Project3 实验报告

Fast Learn，电子教育解决方案

**小组成员：郁正远162210702133**

**孟凡猛162210702221**

**1.实验题目**当前，Fast Learn 使用传统的方法来开办课程教学。每个学生都会被分配到一个学习中心，每个星期都会进行一次联系会议。提供纸制的课程教材。学生必须访问各自的学习中心以获取学习材料。

将在联系会议中通过电话回答学生的询问。然而，通过电话询问解决方案取决于教师是否在场。由于学生数量众多，经常接到许多询问解决方案的电话。通过电话为如此多的学生解决问题是非常困难的。

管理层收到许多来自学生的抱怨，主要包括以下方面：

* 教师经常不能够通过电话对询问的解决方案作出回答。
* 学生必须访问学习中心以获取提供的与课程相关的免费信息。
* 学生必须访问学习中心以获取印刷的学习材料。

由于课程开设的现状，Fast Learn 正在逐渐背离它向学生提供快速学习的核心目标。

此外，学习中心的行政人员也面临着许多问题。引起管理层注意的主要问题是：

* 通过电话处理技术咨询会导致时间的浪费，因为教师将花费大多数时间来回答不同学生的类似询问。
* 从手工维护的记录中搜索学生帐户信息非常耗时。
* 存放印刷的课程资料既不经济也会影响环境。

根据管理层的主要观察，决定为 Fast Learn 开发一个电子教育站点。

**2.实验内容**

开发电子教育站点时需要遵循以下决定：

* 该站点将现有系统维护的所有数据存储在单个中心数据库中。数据库将包含以下信息：
* 学生详细信息
* 教师成员详细信息
* 学生查询和他们的解决方案
* 学生反馈
* 课程信息
* 站点将以电子形式向学生提供课程内容。
* 该站点将显示的所有信息包括课程详细信息、收费结构和学生须知。

该网站将供以下三种类型的用户使用： 学生、教师和管理员。Web 站点将向三种类型的用户分别提供不同的界面。

将向每个用户提供用户名称和密码以登录到站点。每个用户名将被映射到以下三个角色之一： 学生、教师和管理员。当用户提供他/她的用户名和密码以登录网站时，将根据用户的角色向其显示相应的界面。

如果用户未在 5 次尝试机会内登录，则将自动锁定用户帐户。用户将必须请求中心的工作人员手动解锁用户帐户。

学生的界面将提供以下工具：

* 查看课程信息： 该功能将允许学生搜索课程并查看机构提供的关于该课程的信息。
* 查看课程资料： 该功能将允许学生访问他们登记参加的教程的资料。
* 张贴查询： 该功能将允许学生向他们的教师发送查询并查看对这些查询的回答。
* 发送反馈： 该功能将允许学生提供所接受的服务的反馈。
* 更改密码： 该功能允许学生更改登录帐户的密码。

教师的界面将提供以下工具：

* 答案查询： 该功能允许教师查看学生查询并张贴该查询的答复。每个教师将可以仅查看针对其专攻的主题的查询。
* 查看课程资料： 该功能将允许教师查看他们教授的课程的资料。
* 更改密码： 该功能允许教师更改登录帐户的密码。

管理员界面将提供以下管理员工具：

* 课程管理： 该功能将允许管理员添加课程信息，例如课程名称、持续时间、前提条件和费用。
* 学生管理： 该功能将允许管理员添加、编辑和阻止学生帐户。
* 教师管理： 该功能将允许管理员添加、编辑和阻止教师帐户。
* 查看反馈： 该功能将允许管理员查看从学生处收到的反馈。
* 搜索信息： 该功能将允许管理员根据教师和学生的姓名、课程和学习中心来查看与其相关的所有信息。
* 张贴消息： 该功能允许管理员在网站上为学生显示注释和消息。
* 更改密码： 该功能允许管理员更改他/她们的登录帐户的密码。

为了开发电子教育站点，您需要分析现有系统以及预想系统，并相应地创建应用程序。可以使用三层架构设计新系统，该系统由以下几个层组成：

* 表示层： 该层由 Web 表单组成。表单可用于接受数据并向用户显示最终结果。此外，它可用于查看关于用户需求的信息。
* 业务逻辑层： 该层包含的代码提供用于创建电子教育站点的逻辑。
* 数据层： 该层包含 SQL Server 2005 数据库以存储与学生、教师和课程资料相关的数据。

Student

Course

Student ID

Enrolls For

Name

Address

Course

ID

Course Name

Fee

Prerequisites

Is Assigned

Study Center

Center ID

Name

Location

Teaches

Faculty

Faculty

ID

Address

Name

Gives

Feedback

Phone

Posts

Query

Query ID

Query Text

Date of Query Text

Date of Response Text

Response Text

Is Resolved By

Is Assigned

ID

Feedback Text

Date of Posting

Is Viewed By

Administrator

1. **源码：**

* //控制器

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.Entity;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using LG\_08\_Activity\_02.Models;

using LG\_08\_Activity\_02.DatabaseContext;

namespace LG\_08\_Activity\_02.Controllers

{

public class MovieController : Controller

{

private MovieDbContext db = new MovieDbContext();

public ActionResult Search()

{

return View();

}

[HttpPost]

public ActionResult Search(string q)

{

var movies = db.Movies.Include(m => m.genre).OrderBy(m =>m.MovieName).Where(m => m.MovieName.Contains(q));

return PartialView("\_searchresults", movies);

}

public string getRow(string displayName, string fieldValue)

{

string strResponse = "";

strResponse += "<tr style='font size:16pt;'>";

strResponse += "<td>";

strResponse += "<b>";

strResponse += displayName;

strResponse += "</b>";

strResponse += "</td>";

strResponse += "<td>";

strResponse += fieldValue;

strResponse += "</td>";

strResponse += "</tr>";

return strResponse;

}

[Authorize]

public ActionResult MovieRatings()

{

System.Xml.Linq.XDocument xmlDoc = System.Xml.Linq.XDocument.Load(System.Web.HttpContext.Current.Server.MapPath("~/MovieRatings/MovieRating.xml"));

var movies = from m in xmlDoc.Descendants("Movie")

select m;

string strResponse = "<table style='width:100%; padding:30px;'>";

foreach (var mov in movies)

{

strResponse += getRow("Movie Name", mov.Element("Name").Value);

strResponse += getRow("Director", mov.Element("Director").Value);

strResponse += getRow("Producer", mov.Element("Producer").Value);

strResponse += getRow("Cast", mov.Element("Cast").Value);

string rating = "";

for (int i = 0; i < Convert.ToInt32(mov.Element("Rating").Value); i++)

{

rating += "<img src='../../Images/Star.png' style='height:20px; width:20px;'/>";

}

strResponse += getRow("Rating", rating);

//to insert blank row after each movie rating record

strResponse += getRow("<br/>", "<br/>");

}

strResponse += "</table>";

ViewBag.movieRatings = strResponse;

return View();

}

[HttpPost]

/\* To accept even null values the nullable type (int?) is used.

Nullable types can represent all the values

of an underlying type, and an additional null value. \*/

public ActionResult Index(int? GenreID)

{

var movies = db.Movies.Include(m => m.genre);

ViewBag.GenreID = new SelectList(db.Genres, "GenreID", "strGenre");

if (GenreID == null)

{

//retrieve all records

movies = movies.OrderBy(k => k.MovieName);

}

else

{

movies = movies.Where(g => g.GenreID == GenreID).OrderBy(k => k.MovieName);

Response.Cookies["Genre"].Value = GenreID.ToString();

Response.Cookies["Genre"].Expires = DateTime.Now.AddDays(2);

}

return View(movies);

}

//

// GET: /Movie/

public ActionResult Index()

{

if (Request.Cookies["Genre"] != null)

{

var movies = db.Movies.Include(m => m.genre);

ViewBag.GenreID = new SelectList(db.Genres, "GenreID", "strGenre", Request.Cookies["Genre"].Value);

var Genre = Request.Cookies["Genre"].Value;

int GenreI = Convert.ToInt32(Genre);

movies = movies.Where(g => g.GenreID == GenreI).OrderBy(k => k.MovieName);

return View(movies);

}

else

{

var movies = db.Movies.Include(m => m.genre).OrderBy(k => k.MovieName);

ViewBag.GenreID = new SelectList(db.Genres, "GenreID", "strGenre");

return View(movies);

}

}

//

// GET: /Movie/Details/5

public ActionResult Details(long id = 0)

{

Movie movie = db.Movies.Find(id);

if (movie == null)

{

return HttpNotFound();

}

return View(movie);

}

//

// GET: /Movie/Create

[Authorize(Roles = "Admin")]

public ActionResult Create()

{

ViewBag.GenreID = new SelectList(db.Genres, "GenreID", "strGenre");

return View();

}

//

// POST: /Movie/Create

[HttpPost]

[Authorize]

public ActionResult Create(Movie movie)

{

if (ModelState.IsValid)

{

db.Movies.Add(movie);

db.SaveChanges();

return RedirectToAction("Index");

}

ViewBag.GenreID = new SelectList(db.Genres, "GenreID", "strGenre", movie.GenreID);

return View(movie);

}

//

// GET: /Movie/Edit/5

[Authorize(Roles = "Admin")]

public ActionResult Edit(long id = 0)

{

Movie movie = db.Movies.Find(id);

if (movie == null)

{

return HttpNotFound();

}

ViewBag.GenreID = new SelectList(db.Genres, "GenreID", "strGenre", movie.GenreID);

return View(movie);

}

//

// POST: /Movie/Edit/5

[HttpPost]

[Authorize]

public ActionResult Edit(Movie movie)

{

if (ModelState.IsValid)

{

db.Entry(movie).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("Index");

}

ViewBag.GenreID = new SelectList(db.Genres, "GenreID", "strGenre", movie.GenreID);

return View(movie);

}

//

// GET: /Movie/Delete/5

[Authorize(Roles = "Admin")]

public ActionResult Delete(long id = 0)

{

Movie movie = db.Movies.Find(id);

if (movie == null)

{

return HttpNotFound();

}

return View(movie);

}

//

// POST: /Movie/Delete/5

[HttpPost, ActionName("Delete")]

[Authorize]

public ActionResult DeleteConfirmed(long id)

{

Movie movie = db.Movies.Find(id);

db.Movies.Remove(movie);

db.SaveChanges();

return RedirectToAction("Index");

}

protected override void Dispose(bool disposing)

{

db.Dispose();

base.Dispose(disposing);

}

}

* }
* //模型

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Web;

namespace LG\_08\_Activity\_02.Models

{

public class Movie

{

public long ID { get; set; }

[Required(ErrorMessage = "Please provide the Movie name.")]

[StringLength(255, ErrorMessage = "The name of the Movie cannot have more than 255 characters.")]

[Display(Name = "Movie Name")]

public string MovieName { get; set; }

[Required(ErrorMessage = "Please provide the Director name.")]

[StringLength(50, ErrorMessage = "The name of the movie Director cannot have more than 50 characters.")]

[Display(Name = "Director")]

public string Director { get; set; }

[Required(ErrorMessage = "Please provide the Producer name.")]

[StringLength(50, ErrorMessage = "The Producer name cannot have more than 50 characters.")]

[Display(Name = "Producer")]

public string Producer { get; set; }

[Required(ErrorMessage = "Please provide the star cast.")]

[StringLength(255, ErrorMessage = "Star cast cannot have more than 255 characters.")]

[Display(Name = "Star Cast")]

public string Cast { get; set; }

[Required(ErrorMessage = "Please provide the duration of a movie.")]

[Range(1, 240, ErrorMessage = "The duration of a movie should be between 1 and 240 minutes.")]

[Display(Name = "Duration (minutes)")]

public int Duration { get; set; }

[StringLength(255, ErrorMessage = "The movie description can not have more than 255 characters.")]

[DataType(DataType.MultilineText)]

[Display(Name = "Description")]

public string Description;

//Navigational Property

public virtual Genre genre { get; set; }

//Foreign Key

public int GenreID { get; set; }

}

* }
* //登陆

@if (Request.IsAuthenticated) {

<text>

Hello, @Html.ActionLink(User.Identity.Name, "Manage", "Account", routeValues: null, htmlAttributes: new { @class = "username", title = "Manage" })!

@using (Html.BeginForm("LogOff", "Account", FormMethod.Post, new { id = "logoutForm" })) {

@Html.AntiForgeryToken()

<a href="javascript:document.getElementById('logoutForm').submit()">Log off</a>

}

</text>

} else {

<span>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</span>

@Html.ActionLink("Register", "Register", "Account", routeValues: null, htmlAttributes: new { id = "registerLink" })

<span>&nbsp;|&nbsp;</span>

@Html.ActionLink("Log in", "Login", "Account", routeValues: null, htmlAttributes: new { id = "loginLink" })

}

<div class="frmAccount">

@model LG\_08\_Activity\_02.Models.LoginModel

@{

ViewBag.Title = "Log in";

}

<hgroup class="title">

<h1>@ViewBag.Title.</h1>

</hgroup>

<section id="loginForm">

<h2>Use a local account to log in.</h2>

@using (Html.BeginForm(new { ReturnUrl = ViewBag.ReturnUrl }))

{

@Html.AntiForgeryToken()

@Html.ValidationSummary(true)

<fieldset>

<legend>Log in Form</legend>

<ol>

<li>

@Html.LabelFor(m => m.UserName)

@Html.TextBoxFor(m => m.UserName)

@Html.ValidationMessageFor(m => m.UserName)

</li>

<li>

@Html.LabelFor(m => m.Password)

@Html.PasswordFor(m => m.Password)

@Html.ValidationMessageFor(m => m.Password)

</li>

<li>

@Html.CheckBoxFor(m => m.RememberMe)

@Html.LabelFor(m => m.RememberMe, new { @class = "checkbox" })

</li>

</ol>

<input type="submit" value="Log in" />

</fieldset>

<p>

@Html.ActionLink("Register", "Register") if you don't have an account.

</p>

}

</section>

<section class="social" id="socialLoginForm">

<h2>Use another service to log in.</h2>

@Html.Action("ExternalLoginsList", new { ReturnUrl = ViewBag.ReturnUrl })

</section>

@section Scripts {

@Scripts.Render("~/bundles/jqueryval")

}

</div>

//注册

<div class="frmAccount">

@model LG\_08\_Activity\_02.Models.RegisterModel

@{

ViewBag.Title = "Register";

}

<hgroup class="title">

<h1>@ViewBag.Title.</h1>

<h2>Create a new account.</h2>

</hgroup>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

@Html.ValidationSummary()

<fieldset>

<legend>Registration Form</legend>

<ol>

<li>

@Html.LabelFor(m => m.UserName)

@Html.TextBoxFor(m => m.UserName)

</li>

<li>

@Html.LabelFor(m => m.Password)

@Html.PasswordFor(m => m.Password)

</li>

<li>

@Html.LabelFor(m => m.ConfirmPassword)

@Html.PasswordFor(m => m.ConfirmPassword)

</li>

</ol>

<input type="submit" value="Register" />

</fieldset>

}

@section Scripts {

@Scripts.Render("~/bundles/jqueryval")

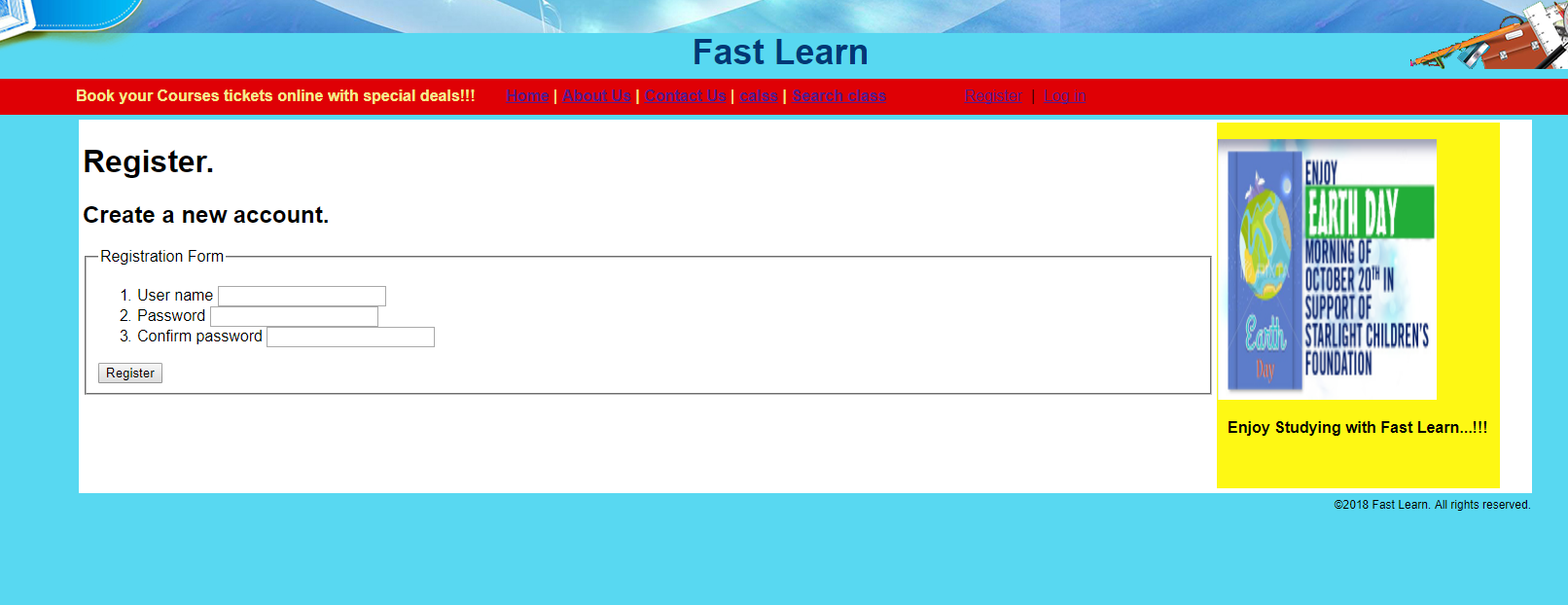
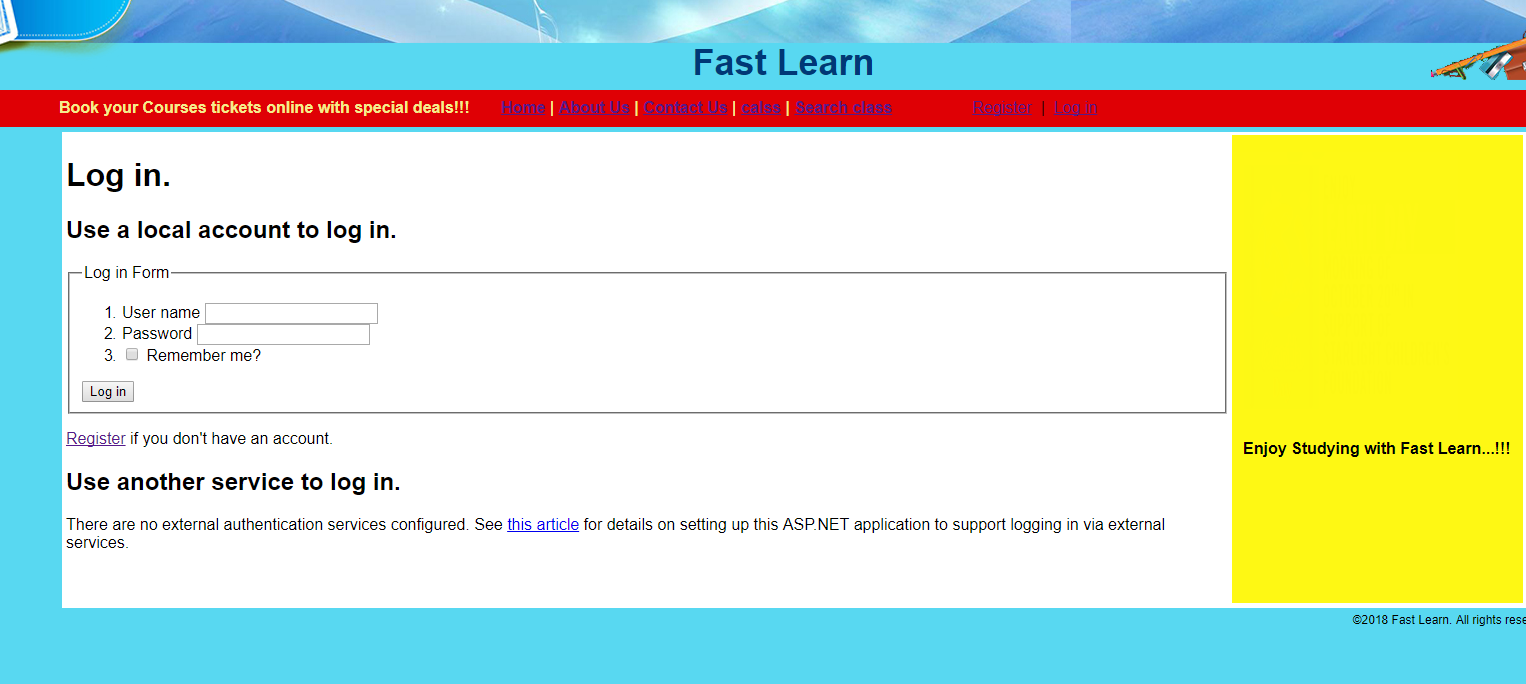
}

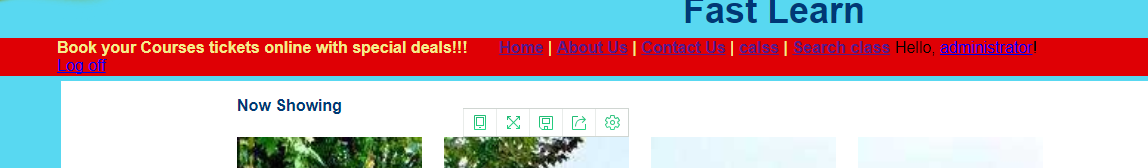
</div>

1. **实验结果：**

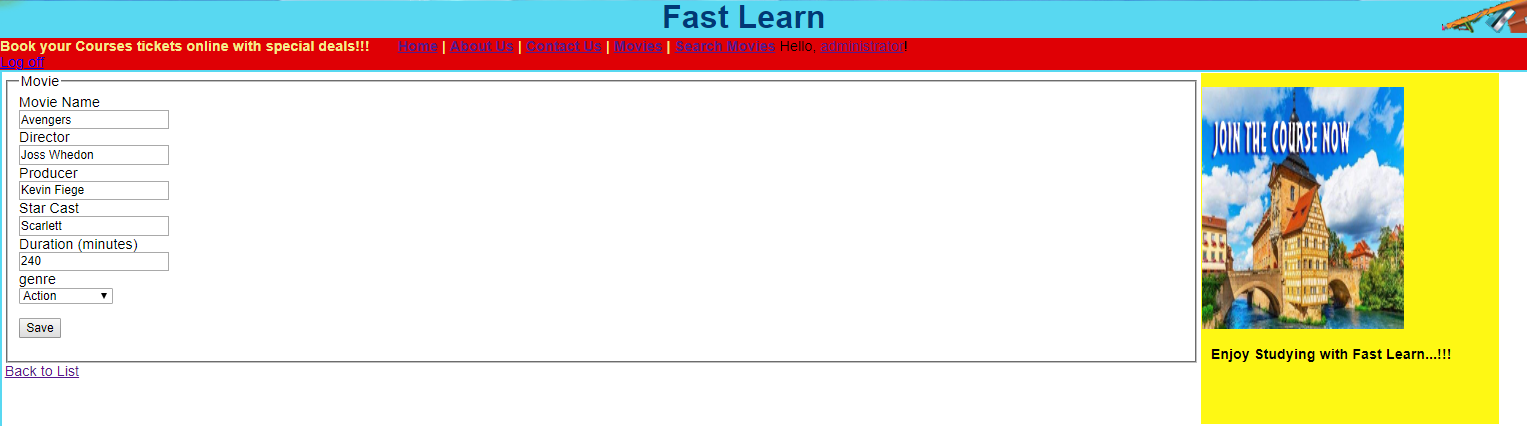
**主页**



* **注册界面**
* ****
* **登陆**
* ****
* **以普通用户身份**
* ****
* **管理员身份**

****

**以管理员身份修改信息：**

****

**搜索个人信息：**

