProvider;
using CMS.Membership;
using CMS.DataEngine;
using CMS.EmailEngine;

[MyHandlerEvents]
public partial class CMSModuleLoader
{
    private class MyHandlerEventsAttribute : CMSLoaderAttribute
    {
        {
            public override void Init()
            {
                // Assign the handler to the Authenticate event
                SecurityEvents.Authenticate.Execute += Security_Authenticate_Execute;
            }

            private void Security_Authenticate_Execute(object sender,
                AuthenticationEventArgs e)
            {
                // Start only when the original authentication failed
            }
        }
    }
}
```

```
if (e.User == null)
{
    // Find the username in the database
    var existingUser = UserInfoProvider.GetUserInfo(e.UserName);

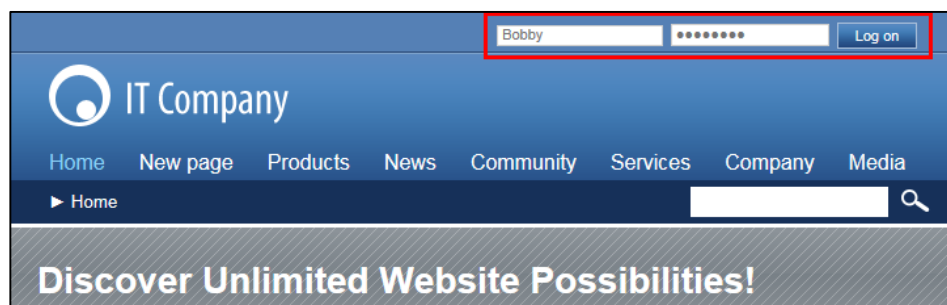
    // Check if the user exists, continue only if not
    if (existingUser == null){
        // Create a new user;
        UserInfo newUser = new UserInfo()
        {
            UserName = e.UserName,
            Enabled = true
        };

        // set password for the user (we need to use external method as Password hashed and only a read-
        only property)
        UserInfoProvider.SetPassword(newUser.UserName, e.Password, false); // we don't want to save user

        // Assign newly created user to e.User (user will be logged and created in DB as well)
        e.User = newUser;
    }
}
}
```

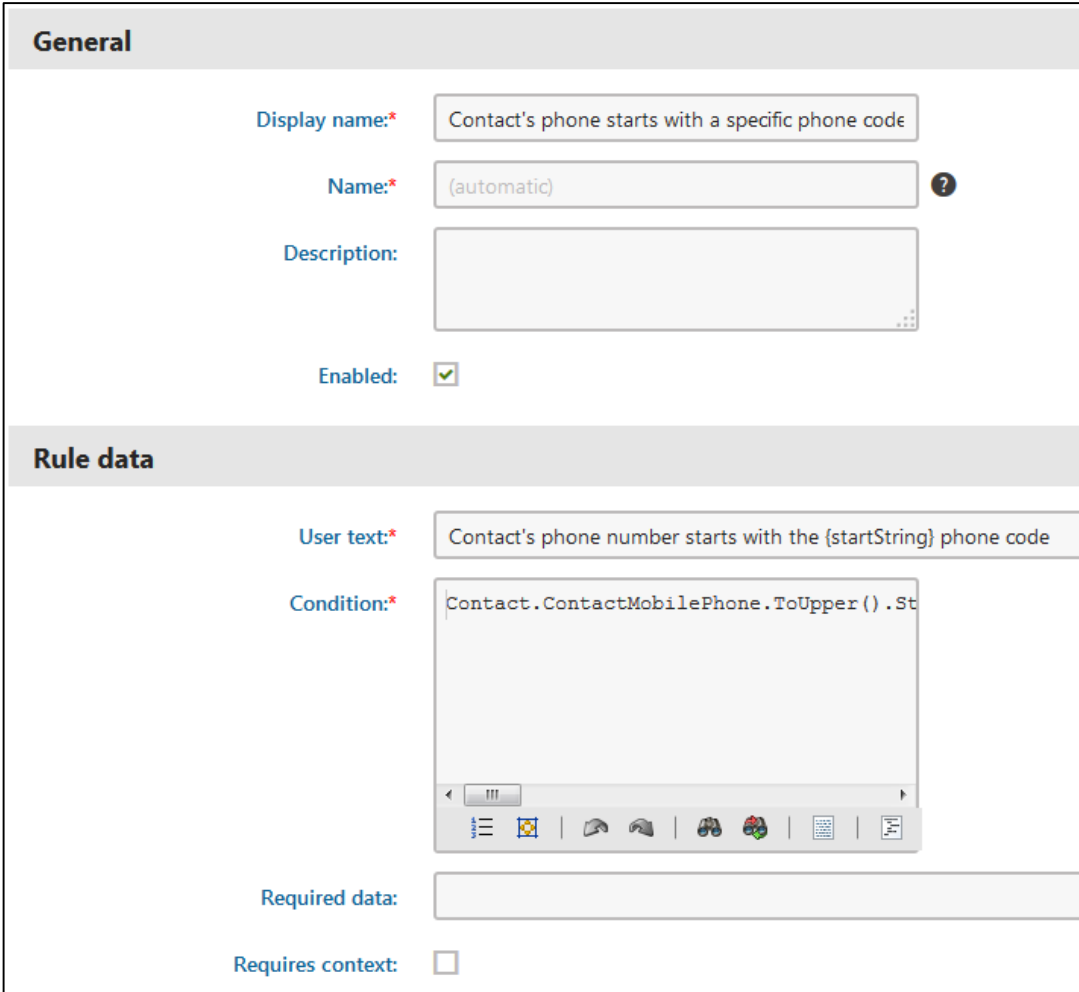
When a user tries to log in and they don't have a user account, the system automatically creates a new user account for them with the typed password and logs them into the new account.

Go to the live site and log in with a non-existing user (for example, Bobby). The system will log Bobby in, even though the user doesn't exist. Then, log into the administration interface as Administrator and go to the **Users** application where you can see the new user.



# K# Macros

1. On the **Configuration** → **Macro rules** tab in the **Contact Management** application, click **New macro rule**.
2. Enter the following properties:
  - **Display name:** Contact's phone number starts with a specific phone code
  - **User text:** Contact's phone number starts with the {startString} phone code
  - **Condition:**
    - Contact.ContactMobilePhone.StartsWith("{startString}")
    - || Contact.ContactHomePhone.StartsWith("{startString}")
    - || Contact.ContactBusinessPhone.StartsWith("{startString}")
3. **Save** the rule.



**General**

**Display name:** Contact's phone starts with a specific phone code

**Name:** (automatic) ?

**Description:**

**Enabled:**

**Rule data**

**User text:** Contact's phone number starts with the {startString} phone code

**Condition:**  
`Contact.ContactMobilePhone.ToUpper().St`

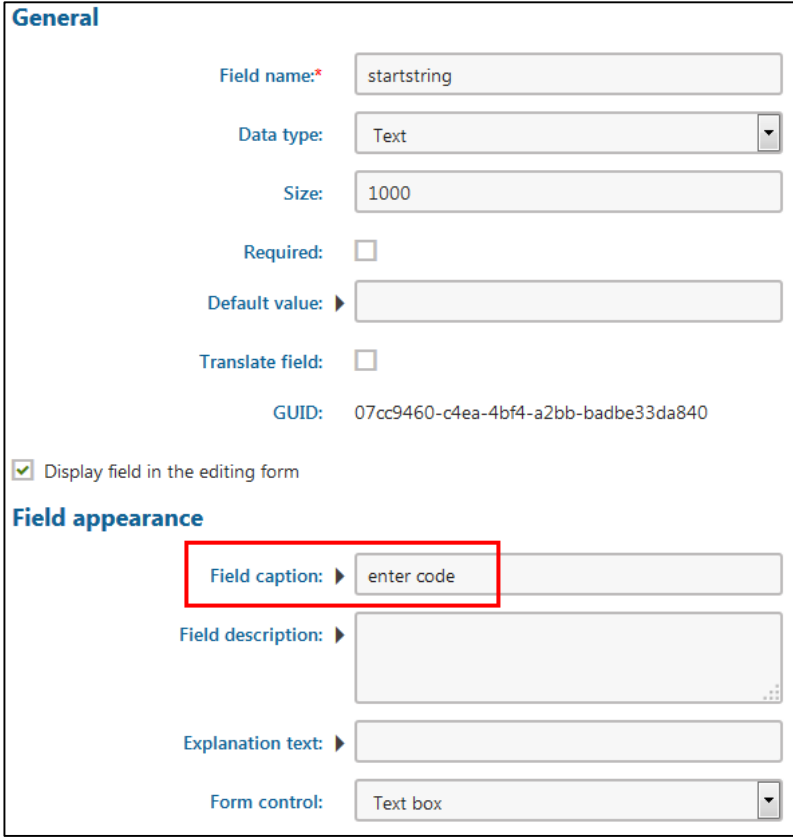
**Required data:**

**Requires context:**

4. Switch to the **Parameters** tab.



5. Change the **Field caption** to *enter code* and click **Save**.



**General**

Field name\*: startstring

Data type: Text

Size: 1000

Required:

Default value: ▶

Translate field:

GUID: 07cc9460-c4ea-4bf4-a2bb-badbe33da840

Display field in the editing form

**Field appearance**

Field caption: ▶ enter code

Field description: ▶

Explanation text: ▶

Form control: Text box

You can now add macro rules to personas based on contacts' phone codes.

Log into the administration interface, and in the **Contacts** tab of the **Contact Management** application, create a **New contact** with a specific phone number. Then, in the **Personas** application, create a **New persona**. After saving the persona, switch to the **Rules** tab and **create a rule** with a macro rule type and enter the same phone code as for the user before. Select your macro with phone codes and **save** it. Then, switch back to the **Rules** tab and click **Recalculate** to recalculate the contacts. Switch to the **Contacts** tab to see the contact assigned to the persona.

# Thank you for participating in our training!

We value the time you have spent with us and want to hear your opinions and thoughts!

We'd greatly appreciate it if you could share your opinion with us by submitting this feedback form:

<http://www.kentico.com/Training/Training-Feedback>