FIT5032 (Suzhou)

Internet Applications Development

Week 6: Sending Email, File Upload and Signal R

Murray Mount / ABM Russel

Unit Topics

Week	Activities	Assessment
0		No formal assessment or activities are undertaken in week 0
1A	Intro to Web development and ASP.NET	Note: Studio classes commence in week 1
1B	The front end, user experience, accessibility and ASP.NET Scaffolding	
2	Introduction to C# & Version Control	
3	Entity Framework	
4	Fundamentals of Client side Javascript	Studio assessment task 1 due
5A	Validation	
5B	Security and Identity	
6	Sending Email, File Upload and Signal R	Studio assessment task 2 due
7	Web Optimisations & Evolution of ASP.NET CORE	
8A	Modern JavaScript Web Development Approaches	
8B	Testing and Deployment in Cloud	Final Portfolio and Learning Summary due
9	Review & Revision	
	Examination period	
		LINK to Assessment Policy:http://policy.monash.edu.au/policy-bank/academic/education/assessment/assessment-in-coursework-

Today

- Recap: Security and Identity
- Sending Email with ASP.NET
- Accessing the Web Server File System
- SignalR

Recap: Security and Identity

Log In Concepts

- Almost all real world web applications require users to log in to the website
    - to use more than the basic functionality.
- Require usernames and passwords
- Some applications use role based authentication
    - administrator roles, user roles etc
- Security and account information stored
    - on file system
    - or database

Log In Systems for ASP.Net MVC

- ASP.Net MVC application
    - Can auto-generate applications with log in functionality
- Basic ASP.Net MVC application
    - with users
        - register
        - interact with public areas before log in
        - interact with private areas after log in

Securing an Action

- An Action (e.g. from the HomeController) can be restricted to logged in users
    - Use the [Authorize] annotation

```
[Authorize]
 public ActionResult Contact()
 {
     ViewBag.Message = "Your contact page.";

     return View();
 }
```

- Now the user must log in to access the Contact action.

Securing/Unsecuring Actions

- Smaller sections secured
    - adding the “[Authorize]” annotation to the action.
- Secured controller, can have unsecured action
    - “[AllowAnonymous]”annotation for that action.

Securing Controllers/Actions based on roles

- Application (controllers and actions)
    - secured using the roles
    - defined for the application (in the AspNetRoles table)
- Use “[Authorize(Roles = “Administrator”)]”
    - name of the roles are your choice.

Allowing Access to Own Data (Only)

- Selecting/Viewing items owned by log in user
    - ASP.Net MVC allows us to access the currently logged in user:

```
using Microsoft.AspNet.Identity;
......

string currentUserId = User.Identity.GetUserId();
.....
```

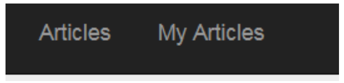
Selecting/Viewing items owned by log in user (Part 2)

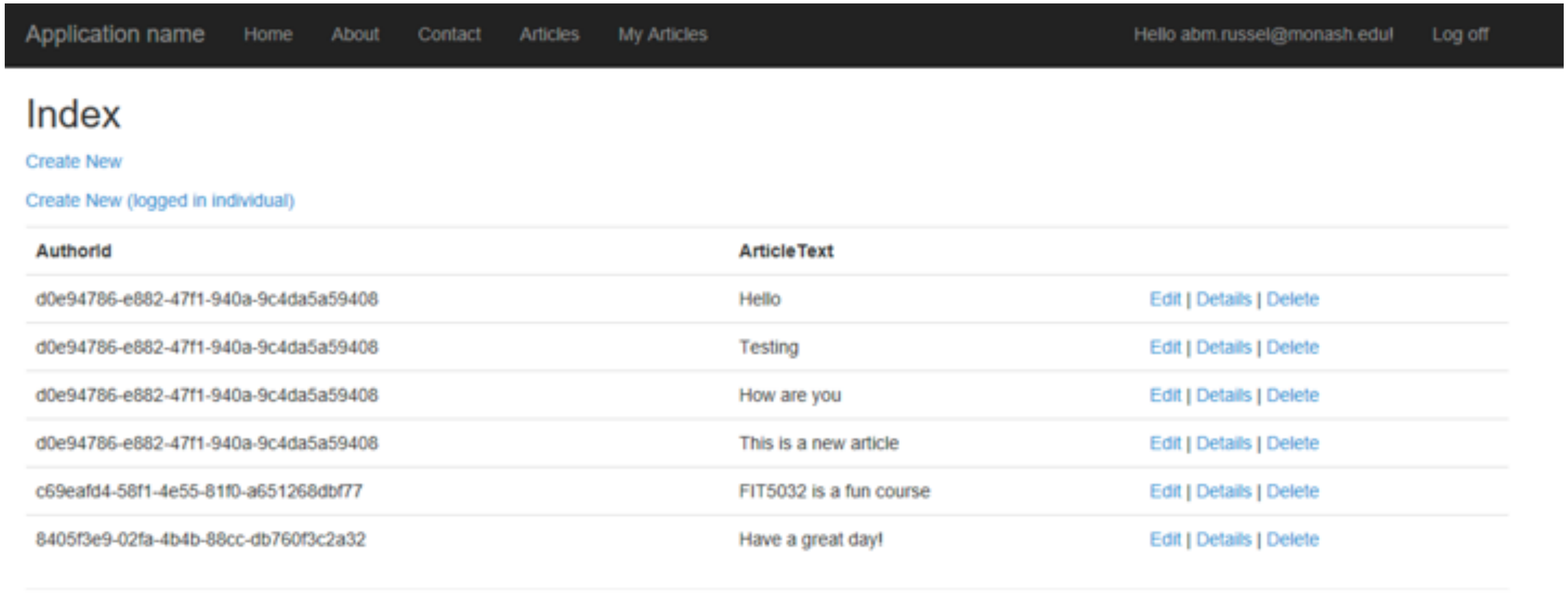
- User id to select just the items that are created by the user (for viewing in the index view.)

```
// GET: Articles
        public ActionResult IndexUserNames()
        {
            //return View(db.Articles.ToList());
            string currentUserId = User.Identity.GetUserId();
            return View(db.Articles.Where(m=> m.AuthorId ==
currentUserId).ToList());
        }
```

Selecting/Viewing items owned by log in user (Part 3)

- Only the users own data is shown





Index

Create New

Create New (logged in individual)

AuthorId	ArticleText	
d0e94786-e882-47f1-940a-9c4da5a59408	Hello	Edit | Details | Delete
d0e94786-e882-47f1-940a-9c4da5a59408	Testing	Edit | Details | Delete
d0e94786-e882-47f1-940a-9c4da5a59408	How are you	Edit | Details | Delete
d0e94786-e882-47f1-940a-9c4da5a59408	This is a new article	Edit | Details | Delete
c69eafd4-58f1-4e55-81f0-a651268dbf77	FIT5032 is a fun course	Edit | Details | Delete
8405f3e9-02fa-4b4b-88cc-db760f3c2a32	Have a great day!	Edit | Details | Delete

Creating item (automatically adding userID)

- Action takes the current user id and adds it to the model before calling the View (with the model)

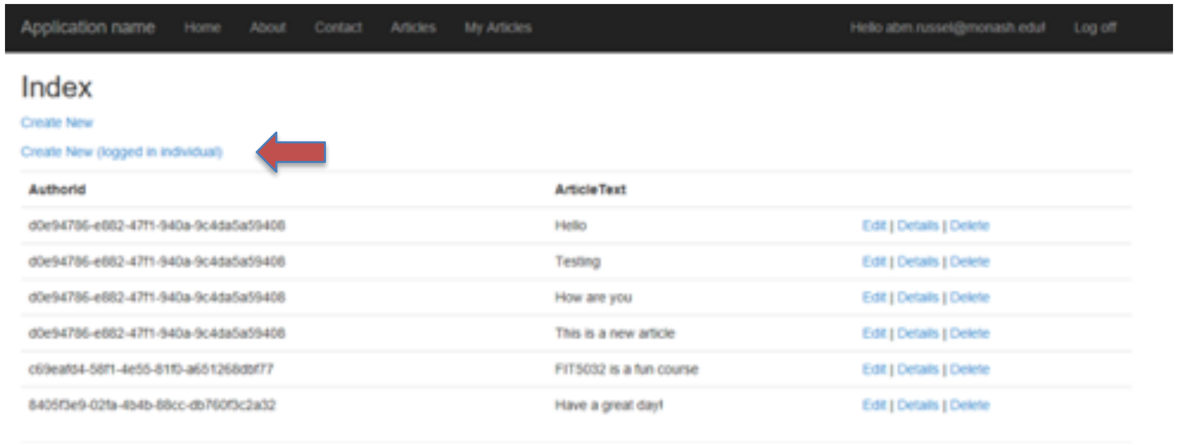
```
// GET: Articles/Create
        public ActionResult CreateIndividual()
        {
            Article article = new Article();
            string currentUserId = User.Identity.GetUserId();
            article.AuthorId = currentUserId;
            return View(article);
        }
```

Creating item (automatically adding userID) Part 2

- Process the Posted Model Values when the user Completes the form

```
  [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult CreateIndividual([Bind(Include = "NewsId, AuthorId, 
ArticleText")] Article article)
       …..
```

Creating item (automatically adding userID) Part 3

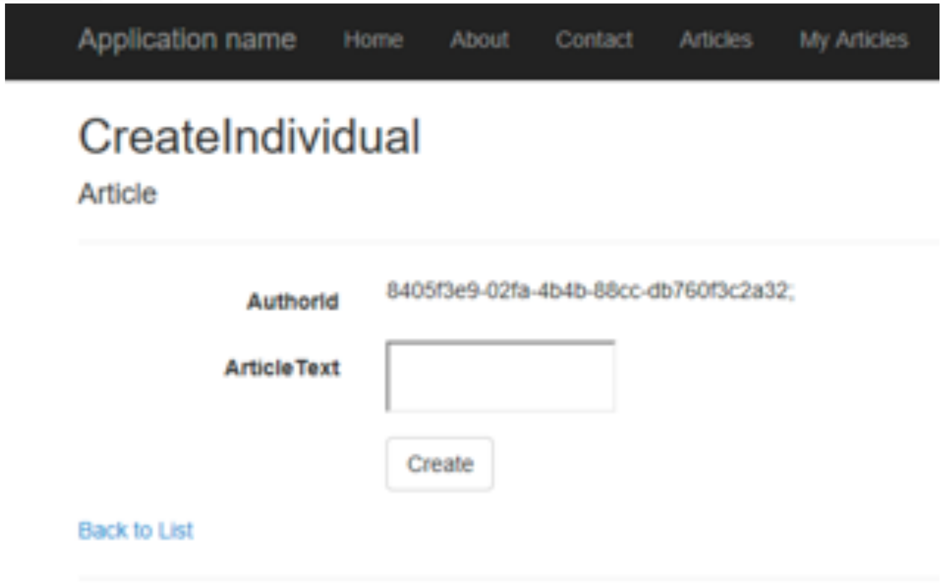


Index

Create New

Create New (logged in individual)

AuthorId	ArticleText	
d0e94786-e882-47f1-940a-9c4da5a59408	Hello	Edit | Details | Delete
d0e94786-e882-47f1-940a-9c4da5a59408	Testing	Edit | Details | Delete
d0e94786-e882-47f1-940a-9c4da5a59408	How are you	Edit | Details | Delete
d0e94786-e882-47f1-940a-9c4da5a59408	This is a new article	Edit | Details | Delete
c69eafd4-58f1-4e55-81f0-a651268dbf77	FIT5032 is a fun course	Edit | Details | Delete
8405f3e9-02fa-4b4b-88cc-db760f3c2a32	Have a great day!	Edit | Details | Delete



CreateIndividual

Article

AuthorId	8405f3e9-02fa-4b4b-88cc-db760f3c2a32;
ArticleText	

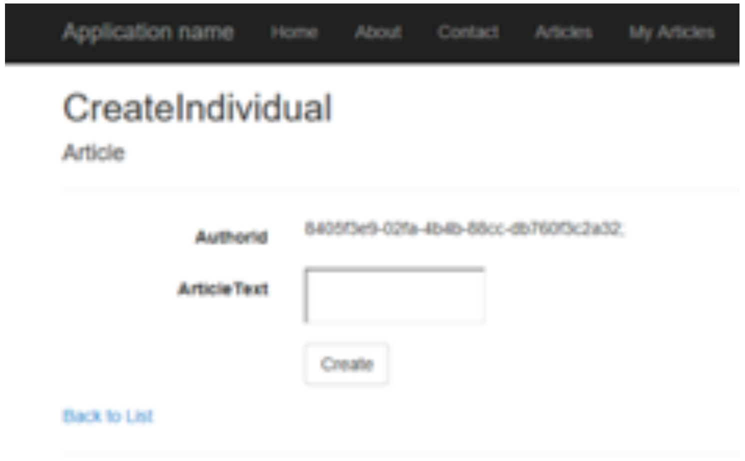
Back to List

Creating item (automatically adding userID) Part 4

- A hidden field (model.AuthorId) is required to pass the AuthorID to the user.

```
@Html.HiddenFor(model => model.AuthorId,
htmlAttributes: new { @class = "form-control"})
```

- Normally the internal Id values (such as AuthorID) are not displayed in the user interface, this is just for debugging purposes.



CreateIndividual

Article

Authorid 8405f3e9-02fa-4b4b-88cc-db760f3c2a32;

ArticleText

Email

Sending Email in an Application

- confirm a user who has registered
- distribute a monthly newsletter or
- request a forgotten password
- .NET Framework
  - SmtpClient class
  - MailMessage class
- Part of System.Net.Mail namespace.

System.Net.Mail

- SmtpClient class
    - sends the email using the Microsoft SMTP (Simple Mail Transport Protocol) Service included in IIS
- MailMessage class
    - contains properties as the message body, sender and receiver.

Sending email

Sending Email Example

```
using System.Net.Mail;


….
var body = "<p>Email From: {0} ({1})</p><p>Message:</p><p>{2}</p>";
var message = new MailMessage();
message.To.Add(new MailAddress(“abm.russel@monash.edu”));
// replace with valid value
message.From = new MailAddress(“abm.russel@monash.edu”);
// replace with valid value
message.Subject = "Your email subject";
message.Body = string.Format(body, model.FromName, model.FromEmail,
model.Message);
message.IsBodyHtml = true;
```

Change examples to your email

Model for email example

```
using System.ComponentModel.DataAnnotations;
using System.Web;
namespace Week8Email.Models
{
    public class EmailFormModel
    {
        [Required, Display(Name = "Your name")]
        public string FromName { get; set; }
        [Required, Display(Name = "Your email"), EmailAddress]
        public string FromEmail { get; set; }
        [Required]
        public string Message { get; set; }
     }
}
```

SmtpClient send method

- *SendMailAsync* method of the SmptClient class, which takes argument, message which includes the:

From
To
Subject
Message Text

MailMessage Object

MailAddress constructor parameters:

email address and a display name

email will be sent to the users email address

firstname and surname will be displayed in the To field of the email client when the email is received.

MailChimp & SendGrid

- MailChimp is the world's largest marketing automation platform.

- SendGrid developed an industry-disrupting, cloud-based email service to solve the challenges of reliably delivering emails on behalf of growing companies.

What is the main advantage of serverside email functionality

What is the main advantage of serverside email functionality?

A. No need to employ people to email information to clients/ customers

B. Better confidentiality as automated

C. Better quality of service as can be instantaneous 24/7

D. All of the answers (except none)

E. None of the answers

Does the ASP.Net email functionality use a built in mailserver (in IIS)

Does the ASP.Net email functionality use a built in mailserver (in IIS)?

A. Yes, it uses a builtin Mail Server

B. Yes, it uses a builtin Mail Server (but only in Visual Studio)

C. Yes, it uses a built in Mail Server, but it has to be manually configured

D. No, it uses an external Mail Server

E. No, it uses a Mail Client and an external Mail Server

What use is the display name field in the ASP.Net MailAddress object

What use is the display name field in the ASP.Net MailAddress object?

A. It provides extra security for the person sending the email

B. It allows the details to be displayed in email clients (as well as the email address)

C. It allows a user to server to specify how the receivers email addresses are displayed in the client

D. All the answers (except none)

E. None of the answers

Sending email with an attachment

Attachments

- Use the Attachment class

  a collection of the MailMessage object.

```
Attachment newAttach = new Attachment(Server.MapPath(“~/
    MyFile.txt”));
 newMsg.Attachments.Add(newAttach);
```

Adds "MyFile.txt", located in the root directory of web application, as an attachment to the email.

.

Attachments

- Can add an attachment sent by the user from a form

```
 if (model.Upload != null && model.Upload.ContentLength > 0)
            {
                message.Attachments.Add(new
Attachment(model.Upload.InputStream,
System.IO.Path.GetFileName(model.Upload.FileName)));
            }
```

Update model with

- public HttpPostedFileBase Upload { get; set; }

What sorts of attachments would be added to emails sent from an ASP.Net application

What sorts of attachments would be added to emails sent from an ASP.Net application?

A. Monthly newsletters in word format

B. Online tickets in pdf format

C. Image of booking confirmation in png format

D. All the answers (except none)

E. None of the answers

Why is it important to set a maximum file upload size

Why is it important to set a maximum file upload size?

A. To give the user an indication of the file sizes that should be uploaded

B. One way to stop a file upload attack on the server

C. So that files smaller that this limit are not accepted by the server

D. All the answers (except none)

E. None of the answers

Accessing the Web Server File System

File Upload

```
[HttpPost]
        public ActionResult Index(HttpPostedFileBase postedFile)
        {
            if (postedFile != null)
            {
                string path = Server.MapPath("~/Uploads/");
                if (!Directory.Exists(path))
                {
                    Directory.CreateDirectory(path);
                }
                postedFile.SaveAs(path +
Path.GetFileName(postedFile.FileName));
                ViewBag.Message = "File uploaded successfully.";
            }

            return View();
        }
```

File Upload

Specify a directory to save the uploaded file.

Server.MapPath("~") returns the root directory, e.g.
“C:\inetpub\wwwroot\ASPNET”.

Adding "\UploadFiles\" and the actual filename onto this path e.g.
“c:\inetpub\wwwroot\ASPNET\UploadFiles\image1.gif”.

PostedFile. SaveAs method, saves the file

Restricting File Extensions

File Upload

A good precaution is to restrict the types of files that users are able to upload. E.g. restrict uploads to files with certain extensions.

```
if ((strExt != ".gif") && (strExt !=".jpg"))
        {ErrorMessage = "Invalid File Type";}

else {
       binFileOK = true;
       strPath = Server.MapPath(“~”) + “/UploadFiles/”+ strFileName;
       fileUpload.PostedFile.SaveAs(strPath); }
        ErrorMessage = “File Saved”;
       }
```

Why is it a good idea to restrict the files uploaded to the expected types (e.g. .jpg etc)

Why is it a good idea to restrict the files uploaded to the expected types (e.g. .jpg etc)?

A. So that the expected file types are uploaded

B. To stop an attack, such as uploading .aspx files with malicious code

C. Some code may not work if the correct files types are not used (e.g. displaying unknown image types)

D. All the answers (except none)

E. None of the answers

System.IO Namespace

File Information

```
FileInfo file = new FileInfo(Server.MapPath("~/Uploads/Test.txt"));
            string fileProp;
fileProp = "<b>File Information</b><br />";
fileProp += "<b>Name:</b> " + file.Name + "<br />";
fileProp += "<b>Path:</b> " + file.DirectoryName + "<br />";
fileProp += "<b>Is Read Only:</b> " + file.IsReadOnly + "<br />";
fileProp += "<b>Last Access:</b> " + file.LastAccessTime + "<br />";
fileProp += "<b>Last Write:</b> " + file.LastWriteTime + "<br />";
fileProp += "<b>Length:</b> " + file.Length / 1024;
```

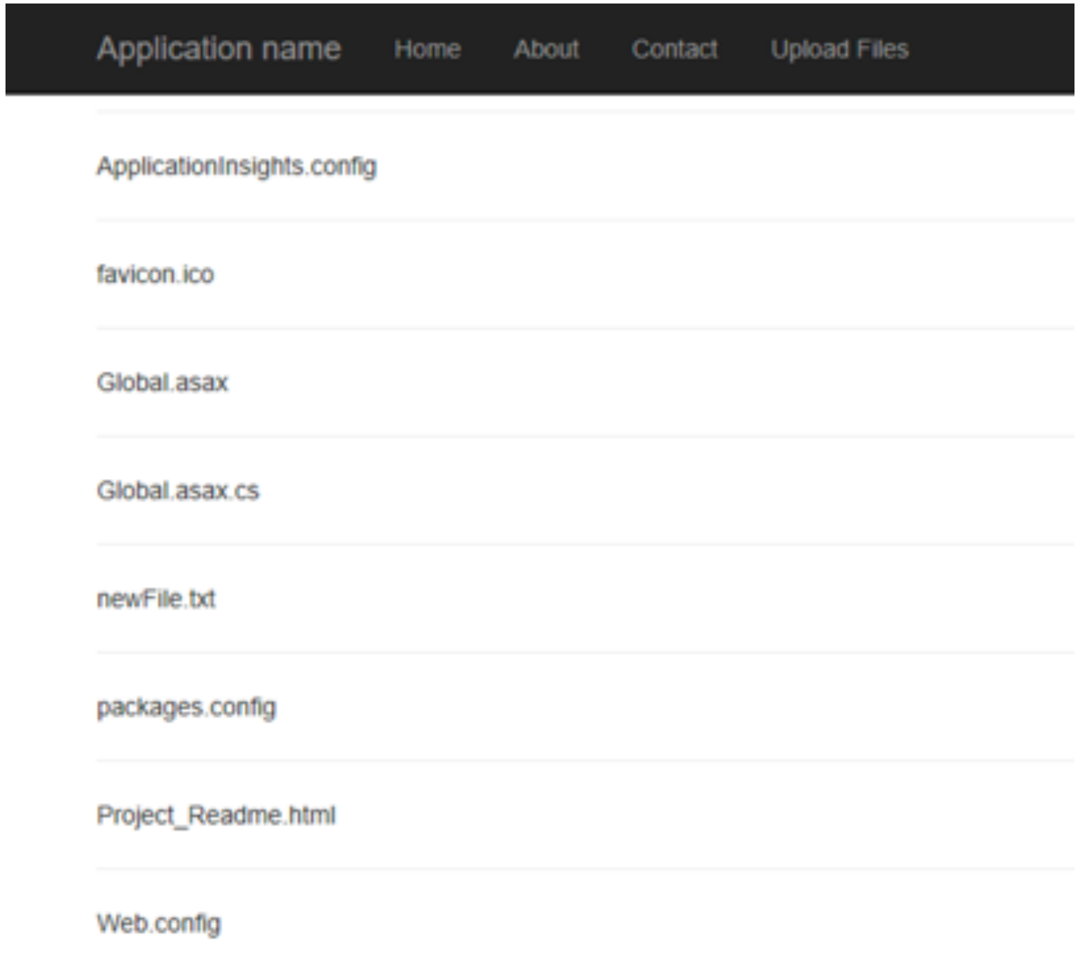
Directory Information

```
DirectoryInfo dir = new DirectoryInfo(Server.MapPath("~"));
string dirProp;

dirProp = "<b>Directory Information</b><br />";
dirProp += "<b>Name:</b> " + dir.Name + "<br />";
dirProp += "<b>Parent:</b> " + dir.Parent + "<br />";
dirProp += "<b>Full Name:</b> " + dir.FullName + "<br />";
dirProp += "<b>Attributes:</b> " + dir.Attributes + "<br />";
dirProp += "<b>Creation Time:</b> " + dir.CreationTime;
```

Iterating through the Files in a Directory

Listing Directory



ApplicationInsights.config

favicon.ico

Global.asax

Global.asax.cs

newFile.txt

packages.config

Project_Readme.html

Web.config

Listing Directory Example

```
[HttpGet]
        public ActionResult DirectoryListLink()
        {
            ArrayList fileList = new ArrayList();
            DirectoryInfo dir = new DirectoryInfo(Server.MapPath("~/"));
            foreach (FileInfo file in dir.GetFiles())
            {
                if (file.Extension != ".mdb")
                {
                    fileList.Add(file.Name);
                }
            }
            ViewBag.fileList = fileList;
            return View();
        }
```

Listing Directory Example

```
@{
    ViewBag.Title = "DirectoryList";
}

<h2>DirectoryList</h2>
@foreach (var item in ViewBag.fileList)
{
    <div>
        @item
        <hr />
    </div>
}
```

Why are listing of directory contents and file contents normally considered risky in terms of security

Why are listing of directory contents and file contents normally considered risky in terms of security?

A. Users are able to execute the code that is listed

B. Users are able to modify the code that is listed

C. User are able to see if there are any exploits

D. All the answers (except none)

E. None of the answers

Reading Files

Reading Files Example

Link to a file that ListFile action will display the contents of a file:

```
@foreach (var item in ViewBag.fileList)
{
    <div>
        @Html.ActionLink(
            linkText: (string) item,
            controllerName: "FileUpload",
            actionName: "ListFile”,
            routeValues: new
            {
                FileName = item
            },
            htmlAttributes: null
        )
        <hr />
    </div>
}
```

Checking Extension Type

```
string filePath = Server.MapPath("~/"+FileName);

FileInfo file = new FileInfo(filePath);
String Code;
if (file.Extension != ".mdb" && file.Extension != ".xml" &&
file.Extension != ".exe") {
  Code = ReadFile(filePath); }
else {
  Code = "Sorry you can't read a file with an extension of " +
file.Extension;
}
```

ReadFile function

```
StreamReader FileReader = new StreamReader(filepath);
//The returned value is -1 if no more characters are
//currently available.
while (FileReader.Peek() > -1) {
//ReadLine() Reads a line of characters from the
//current stream and returns the data as a string.
fileOutput += FileReader.ReadLine().Replace("<", "&lt;").
   Replace(" ", "&nbsp;&nbsp;&nbsp;&nbsp;") + "<br />"; }
FileReader.Close();
```

Creating, Copying and Deleting Files

Creating Files

```
string filePath = Server.MapPath("~/" + "/newFile.txt");

StreamWriter file = File.CreateText(filePath);
for (int i = 1; i <= 4; i++) {
  file.WriteLine("This is text line " + i);
}
file.WriteLine("The Date is " + DateTime.Now);
file.Close();
```

Copying Files

```
string fromPath = Server.MapPath("~") + "/newFile.txt";
string toPath = Server.MapPath("~") + "/newFile2.txt";

File.Copy(fromPath, toPath);
```

Overwriting Files

If the file to be copied to already exists, an error will be created.
The Copy method can take a third argument.
A boolean value, indicates if the destination file is to be overwritten if it already exists.

```
File.Copy(fromPath, toPath, true);
```

Copy operation succeeds and will overwrite the destination file if it already exists.

Deleting Files

```
string filePath = Server.MapPath("~") + "/newFile2.txt";

File.Delete(filePath);
```

File Permissions

Note: success of the file manipulation functions in this topic are dependent on the security permissions set on the web server.

If users do not have permission to create and/or delete files then the execution of these files will fail.

Drive Listings

Drive Information

The List of Drives on the machine can be retrieved:

```
drvList = DriveInfo.GetDrives();
```

If the drive is ready, the following drive properties are displayed to the user:

Name
DriveType (Fixed, CDRom, Network etc)
DriveFormat (NTFS, FAT32)
TotalSize (by default in bytes)
TotalFreeSpace (by default in bytes)
RootDirectory
VolumeLabel

Drive Info Code

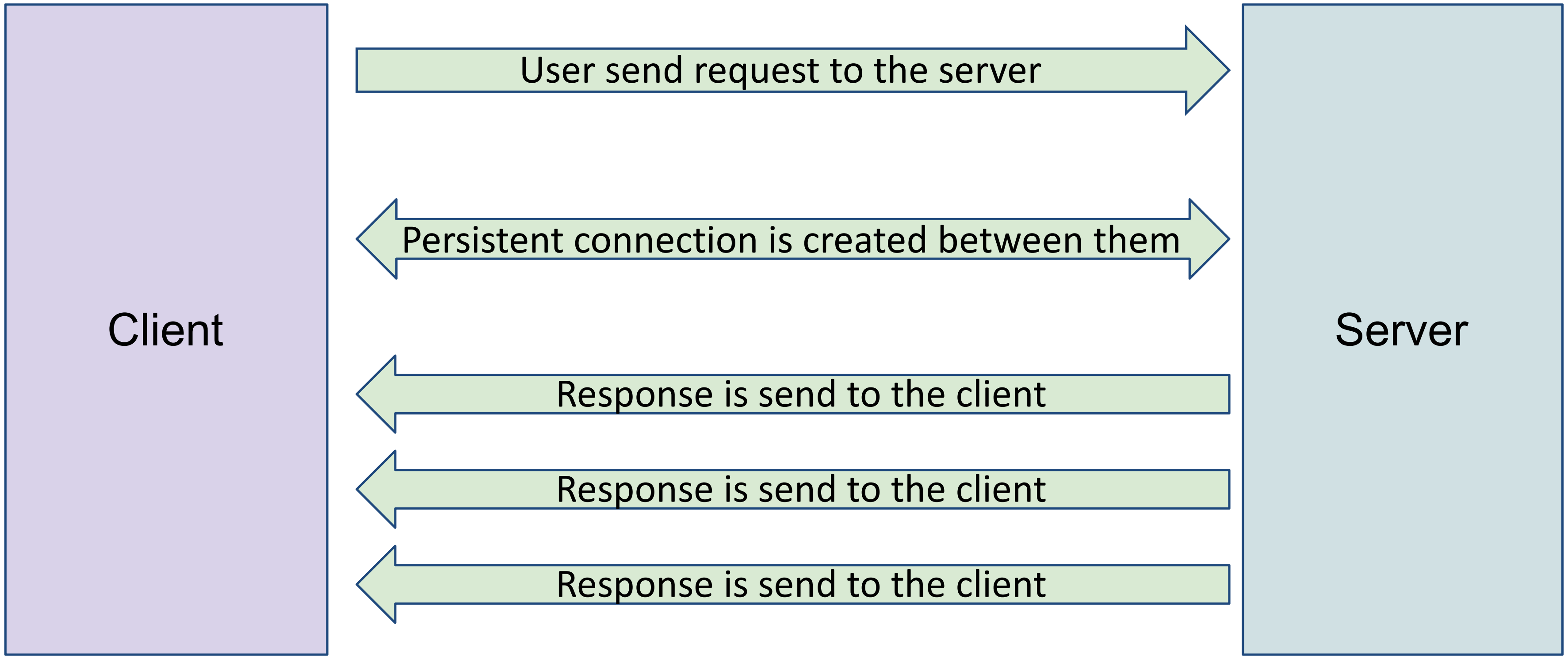
```
drvList[index].IsReady
drvList[index].Name
drvList[index].DriveType
drvList[index].DriveFormat
drvList[index].TotalSize
drvList[index].TotalFreeSpace
drvList[index].RootDirectory
drvList[index].VolumeLabel
```

SignalR

SignalR

- ASP.NET SignalR is a library for ASP.NET developers that simplifies the process of adding **real-time web functionality** **to applications**.
- Real-time web functionality is the ability to have server code push content to connected clients instantly as it becomes available, rather than having the server wait for a client to request new data.
- Use cases for SignalR:
  - Dashboards and monitoring applications,
  - collaborative applications (such as simultaneous editing of documents),
  - job progress updates, and real-time forms.
  - **One of the more obvious use case is the ability to create a “chat” room.**

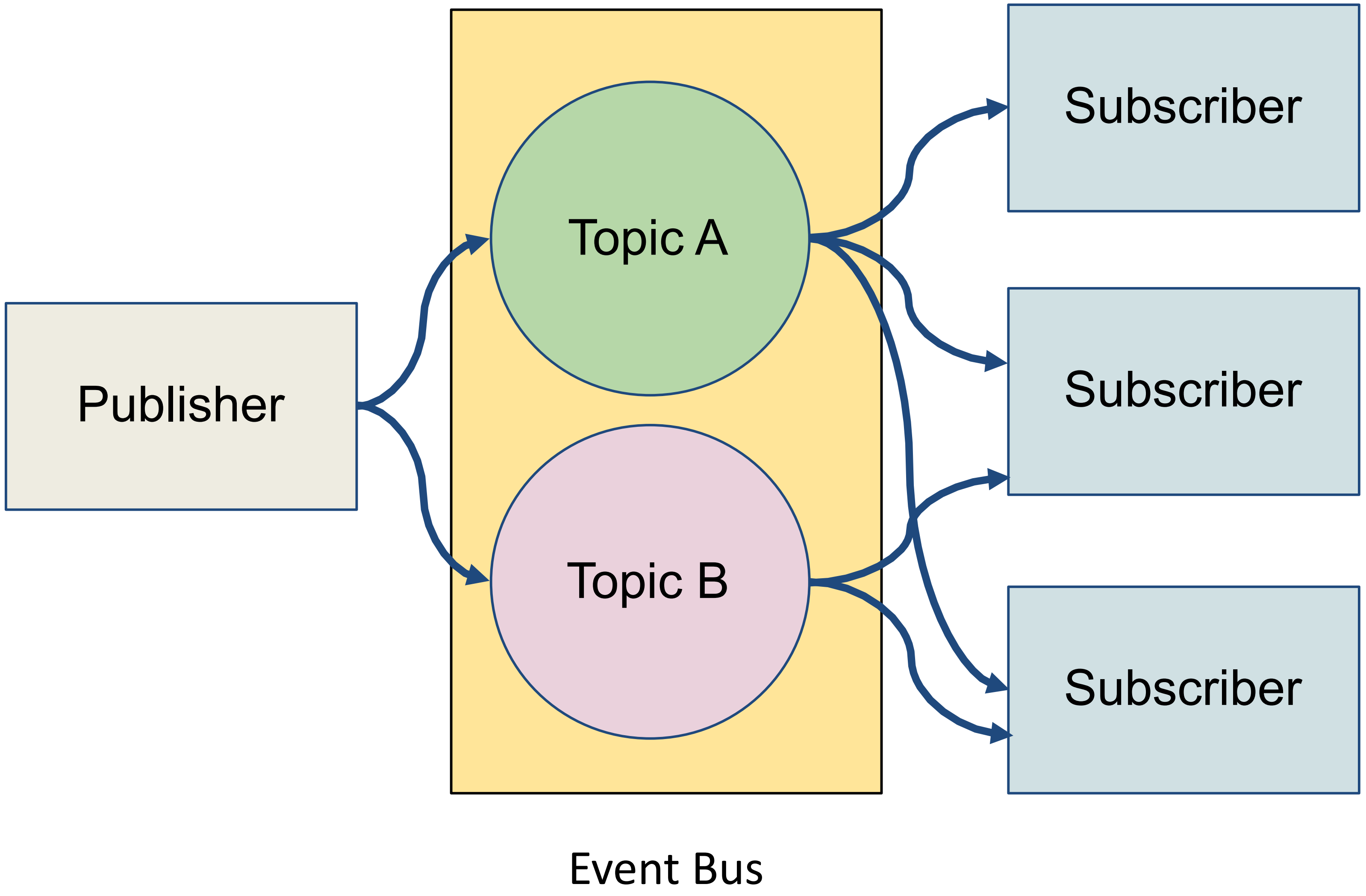
Real Time Web Functionality


Client
User send request to the server
Persistent connection is created between them
Response is send to the client
Response is send to the client
Response is send to the client
Server

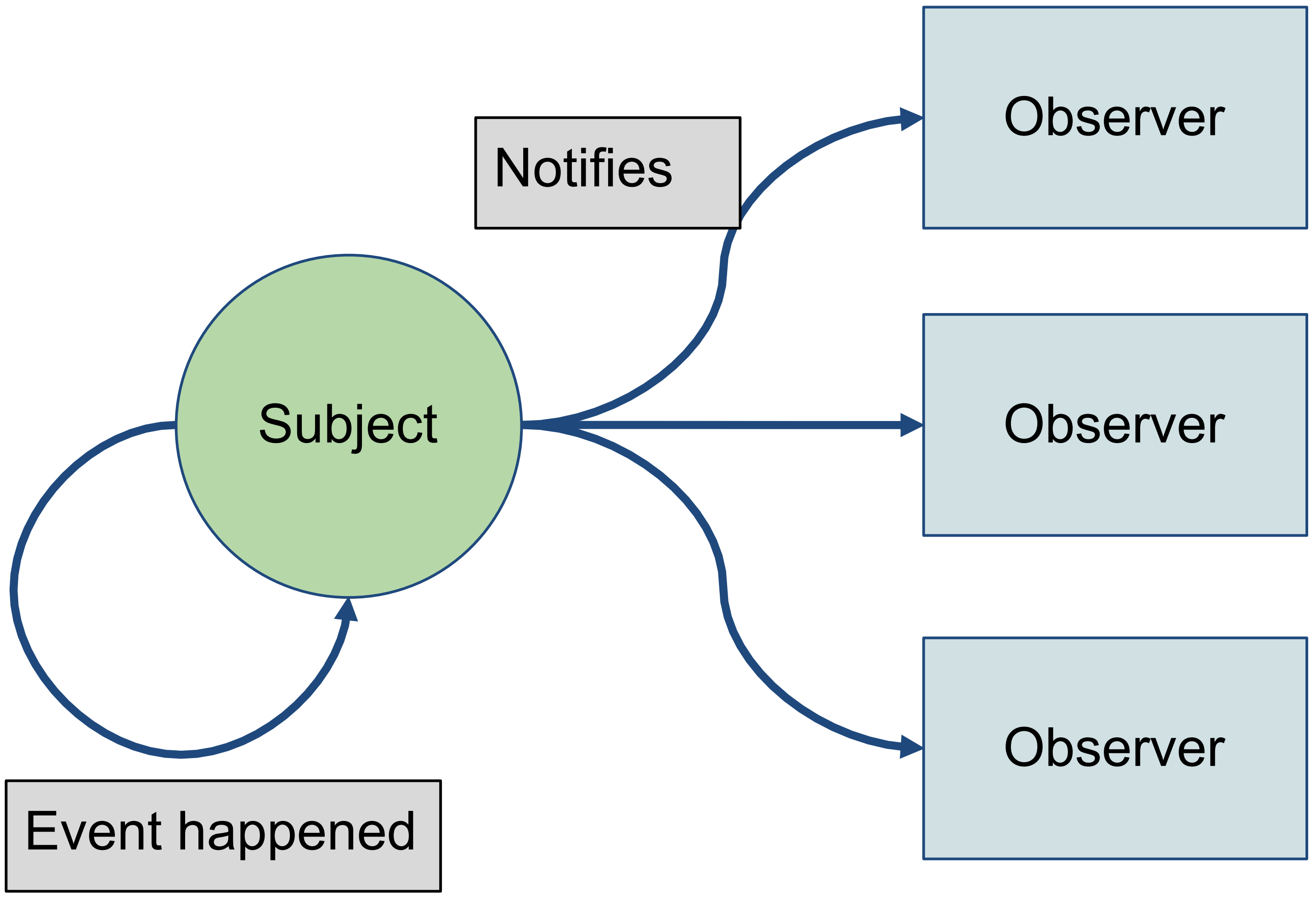
This happens **without the need to refresh** the browser. (hence real time)

The client receives update as soon as there is an update on the server.

Publish & Subscribe

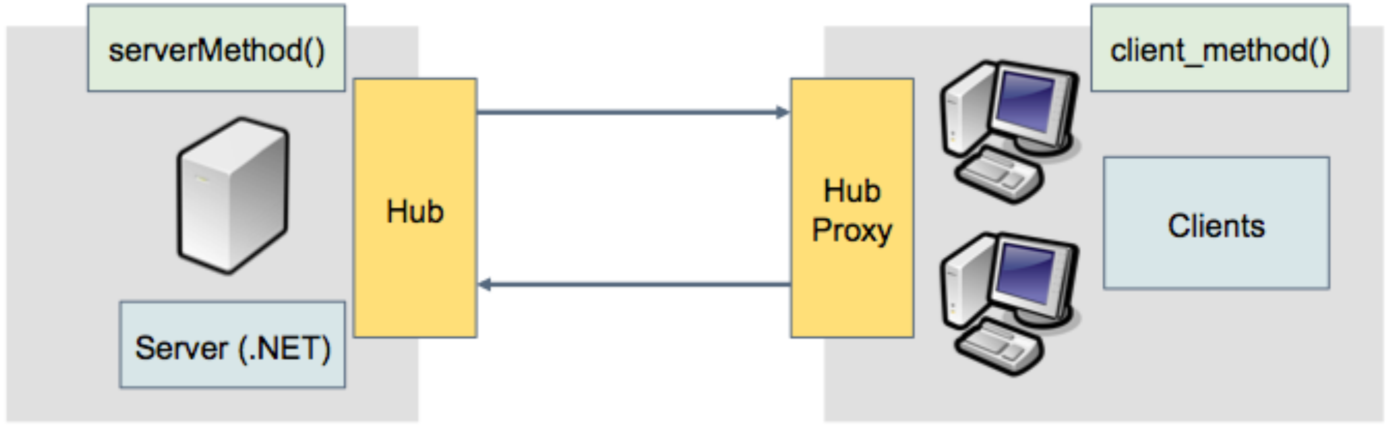


Observer


Notifies
Observer
Subject
Observer
Observer
Event happened

SignalR continued...

- SignalR provides a simple API for creating server-to-client **remote procedure calls (RPC)** that call JavaScript functions in client browsers (and other client platforms) from server-side .NET code.



.

SignalR

- SignalR handles connection **management automatically**, and lets you broadcast messages to all connected clients simultaneously, like a chat room.

- The connection between the client and server is persistent, unlike a classic HTTP connection, which is re-established for each communication.

- SignalR supports "server push" functionality, in which server code can call out to client code in the browser using Remote Procedure Calls (RPC), rather than the request-response model common on the web today.

- SignalR uses the new WebSocket transport where available, and falls back to older transports where necessary.

Connections and Hubs

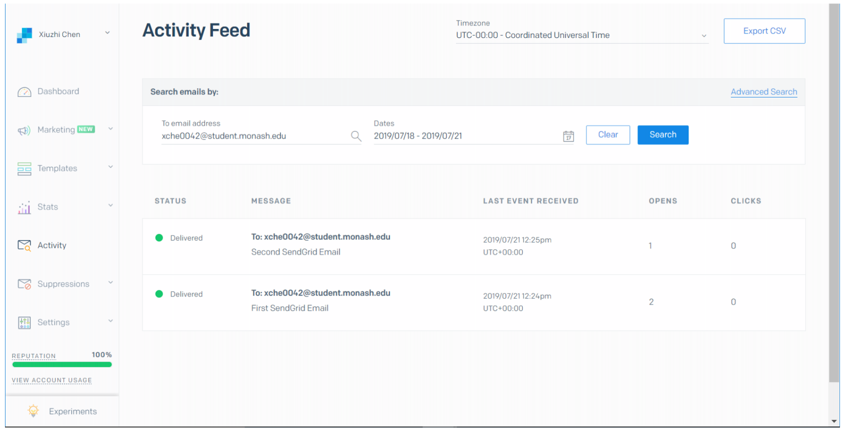
- The SignalR API contains two models for communicating between clients and servers: Persistent Connections and Hubs.
  - A Connection represents a simple endpoint for sending single-recipient, grouped, or broadcast messages.
  - A Hub is a more high-level pipeline built upon the Connection API that allows your client and server to call methods on each other directly.

Lecture Summary

- Recap: Security and Identity
- Sending Email with ASP.NET
- Accessing the Web Server File System
- SignalR

Week 8 Studio Overview

- Email using SendGrid
-



Activity Feed

Timezone
UTC-00:00 - Coordinated Universal Time

Search emails by:

To email address
xche0042@student.monash.edu

Dates
2019/07/18 - 2019/07/21

STATUS	MESSAGE	LAST EVENT RECEIVED	OPENS	CLICKS
● Delivered	To: xche0042@student.monash.edu Second SendGrid Email	2019/07/21 12:25pm UTC+00:00	1	0
● Delivered	To: xche0042@student.monash.edu First SendGrid Email	2019/07/21 12:24pm UTC+00:00	2	0

Next week: Web Optimisations & Evolution of ASP.NET CORE

- Web Optimisations & Evolution of ASP.NET CORE