```
public abstract class Person
    {
        //Class variable
        private static int counter = 0;

        //Instance variables
        protected int id = 0;
        protected string name;
        protected string password;

        //Constructor
        public Person(string name, string password)
        {
            id = counter;
            this.name = name;
            this.password = password;
            counter++;
        }

        //Id property
        public int Id
        {
            get { return id; }
        }

        //name property
        public string Name
        {
            get { return name; }
        }

        //password property
        public string Password
        {
            get { return password; }
            set { password = value; }
        }

        //ToString method
        public override string ToString()
        {
            return "The person with the id: " + id + " is named " + name + " and has
            the password: " + password;
        }
    }
```

# Magical Show

Make a Person class.

You decide which information is needed, but id, name and password must be present.

The Person class must be abstract.

The Person class is abstract, as we can see in the first line.

The instance variables id, name and password.

The id is autoincremented, which means that every person, whenever they a staff or magician, has their own unique id.

# Magical Show

Make two subclasses Magician and Staff which inherit Person.

You decide which information is needed, but staff needs a salary.

Staff is based on the Person class and has the specific instance variables, occupation and salary.

A Staff person must have an occupation assigned when it is created, the salary can only be set when the person is created and changes their salary.

```
public class Staff : Person
    {
        //Instance variables
        private string occupation;
        private int salary;

        //Constructor
        public Staff(string name, string password, string occupation, int salary) :
        base(name, password)
        {
            this.occupation = occupation;
            this.salary = salary;
        }

        //occupation property
        public string Occupation
        {
            get { return occupation; }
            set { occupation = value; }
        }

        //salary property
        public int Salary
        {
            get { return salary; }
            set { salary = value; }
        }


        //ToString method
        public override string ToString()
        {
            return base.ToString() + ". " + name + "s occupation is " + occupation + "
            and earns " + salary;
        }
    }
```

```
public class Magician : Person
    {
        string tricks;
        //Instance variable
        private ArrayList favouriteTrick;

        //Constructor

        public Magician(string name, string password) : base(name, password)
        {
            favouriteTrick = new ArrayList();
        }

        public Magician(string name, string password, string trick) : base(name,
        password)
        {
            favouriteTrick = new ArrayList();
            favouriteTrick.Add(trick);
        }

        //No property

        //Add Trick method
        public void AddfavouriteTrick(string item)
        {
            favouriteTrick.Add(item);
        }

        //Tostring
        public override string ToString()
        {

            return base.ToString() + " and has the favourite trick: " +
            getInformation() + ". Lenght: " + favouriteTrick.Count;

        }

        //getInformation method
        public override string getInformation()
        {
            return Name + " has the Id of " + Id + ". List of tricks: " +
            getFavouritTricks();
        }

        //get Favourite Tricks method
        public string getFavouritTricks()
        {
            tricks = "";
            for (int i = 0; i < favouriteTrick.Count; i++)
            {
                tricks += favouriteTrick[i] + ", ";
            }
            return tricks;
        }
    }
```

# Magical Show

Magicians need a collection of favourite tricks and must have an AddFavouriteTrick method.

It must be possible to create magicians with or without a single favourite trick.

Magician is based on the person class and have the specific instance variables; favouriteTrick.

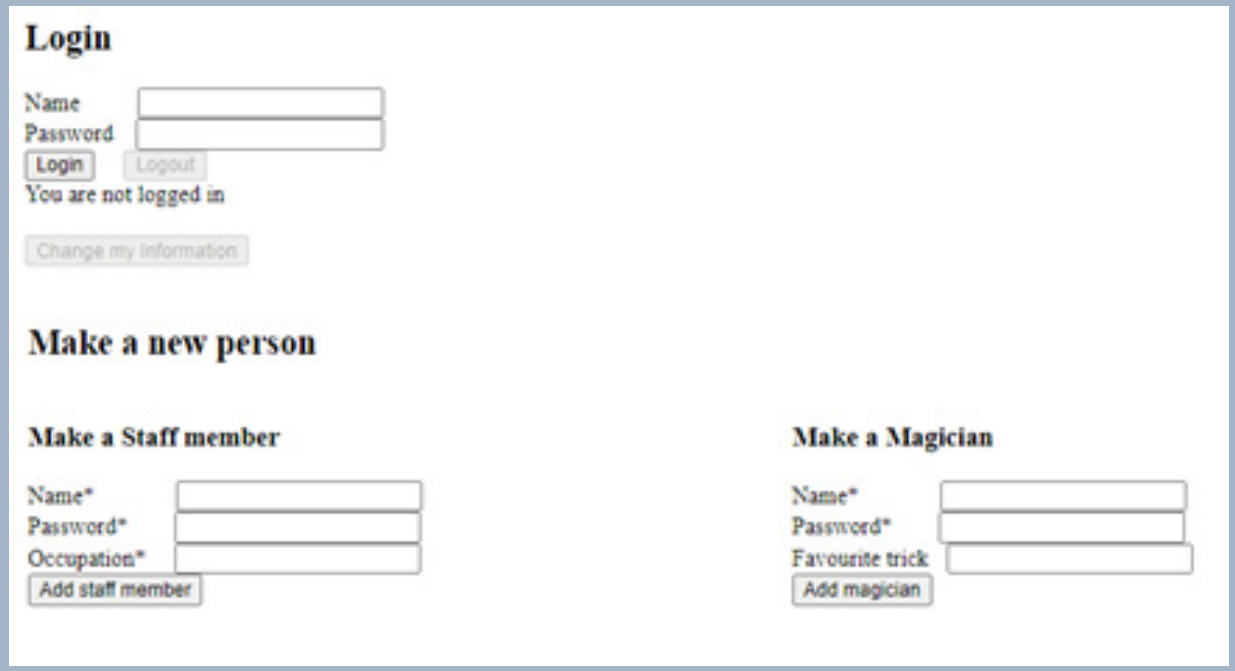
The favouriteTrick is a collection, and when we create a magician, we can choose to either fill in the first trick or no.

We also have the option to add tricks later on, using AddfavouriteTrick function.

# Magical Show

Make web pages where magicians and staff can enlist by entering the necessary data.
Persons must be created as objects and placed in an ArrayList.

On the index page, the first thing you see is the forms where you either login or enlist new staff members or magicians.



```
if (Application["personcollection"] == null)
            {
                personArrayList = new ArrayList();
                Magician magician1 = new Magician("Wizardus", "Hey321");
                Staff staff2 = new Staff("Linsey", "BeautiIsAll", "Host", 635);
                personArrayList.Add(magician1);
                personArrayList.Add(staff2);
                Application["personcollection"] = personArrayList;
            }
            personArrayList = (ArrayList)Application["personcollection"];
```

```
    protected void ButtonAddStaff_Click(object sender, EventArgs e)
    {
        Staff s = new Staff(TextBoxStaffName.Text, TextBoxStaffPassword.Text,
        TextBoxStaffOccupation.Text, 0);
        personArrayList.Add(s);
        TextBoxStaffName.Text = "";
        TextBoxStaffPassword.Text = "";
        TextBoxStaffOccupation.Text = "";
        SetUpPage();
    }

    protected void ButtonAddMagician_Click(object sender, EventArgs e)
    {
        if(TextBoxMagicianTrick.Text == "" || TextBoxMagicianTrick.Text == null)
        {
            Magician m = new Magician(TextBoxMagicianName.Text,
            TextBoxMagicianPassword.Text);
            personArrayList.Add(m);
            TextBoxMagicianName.Text = "";
            TextBoxMagicianPassword.Text = "";
            SetUpPage();
        }
        else
        {
            Magician m = new Magician(TextBoxMagicianName.Text,
            TextBoxMagicianPassword.Text, TextBoxMagicianTrick.Text);
            personArrayList.Add(m);
            TextBoxMagicianName.Text = "";
            TextBoxMagicianPassword.Text = "";
            TextBoxMagicianTrick.Text = "";
            SetUpPage();
        }
    }
```

# Magical Show

I had defined a ArrayList personArrayList; which is the array that will contain all the persons.

Inside the Page_Load function it checks if Application["personcollection"] exist, if not it will create the personArrayList put in two people manually and then set Application["personcollection"] equals to personArrayList.

Now we are able to se the arraylist everywhere in the application.

Further down in the code two click function is inserting either a Staff or Magician person inside the personArrayList, so we can put in people from the webpage.

# Magical Show

Make a simple login page where existing magicians and staff can login and logout using your own code.

After some validation the ButtonLogin_Click function checks if the both the name an password can be found in the same person object. If all is good it will set userid equals to the person's Id

Now the "Change my information"-button is enabled , and we can see and change the persons information on the page Account.aspx


```
protected void ButtonLogin_Click(object sender, EventArgs e)
{
    if (TextBoxName.Text != "")
    {
        if (TextBoxPassword.Text != "")
        {
            LabelLoginInfo.Text = "";
            found = false;
            for (int i = 0; i < personArrayList.Count && !found; i++)
            {
                try
                {
                       if (((Staff)personArrayList[i]).Name == TextBoxName.Text &&
                       ((Staff)personArrayList[i]).Password == TextBoxPassword.Text)
                    {
                        found = true;
                        TextBoxName.Text = "";
                        LabelLoginFail.Text = "";
                        userid = ((Staff)personArrayList[i]).Id;
                        SetUpPage();
                    }
                    else
                    {
                        LabelLoginFail.Text = "Wrong name or password";
                    }
                }
                catch
                {
                       if (((Magician)personArrayList[i]).Name == TextBoxName.Text &&
                       ((Magician)personArrayList[i]).Password == TextBoxPassword.Text)
                    {
                        found = true;
                        TextBoxName.Text = "";
                        LabelLoginFail.Text = "";
                        userid = ((Magician)personArrayList[i]).Id;
                        SetUpPage();
                    }
                    else
                    {
                        LabelLoginFail.Text = "Wrong name or password";
                    }
                }
            }
        }
        else
        {
            LabelLoginFail.Text = "You need to write the password";
        }
    }
    else
    {
        LabelLoginFail.Text = "You need to write a name";
    }
    Session["Login"] = userid;
    SetUpPage();
}
```

```
private void SetUpPage()
        {
            if(userid < 0) { Response.Redirect("LoginFailed.aspx"); }
            found = false;
            localArrayList = (ArrayList)Application["personcollection"];
            for (int i = 0; i < localArrayList.Count && !found; i++)
            {
                try
                {
                        if (((Staff)localArrayList[i]).Id ==
                        Convert.ToInt32(Session["Login"]))
                    {
                        UserIsStaff.Style["display"] = "block";
                        UserIsMagician.Style["display"] = "none";
                        ShowStaffMebers();

                        TextBoxChangeStaffName.Text = ((Staff)localArrayList[i]).Name;
                        TextBoxChangeStaffPassword.Attributes.Add("placeholder",
                        ((Staff)localArrayList[i]).Password);
                        TextBoxChangeOccupation.Attributes.Add("placeholder",
                        ((Staff)localArrayList[i]).Occupation);
                        TextBoxChangeSalery.Attributes.Add("placeholder",
                        ((Staff)localArrayList[i]).Salary.ToString());
                        password = ((Staff)localArrayList[i]).Password;
                        Occupation = ((Staff)localArrayList[i]).Occupation;
                        salery = ((Staff)localArrayList[i]).Salary;

                        found = true;
                    }
                }
                catch
                {
                    if (((Magician)localArrayList[i]).Id ==
                    Convert.ToInt32(Session["Login"]))
                    {
                        UserIsStaff.Style["display"] = "none";
                        UserIsMagician.Style["display"] = "block";
                        ShowMagicianMembers();

                        TextBoxChangeMagiName.Text =
                        ((Magician)localArrayList[i]).Name;
                        TextBoxChangeMagiPassword.Attributes.Add("placeholder",
                        ((Magician)localArrayList[i]).Password);
                        password = ((Magician)localArrayList[i]).Password;

                        ListBoxMagiciansFavTricks.Items.Clear();
                        ListBoxMagiciansFavTricks.Items.Add(((Magician)localArrayList
                        [i]).getFavouritTricks());

                        found = true;
```

# Magical Show

The code behind setting up the account page

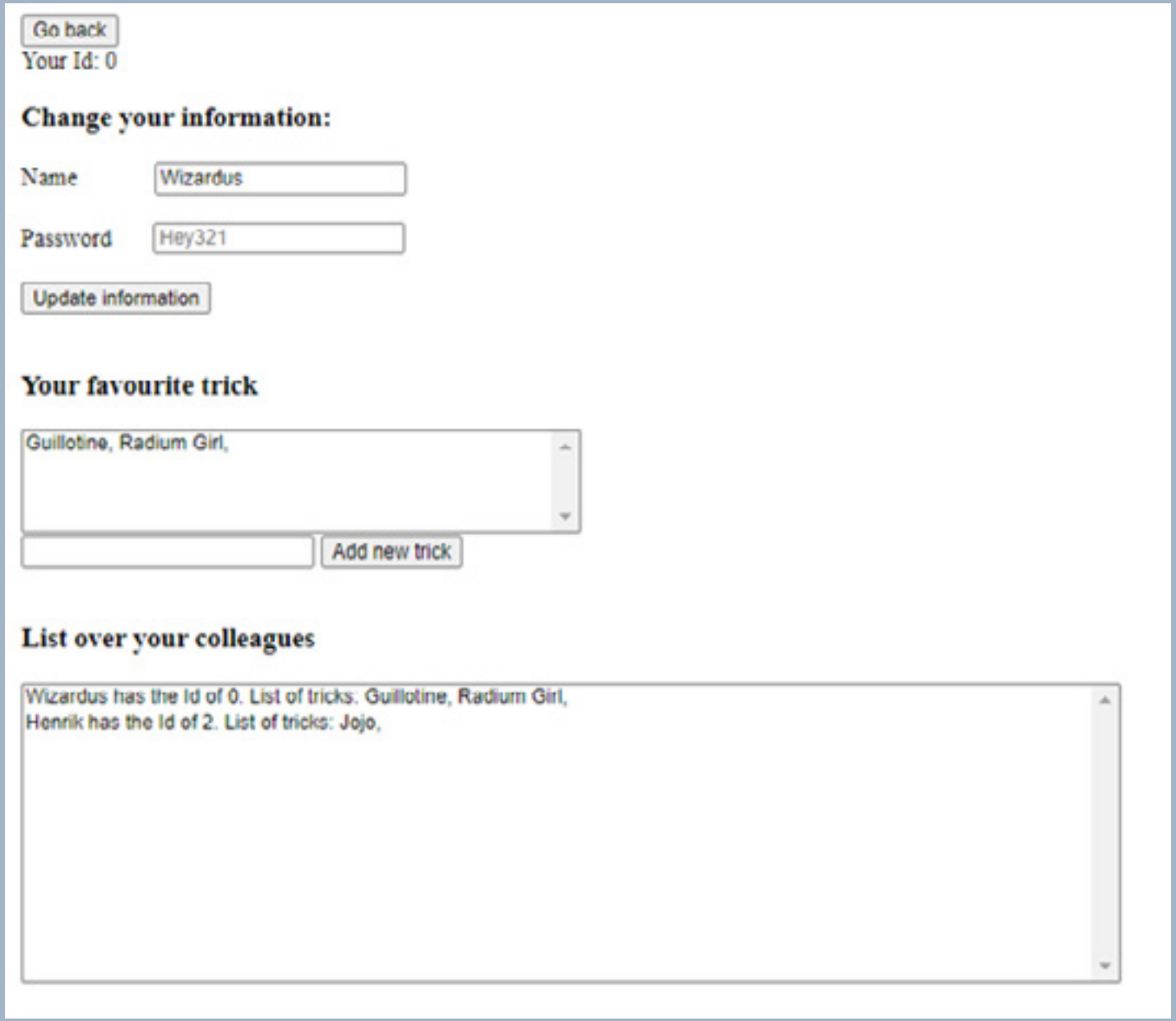
# Magical Show

Make pages for displaying and updating magicians and staff including adding magicians favourite tricks.

A person must be able to change own data only.

Magicians who have logged in must be able to see a list of all magicians.

For the magicians, the account page looks like this



```
protected void ButtonUpdateMagician_Click(object sender, EventArgs e)
        {
            if (TextBoxChangeMagiPassword.Text != "" || TextBoxChangeMagiPassword.Text
            != null)
            {
                for (int i = 0; i < localArrayList.Count; i++)
                {
                    try
                    {
                        if (((Magician)localArrayList[i]).Id == userid)
                        {
                            if(TextBoxChangeMagiPassword.Text == "" ||
                            TextBoxChangeMagiPassword.Text == null)
                            {
                                ((Magician)localArrayList[i]).Password = password;
                            }
                            else
                            {
                                ((Magician)localArrayList[i]).Password =
                                TextBoxChangeMagiPassword.Text;
                            }

                            Application["personcollection"] = localArrayList;
                            SetUpPage();
                        }
                    }
                    catch
                    {
                        LabelChangeMagiMsg.Text = "Your information is updated";
                    }
                }
            }
            else
            {
                LabelChangeMagiMsg.Text = "You need to write a password";
```

# Magical Show

Magician have two options.

1. Update password

I wanted to show what the information is at every point, therefore the Page_Load function inserts the information in the textboxes. (see code)

The password (and the other fields for the staff) has the current value in the placeholder attribute.

This is not good user experience, but the solution I went with.

When you click the Update button it will use the inserted text or the placeholder text if nothing is inserted

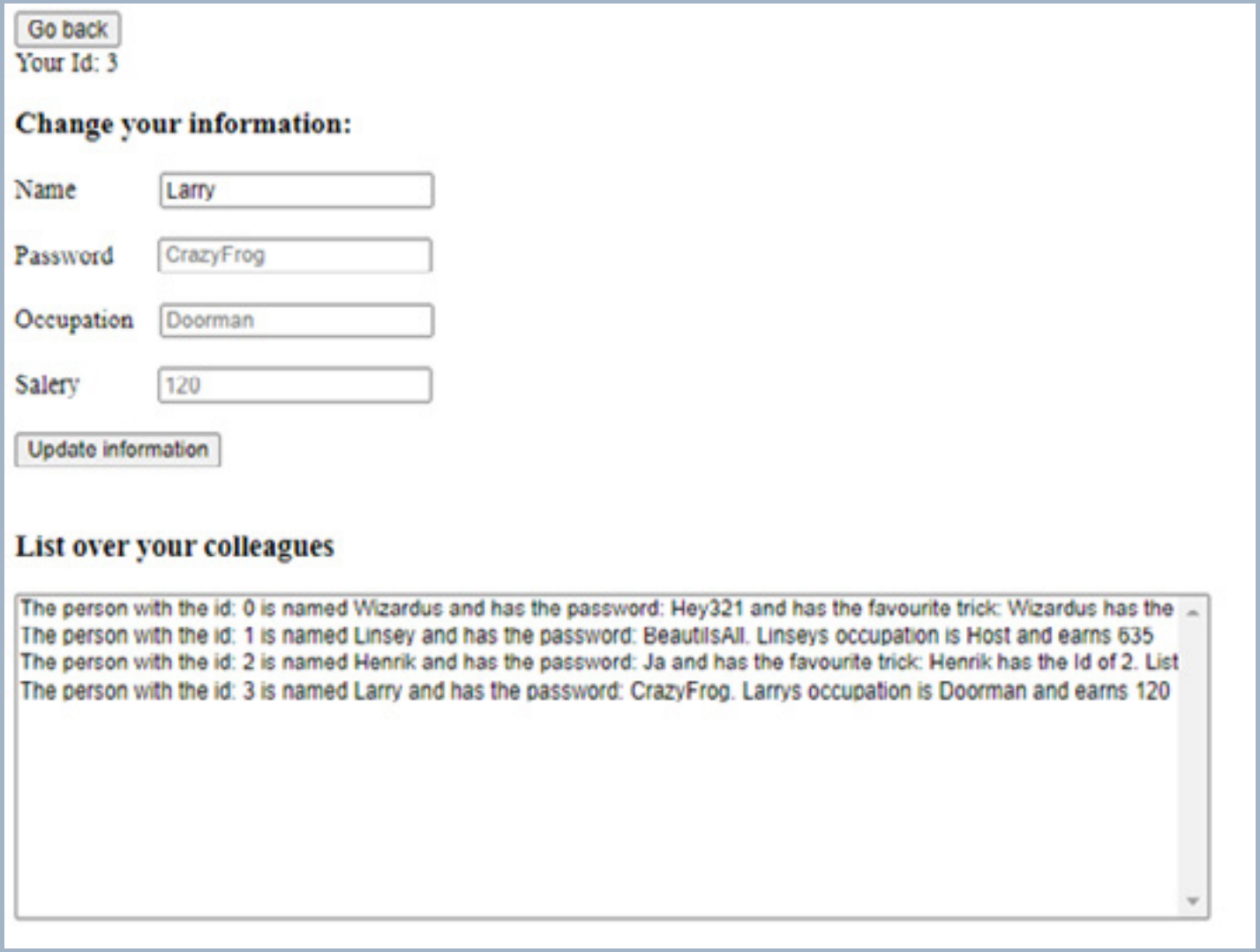
# Magical Show

## 2. Add trick

Because there is no limit to the favouriteTrick arraylist in the Magician class, we can insert as many tricks as we like.

```
protected void ButtonAddTrick_Click(object sender, EventArgs e)
{
    if (TextBoxNewTrick.Text != "" || TextBoxNewTrick.Text != null)
    {
        for (int i = 0; i < localArrayList.Count; i++)
        {
            try
            {
                if (((Magician)localArrayList[i]).Id == userid)
                {
                     ((Magician)localArrayList[i]).AddfavouriteTrick(TextBoxNe
                     wTrick.Text);
                    SetUpPage();
                }
            }
            catch
            {
                LabelAddTrickError.Text = "You have added a trick";
            }
        }
    }
    else
    {
        LabelAddTrickError.Text = "You need to write a trick";
```

# Magical Show




Make pages for displaying and updating magicians and staff including adding magicians favourite tricks.

A person must be able to change own data only.

Staff who have logged in must be able to see a list of all persons.

For a Staff person the account page look like this.

Note that the Magician only could see a list over all magician and their id, name and trick. But a staff member can ses everyone's information including everyone's password.

# Magical Show

Apart from the fact that there are some more input fields, the code is the same as the Update function for the Magicians.

```
protected void ButtonUpdateStaff_Click(object sender, EventArgs e)
        {
            if (TextBoxChangeStaffPassword.Text != "" || TextBoxChangeStaffName.Text
!= null)
            {
                for (int i = 0; i < localArrayList.Count; i++)
                {
                    try
                    {
                        if (((Staff)localArrayList[i]).Id == userid)
                        {
                            if(TextBoxChangeStaffPassword.Text == "" ||
                            TextBoxChangeStaffPassword.Text == null)
                            {
                                ((Staff)localArrayList[i]).Password = password;
                            }
                            else
                            {
                                ((Staff)localArrayList[i]).Password =
                                TextBoxChangeStaffPassword.Text;
                            }
                                if (TextBoxChangeOccupation.Text == "" ||
                                TextBoxChangeOccupation.Text == null)
                            {
                                ((Staff)localArrayList[i]).Occupation = Occupation;
                            }
                            else
                            {
                                ((Staff)localArrayList[i]).Occupation =
                                TextBoxChangeOccupation.Text;
                            }
                            if (TextBoxChangeSalery.Text == "" ||
                            TextBoxChangeSalery.Text == null)
                            {
                                ((Staff)localArrayList[i]).Salary = salery;
                            }
                            else
                            {
                                ((Staff)localArrayList[i]).Salary =
                                Convert.ToInt32(TextBoxChangeSalery.Text);
                            }
                            Application["personcollection"] = localArrayList;
                            SetUpPage();
                        }
                    }
                    catch
                    {
                        LabelChangeStaffMsg.Text = "Your information is updated";
                    }
                }

            }
            else
            {
                LabelChangeStaffMsg.Text = "Tou need to write a password";
```