MONASH University

FIT5032 - Internet Applications Development

Validation

Prepared by - Jian Liew
Updated by ABM Russel

Importance of Data Type

One of the more famous incidents is when the music video Gangnam Style hit the `INT_MAX` limit of YouTube where it used a 32-bit signed integer.

Because of this, when the music video hit `2,147,483,647` views, it stopped to increase.

When YouTube was first developed, **nobody ever imagined that a video would be watched more than 2 billion times, so the view count was stored using a signed 32-bit integer**.

What do you think the "signed" in the signed 32-bit integer means?

Do you think data types are important?

For example, what data type do you plan to use for storing money?

Regular Expressions (Regex)

- Regular expressions provide a **powerful, flexible, and efficient method** for processing text.
- The extensive pattern-matching notation of regular expressions enables you to quickly parse large amounts of text to
  - **find specific character patterns**
  - **to validate text to ensure that it matches a predefined pattern** (such as an email address)
  - **to extract, edit, replace, or delete text substrings; and to add the extracted strings to a collection in order to generate a report**
- For many applications that deal with strings or that parse large blocks of text, **regular expressions are an indispensable tool.**

How Regular Expressions Work

- The centerpiece of text processing with regular expressions is the **regular expression engine**.
- At a minimum, processing text using regular expressions requires that the regular expression engine be provided with the following two items of information:
  - The regular expression pattern to identify in the text.
  - The text to parse for the regular expression pattern.
- The methods of the Regex class let you perform the following operations:
  - Determine whether the regular expression **pattern occurs in the input text** by calling the Regex.IsMatch method.
  - **Retrieve one or all occurrences of text** that matches the regular expression pattern by calling the Regex.Match or Regex.Matches method
  - **Replace text that matches** the regular expression pattern by calling the Regex.Replace method

Common Regular Expression Syntax

Character	Meaning
^	Matches beginning of input. If the multiline flag is set to true, also matches immediately after a line break character.
$	Matches end of input. If the multiline flag is set to true, also matches immediately before a line break character.
*	Matches the preceding expression 0 or more times. Equivalent to {0,}.
+	Matches the preceding expression 1 or more times. Equivalent to {1,}.
?	Matches the preceding expression 0 or 1 time. Equivalent to {0,1}.
{n}	Matches exactly n occurrences of the preceding expression. N must be a positive integer.

Using Regular Expressions

Let's say you have to validate a String that is in the format of

example-1-example

Where it has the requirements of

- **(one or more alphabet)(optional dash symbol)(optional number)(optional dash symbol)(one or more alphabets)**
- It cannot begin with a dash or end with a dash and at least 2 characters in length

The regex would be

`^[A-Za-z]+\-{0,1}[0-9]*\-*[A-Za-z]+$`

```
^  → The start
[A-Z][a-z]+ → Must be more than 1 alphabet
\-{0,1} → Optional Dash
[0-9]* → May be a number
\-* → optional dash again
[A-Za-z]+ → Must be more than 1 alphabet
$ → the end
```

Challenge: Do you think you can write a shorter regex?

Results

```
Regular Expression = ^[A-Za-z]+\-*[0-9]*\-*[A-Za-z]+$
```

Input	Result
aa	true
a-a	true
a-1-a	true
aa-1-aa	true
a-1	false
1-a	false
11	false
aa-a	true
a1a	true
a--a	true

This is true due to the nature of the requirements. Do you think you can improve upon this regex?

Relevant XKCD

Perl is a popular scripting language that has often been referenced favorably in the comic. Perl is also the **most acknowledged language when it comes to the performance while evaluating regular expressions**.

Case Study for Regular Expressions

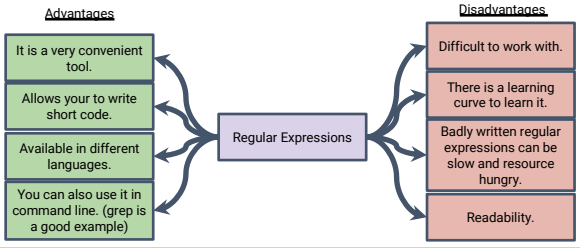
Can you think of a shorter way to write these regex?

Australian vehicle registration plate have different formatting schemes.

`a → alphabet, n → number, 1 → The number 1, Y → Starts with the alphabet "Y"`

State	Format	Regular Expression
Victoria	1aa.naa	`^1[A-Za-z]{2}\.[0-9][A-Za-z]{2}$`
Queensland	nnn.aaa	`^[0-9]{3}\.[A-Za-z]{3}$`
Australian Capital Territory	Yaa·nna	`^Y{1}[0-9]{2}\.[0-9]{2}[A-Za-z]$`
Western Australia	1aaa·nnn	It is not that hard to write this regex. Please try it yourself.

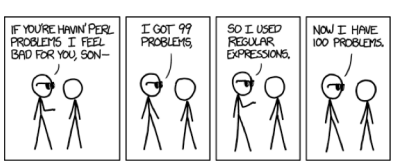
Regular Expressions Advantages & Disadvantages



General Tips When Using Regex

1. **Do not try to do everything in one regex.** Even though this is indeed possible, it should be avoided. There is nothing wrong with having different regexes.
2. **Get a regular expression tool**. Some of the recommended tools are RegexBuddy or https://regexr.com/
3. **Regular expressions are not Parsers**. If you are planning to parse a lot of text, at the end of the day it is better to write a custom parser.
4. **Use whitespace and comments.** Regex can be hard to read and understand, so if possible add comments whenever possible.
5. **Do NOT be afraid to use regular expressions.** It may seem confusing at first but over time you would be able to write less and do more with your code.

Relevant XKCD

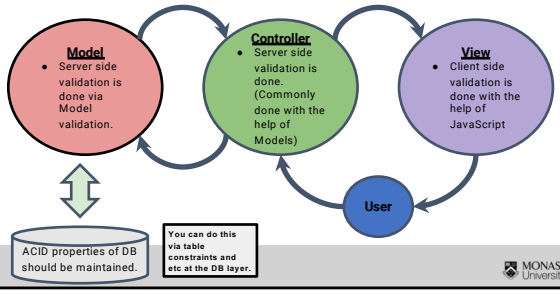


Knowing how to use Regular Expressions can be a good skill but you must understand when to use it so that it will not cause more problems.

Validation

- When we are talking about validation the first question that should and asked is "What are we validating?"
- One general rule of thumb is that **validation should be done for all layers**.
- This means that there should be validation in the **View layer, Control layer as well as the Model layer.**
- Even though this is a tedious and troublesome process, validation should be present at all layers.
- Besides that, there is also the question of how and where should we **validate certain business rules or logic?**

Overview of Validation



Why Form Validation is needed?

There are three main reasons

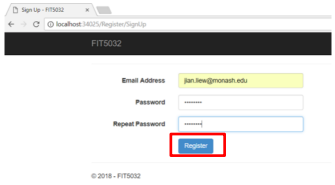
1. **We want to get the right data, in the right format** — our applications won't work properly if our user's data is stored in the incorrect format, if they don't enter the correct information, or omit information altogether.
2. **We want to protect our users' accounts** — by forcing our users to enter secure passwords, it makes it easier to protect their account information.
3. **We want to protect ourselves** — there are many ways that malicious users can misuse unprotected forms to damage the application.

Different Types of Form Validation

There are two different types of form validation which you'll encounter on the web:

1. **Client-side validation** is validation that occurs in the browser, before the data has been submitted to the server. This is more user-friendly than server-side validation as it gives an instant response.
2. **Server-side validation** is validation which occurs on the server, after the data has been submitted.

Client Side Validation



When there is client side validation present, hitting the "Register" button **does not instantly call the controller method** on the server, it calls the validation method on the client side (Browser) first.

When the client side validation passes, it will then call the controller method.

Server Side Validation

- Server-side code is used to validate the data before it is saved into the database.
- If the data fails authentication, a response is sent back to the client to tell the user what corrections to make.
- Server-side validation is not as user-friendly as client-side validation, as it does not provide errors **until the entire form has been submitted.**
- However, server-side validation is your application's last line of defense against incorrect or even malicious data.
- All popular server-side frameworks have features for validating and sanitizing data (making it safe).

Validation Attributes

- Validation attributes are a way to **configure model validation so it's similar conceptually to validation on fields in database tables.**
- This includes constraints such as assigning data types or required field.
- Other types of validation include applying patterns to data to enforce business rules, such as a **credit card, phone number, or email address**.
- Validation attributes make enforcing these requirements much simpler and easier to use.
- MVC supports any attribute that derives from ValidationAttribute for validation purposes.
- Many useful validation attributes can be found in the **System.ComponentModel.DataAnnotations** namespace.

Validating Model with Validation Attributes

```
public class Movie
{
    public int Id { get; set; }

    [Required]
    [StringLength(100)]
    public string Title { get; set; }

    [ClassicMovie(1960)]
    [DataType(DataType.Date)]
    public DateTime ReleaseDate { get; set; }

    [Required]
    [StringLength(1000)]
    public string Description { get; set; }

    [Range(0, 999.99)]
    public decimal Price { get; set; }

    [Required]
    public Genre Genre { get; set; }

    public bool Preorder { get; set; }
}
```

Conceptually similar to validations in the table of the database.

Popular Built-in Attribute Validation

Attribute	Functionality
`[Compare]`	Validates two properties in a model match.
`[EmailAddress]`	Validates the property has an email format.
`[Range]`	Validates the property value falls within the given range.
`[RegularExpression]`	Validates that the data matches the specified regular expression.
`[Required]`	Makes a property required.
`[StringLength]`	Validates that a string property has at most the given maximum length
`[Url]`	Validates the property has a URL format.
`[CreditCard]`	Validates the property has a credit card format.

The Required Attribute

- Non-nullable value types (such as decimal, int, float, and DateTime) are inherently required and don't need the Required attribute.
- MVC model binding, which isn't concerned with validation and validation attributes, rejects a form field submission containing a missing value or whitespace for a non-nullable type.
- In the absence of a `BindRequired` attribute on the target property, model binding ignores missing data for non-nullable types, where the form field is absent from the incoming form data.

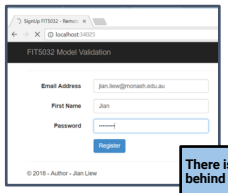
Custom Validation

- Validation attributes work for most validation needs.
- However, some validation rules are specific to your business.
- Your rules might not be common data validation techniques such as ensuring a field is required or that it conforms to a range of values. For these scenarios, custom validation attributes are a great solution.
- A good example is when you have a **business rule**. For example if you a given a scenario where a "User can only make four post per day on a forum board", you can create a custom validation for this scenario.
- Thus, you can actually create a Domain Model with these custom validation if you have business logic or rules that requires it.

Model Binding

- Model binding in ASP.NET MVC maps data from **HTTP requests to action method parameters.**
- The parameters may be simple types such as strings, integers, or floats, or they may be complex types.
- This is a great feature of MVC because mapping incoming data to a counterpart is an often repeated scenario, regardless of size or complexity of the data.
- MVC solves this problem by abstracting binding away so developers don't have to keep rewriting a slightly different version of that same code in every app

How Model Binding Works



POST/GET request

Model
- Each of the fields is binded to the Model.

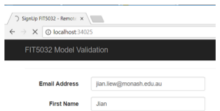
After the **POST** or **GET** request, there will be a "model" that is created based on the fields of the input form. In a way, it is like a new object.

There is a model behind every form.
When the "Register" button is selected, the input values are binded to the model.

Developers at times use the term **ViewModel** too..

Model State

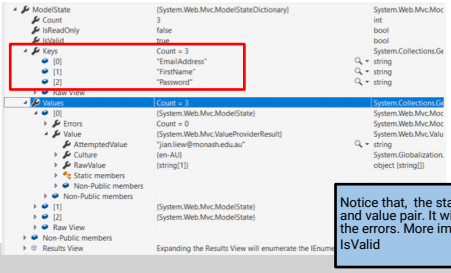
- The Model State is a very important concept for ASP.NET MVC Applications.
- ModelState is a property of a Controller, and can be accessed from those classes that inherit from System.Web.Mvc.Controller
- Model state represents validation errors in submitted HTML form values.



There should be a **Model or ViewModel** behind every form. In this form for example, there are 3 input values, these values are mapped as a Key Value Pair at the Model State.

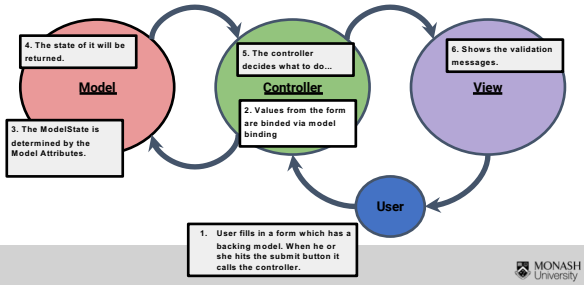
Once the "Register" button is selected, the model state will be known at server side.

Model State Explained...



Notice that, the state of it consist of a key and value pair. It will also keep a count of the errors. More importantly, there is an IsValid

Model Validation Summarized (Server Side)



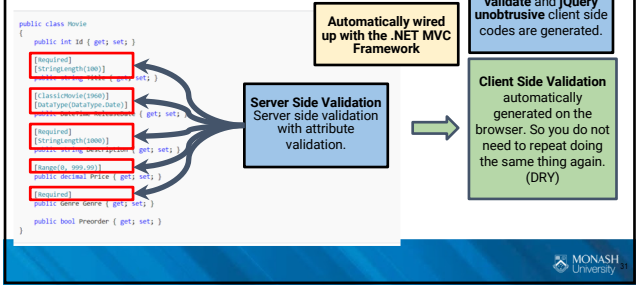
Client Side Validation

- **It saves time they would otherwise spend waiting for a round trip to the server**. Client side validation happens at the client side which is the browser.
- In business terms, even a few fractions of seconds multiplied hundreds of times each day adds up to be a lot of time, expense, and frustration.
- Straightforward and immediate validation enables users to work more efficiently and produce better quality input and output.
- The **jQuery Unobtrusive Validation** script is a custom Microsoft front-end library that builds on the popular jQuery Validate plugin.
- Without jQuery Unobtrusive Validation, you **would have to code the same validation logic in two places:** once in the server side validation attributes on model properties, and then again in client side scripts

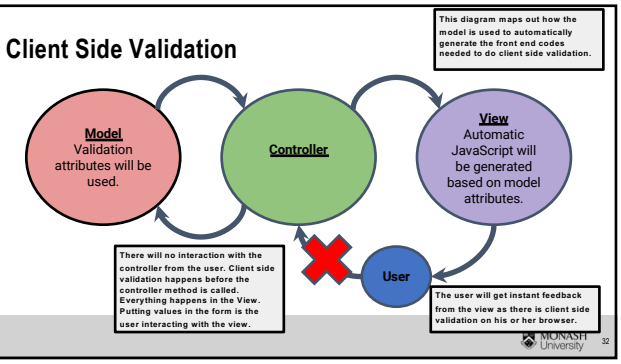
jQuery Unobtrusive Validation

- One of the more useful things MVC includes is Unobtrusive Validation with the combination of usage of the jQuery Validate plugin and the Unobtrusive library.
- This lightweight library allows us to add validation to our MVC views **without any additional client-side coding**.
- In order words, once you have included server side validation via validation attribute, the client side scripts to validate the forms will be **automatically generated.**
- However, the limiting factor of this, is that the auto generated codes does not support dynamically generated forms.

jQuery Unobtrusive Validation



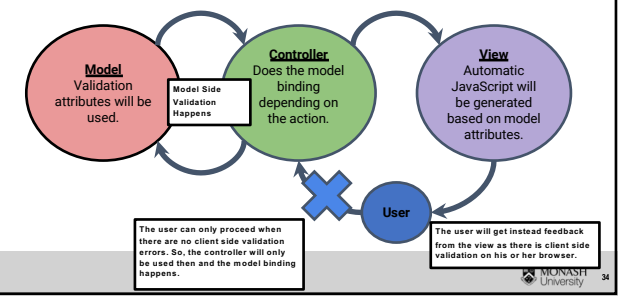
Client Side Validation



Importance of Client Side Validation

- The main reason client side validation is important is because it **significantly increases the user experience.**
- Instead of making round trips to the server, users can get an instant feedback.
- Client side validation will also in a way **offload the workload to the client** instead of the server.

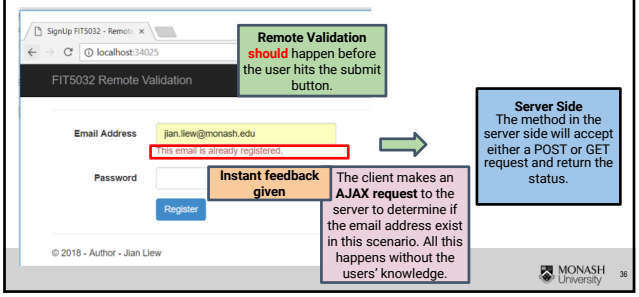
Server Side & Client Side Validation Combined



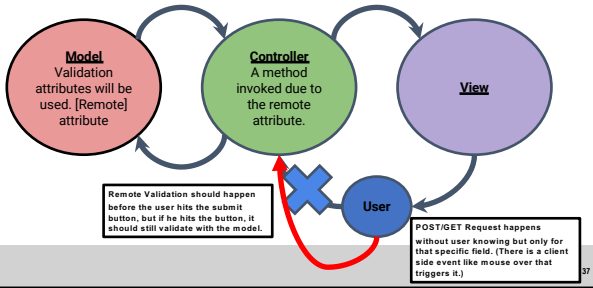
Remote Validation

- Remote validation is a great feature to use when you need to validate data on the client against data on the server.
- For example, your app may need to verify **whether an email or username is already in use**, and it must query a large amount of data to do so.
- Downloading large sets of data for validating one or a few fields consumes too many resources. It may also expose sensitive information. **An alternative is to make a round-trip request to validate a field however this decreases usability.**
- You can implement remote validation in a two step process. First, you must annotate your model with the `[Remote]` attribute. The `[Remote]` attribute accepts multiple overloads you can use to direct client side JavaScript to the appropriate code to call.
- With remote validation it is possible to give is **instant feedback** to the user if the username is in use. This will be shown in the tutorials.

Remote Validation Explained



Remote Validation



Relevant XKCD

There are times when **certain fields are considered to be "so" important** that developers would want you to type it in twice. This comic is an example of that.

The more common practice is to require the user to enter their **password twice during registration.**

Programmer Falsehoods

Ask a schoolchild to write down a date. **They will likely write a month, day of the month and maybe a year too (not necessarily in that order)**.

Ask a programmer to write a date and **they might write not just a date, but also add hours, minutes and seconds to it!**

But **a date does not have hours, minutes and seconds.** Most people know this. Except many programmers. Well they probably know, but many use the word date to mean a combination of date and point of time in hours and minutes (and more detailed than that).

This is an example of what we call **Programmer Falsehoods.**

Falsehoods regarding Time

All of these assumptions are wrong

1. There are always 24 hours in a day.
2. Months have either 30 or 31 days.
3. Years have 365 days.
4. February is always 28 days long.
5. Any 24-hour period will always begin and end in the same day (or week, or month).
6. A week always begins and ends in the same month.
7. A week (or a month) always begins and ends in the same year.
8. The machine that a program runs on will always be in the GMT time zone.
9. Ok, that's not true. But at least the time zone in which a program has to run will never change.
10. Well, surely there will never be a change to the time zone in which a program hast to run in production.
11. The system clock will always be set to the correct local time.

Handling Time

As a developer, you should understand (We need to store the date/time at the end of the day)

- **What is a Unix Timestamp?** Unix timestamp is a way to track time as a running total of seconds. This count starts at the Unix Epoch on January 1st, 1970 at UTC. It **does not change no matter** where you are located on the globe.
- **What is UTC Time?** UTC is the time standard commonly used across the world. The world's timing centers have agreed to keep their time scales closely synchronized - or coordinated - therefore the name Coordinated Universal Time. (This is also known as Zulu Military Time)
- **What is ISO 8601?** The purpose of this standard is to provide an unambiguous and well-defined method of representing dates and times, so as to **avoid misinterpretation of numeric representations of dates and times**, particularly when data are transferred between countries with different conventions for writing numeric dates and times.

What Date format should I use?

This decision will be left up to you. The general idea is, you must pick the correct format for the purpose of your application. However, it is always a good idea to capture more information than needed.

For example, if you are storing the opening hours of the library what date format do you plan to use?

At first look, you can easily store the opening hours in the format of a date. However, would this be the best way to do it? Of course you would then think you can store both the opening hours and closing hours. **But then, what if** the library has a different set of hours for special occasions? What if there is a situation where it has different opening hours in each week? What would the best way to store this information be? This then becomes a decision design on how to store this information.

Relevant XKCD

When abbreviating the date into numerical form, various areas of the world tend to list the year, month, and day in different orders (as well as with different delimiting symbols), which can cause confusion particularly when the day value is 12 or lower allowing it to be easily interpreted as the month and vice versa.

This comic states that there is in fact **one** international standard for writing numeric dates, set by the International Organization for Standardization in its ISO 8601 standard: **YYYY-MM-DD**.

Importance of Server Time

Use This Candy Crush Cheat To Get Unlimited Lives

In certain use cases, server time plays a vital role. It is important to understand that, the **client should never be trusted to provide a time.**

For example, if you are dealing with transactions in a site like eBay, the user should not be allowed to input the date in which the purchase happened. The server should be responsible of keeping track of this.

Falsehoods regarding Name

1. People have exactly one canonical full name.
2. People have exactly one full name which they go by.
3. People have, at this point in time, exactly one canonical full name.
4. People have, at this point in time, one full name which they go by.
5. People have exactly N names, for any value of N.
6. People's names fit within a certain defined amount of space.
7. People's names do not change.
8. People's names change, but only at a certain enumerated set of events.
9. People's names are written in ASCII.

Food for Thought.

How do you think Allocate+ or the Monash Systems handle the situation where you do not have a last name?

Falsehood about addresses

1. An address will start with, or at least include, a building number.
2. When there is a building number, it will be all-numeric.
3. An address will be comprised of road names

Business Logic?

Let's assume there is a scenario where……

- A customer can only make only a maximum of 2 purchases for a specific product. Where do you think this validation would go? Controller or Model?

There answer to this question, is this is up to you. Normally, you can introduce **domain logic layers** to handle this scenario, in fact it is quite common to introduce another layer to handle these validations. (So, if you are doing this, you are creating what is known as "fat" model design pattern) However, there is nothing wrong with having these business logic validation in the controller too. This would create something known as "fat" controllers.

Since, this unit **does not deal with the way you should architecture your application in detail, this decision will be left up to you.**