University of Glasgow

Web Application Development 2

Course admin

- Lecturer: Dr. Alistair Morrison
  - email: alistair.morrison@glasgow.ac.uk
- Lectures: Wed 1100-1200 and Fri 1100-1200, both in Sir Charles Wilson (201 LT)
- Lab sessions: in Islay / Jura labs of the Main University Library each week, including this week!
- Moodle page: https://moodle.gla.ac.uk/course/view.php?id=5728

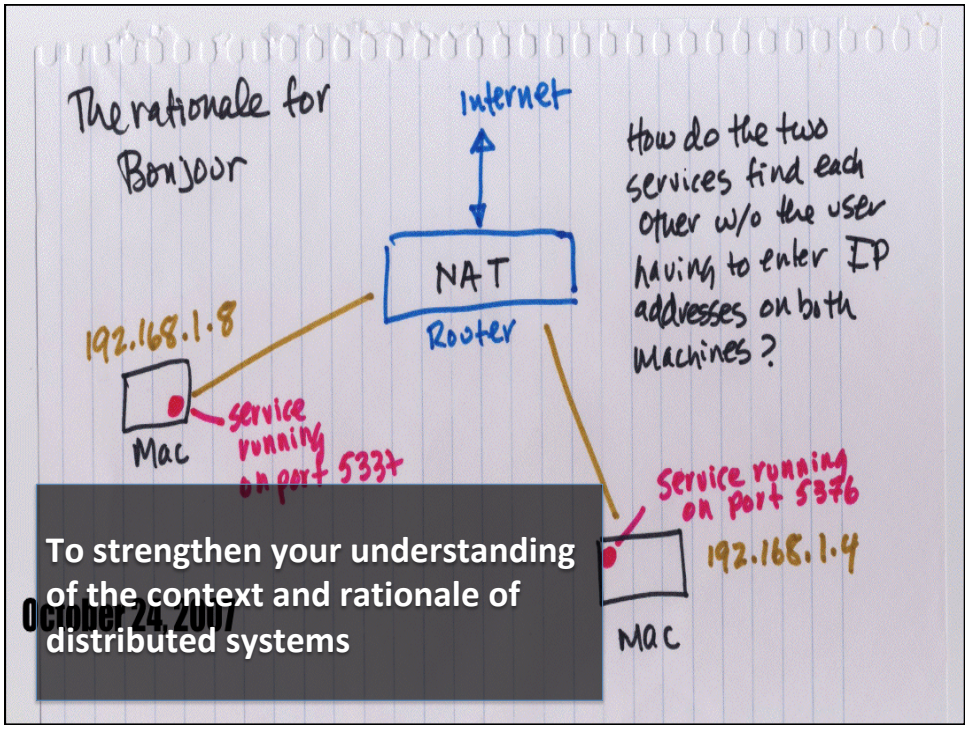
What is a Web App?

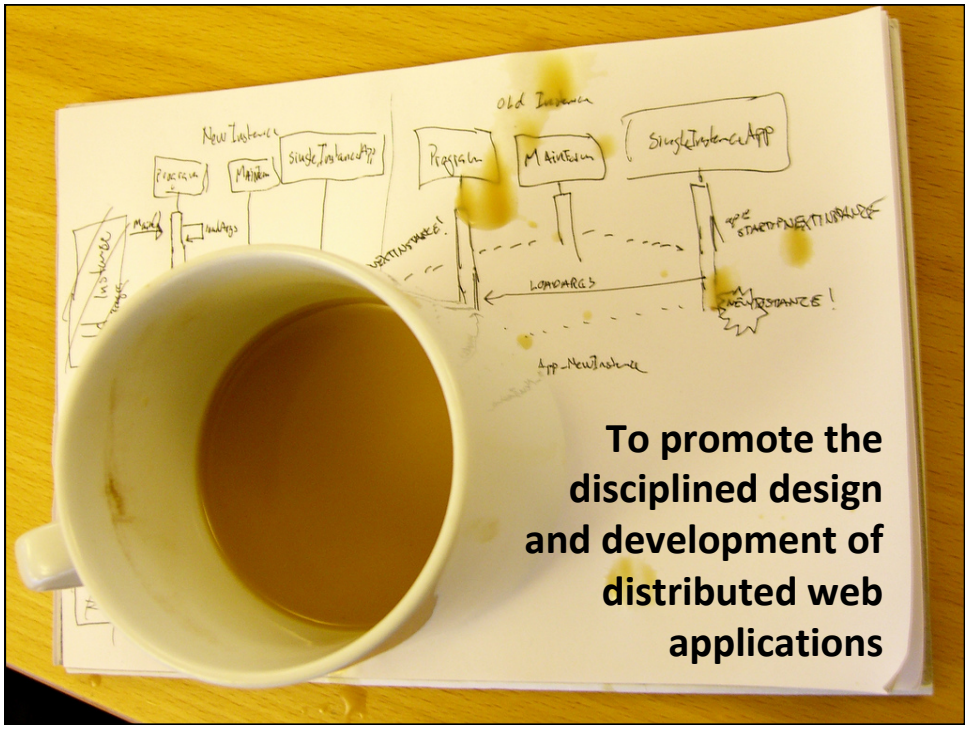
- It is a Distributed Information Management system (DIM)
- Enables the management, sharing, finding, modification and presentation of information
- Does so over a network, in a distributed fashion
- Typically has many users, often geographically separated
- Ideally DIMs enable users to access timely, relevant and useful information in a seamless manner
- Think MyCampus ☺

Examples

COURSE AIMS


To provide an overview of the tools and technologies used in web development
WAVERLEIGH
YOUNGSON
BINNSMEAD
MARYSVILLE
BOYLE
FOSTER
LENTS
ARLETA
LLEWELLYN
LEWIS
LANE
WOODMERE
CLINTON KELLY
DUNIWAY
NORTH SELLWOOD
ERROL HEIGHTS
HARNEY
WHITMAN
SELLWOOD
BATTIN
ARDENWALD
WICHITA
BELL
MILWAUKIE


The rationale for Bonjour
Internet
NAT
Router
How do the two services find each other w/o the user having to enter IP addresses on both machines?
192.168.1.8
Mac
service running on port 5337
Service running on Port 5376
192.168.1.4
Mac
To strengthen your understanding of the context and rationale of distributed systems
October 24, 2007


To promote the disciplined design and development of distributed web applications


To understand the messaging and protocols used as communication mechanisms in web applications


To develop the ability to implement and deploy distributed web applications

Types of Architecture

System Architecture

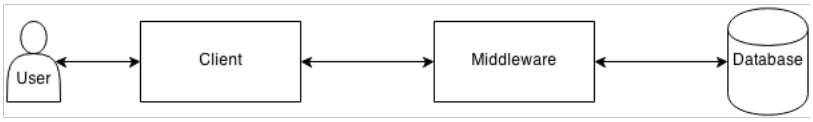
Information Architecture

Design Elements

- System Architecture (system focused)
  - Specifications and Requirements
  - High Level System Architecture
  - Entity Relationship Diagrams
  - Sequence Diagrams
- Information Architecture (user focused)
  - User Personas
  - User needs matrix
  - Site design / URL design
  - Wireframes and Walkthroughs

SYSTEM COMPONENTS

High Level System Architecture



User and Client

- The **User** could be human or machine, which
  - Initiates contact and interacts with the client
  - Ranges in skills and abilities
  - Has a number of requirements that need to be satisfied
- The **Client** is a program sitting on a client device which
  - sends **request messages**
  - accepts **response messages**
  - acts on the message
    - either communicating to the user
    - or affecting the environment in some way

Messaging

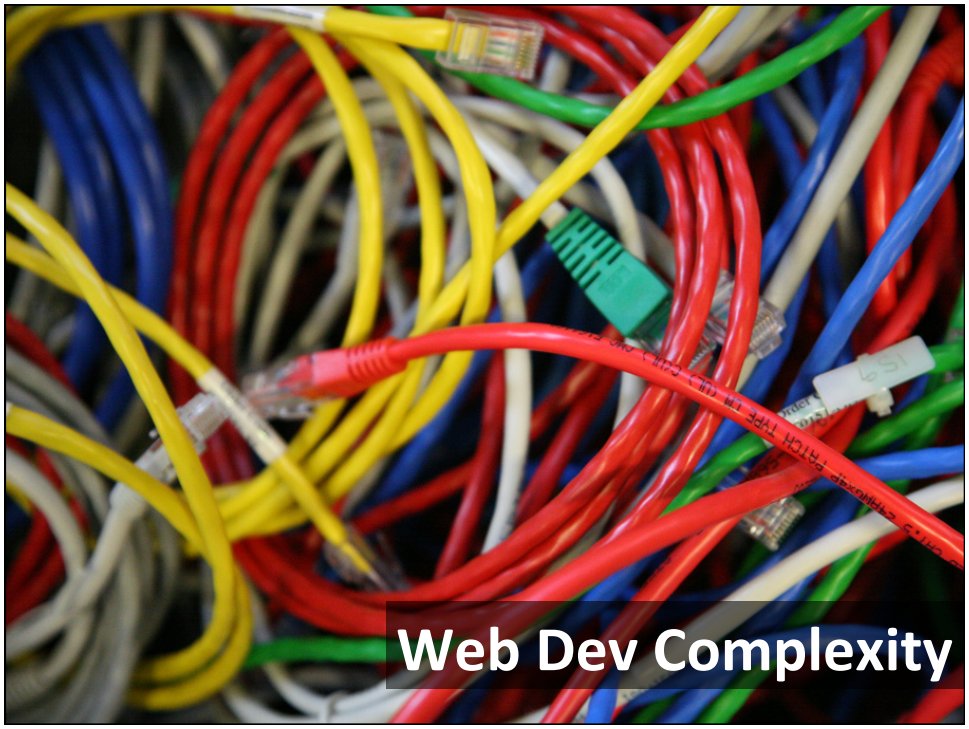
- The **request message** is sent from the client to the server to:
  - ask for some information
  - send some information to be stored
    - either from user input
    - or from some device (sensor)
- The **response message** is sent from the server to a client to:
  - return the requested information
  - effect some change in the environment

Messaging Protocols

- The **request message protocol**
  - is usually an HTTP request
  - embedding any data to be sent
- The **response message protocol**
  - is an HTTP response
  - with content that is often in JSON/XML

Middleware & Backends

- The **Middleware/Application Server** is the central component which
  - accepts **request messages** from clients
  - returns **response messages**
  - co-ordinates the application components
- The **Backend/Database** is typically on a separate node and
  - stores the data for the application
  - provides the data when needed
  - needs to be scalable and reliable
  - could be a database, an index, a flat file

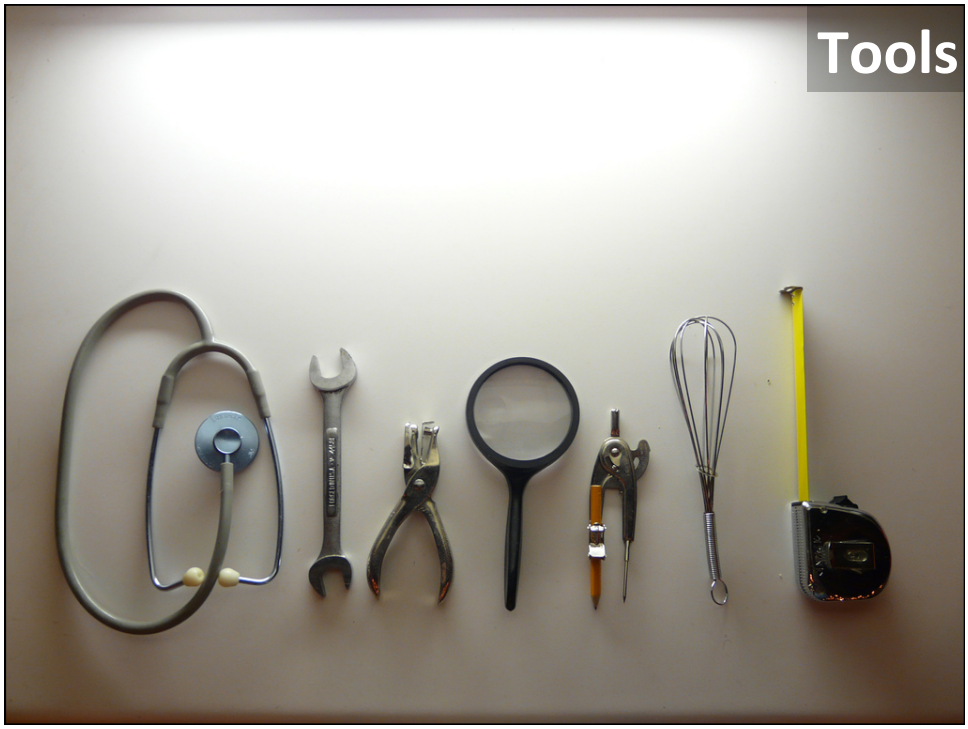


Web Development Complexity

- **Collision of Languages**
  - Markup Languages
  - Programming Languages
  - Database Query Languages
- **Shifting Standards**
  - Document Object Model
  - XML/JSON
- **Web Browser Compatibility**
  - Browser wars encourage new 'unique' features
- **HTTP is a Stateless Protocol**
  - But most applications require the persistence of state

Signs of Hope

- **Web development has become a serious business**
  - Despite all these challenges, there are many incentives to persevere
- **In line with this, the methods of development are maturing and increasingly adopting good practices of 'classical' Software Engineering:**
  - Application Programmer Interfaces (APIs)
  - Libraries
  - Frameworks
  - Tools
  - Standards



Web Development Tools

- **The nature of web development is disjoint**
  - a developer must become familiar with a set of distinct and (typically) non-integrated tools
- **Web development tool support is not yet as advanced as with 'classic' software development**
  - Most languages have several complex IDEs (Integrated Development Environments) to choose from
    - IDLE, Eclipse, PyCharm, etc

What you code in…

- An important tool is the text editor or IDE
- The choice of text editor and your expertise in its usage can seriously affect your productivity
  - Syntax Awareness
  - Auto-completion
  - Snippets
  - Scripts & Macros
  - Integration with other development tools (revision control)
- Some IDEs have plugins for scripting languages (e.g., Eclipse has a PHP plug-in)

Checking your code

- Interpreted languages lack the compilation stage where errors and warnings can be raised
- Frustratingly, scripts will just run until they reach an error and fall over
  - If you are lucky, there will be an error message
    - But it won't make much sense most of the time
- There are a wide range of tools that will perform static analysis of scripts to spot errors
  - PyLint / JSLint
  - PHP_CodeSniffer (works on PHP, Javascript & CSS)
  - Chrome's Developer Tools, Firebug for Firefox

Understanding W3C standards

- Each aspect of the software is specified by a W3C (https://www.w3.org/) **activity** run by a working group which produces the following reports:
  - **working draft** - no consensus yet
  - **candidate recommendation** - published in order to gather implementation experience and feedback
  - **proposed recommendation** - sent for final approval by advisory committee
  - **w3c recommendation** - approved by the W3C
- Note that this means everything evolves (and gets out of date) very fast

Complying with Standards

- There are a variety of tools to check that certain parts of a web application conforms to standards
  - (X)HTML
    - http://validator.w3.org/
  - CSS
    - http://jigsaw.w3.org/css-validator/
  - XML Feeds (RSS/Atom)
    - http://validator.w3.org/feed/
  - JavaScript Code Quality Tool
    - http://www.jslint.com/

Main Technologies in this Course

- Python
- Django
- HTML
- CSS
- HTTP GET/POST
- XML/JSON
- XHTML
- JavaScript
- JQuery
- AJAX

- As well as:
  - GitHub
  - Pip
  - PythonAnywhere
  - IDLE
  - Virtual Environments
  - Draw.io

Lots to Learn

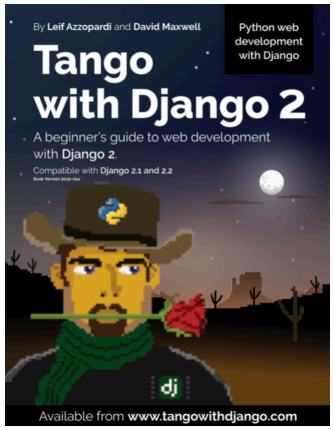
- **Developing Web Applications requires the use of many different languages, technologies, protocols, standards, and formats.**
- **You will be expected to be proficient in each of these technologies**
- **You will need to understand how the different components fit together**
- **You will need to answer code-based questions in the exam**

Assessments

	Deadlines
• (1) Lab Exercises (10%)	
– development of the Rango application	31 Jan & 14 Feb
• (2) Group Project (40%)	
– Design specification (10%)	28 Feb
– Project presentation (5%)	26/27 Mar
– Project application (25%)	27 Mar
• (3) In-class quizzes (10%)	
– Multiple-choice questions	
• (4) Exam (40%)	
– Multiple-choice questions	
– Includes numerous code-based questions	

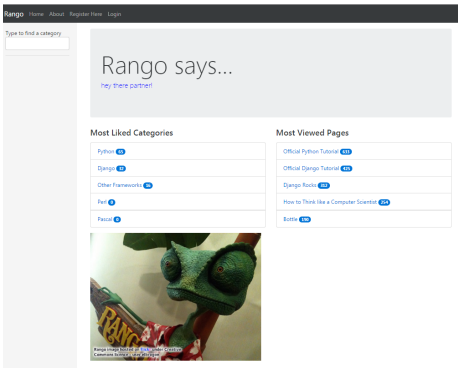
Rango application

- Main focus of the lab sessions in weeks 1-5
- Developed in Python and Django using "Tango with Django" (updated for v2.1 & 2.2), by Leif Azzopardi and David Maxwell
  - Version **2020-01a**
- Obtain the book via course Moodle page



Rango application (cont)

- To be developed individually
- Will be marked using automated tests
- Checkpoint 1: 31 Jan – 10% of marks
- Checkpoint 2: 14 Feb – 90% of marks



WAD2 Project

- In small groups, you will be responsible for the design, development and deployment of a web application using Python / Django
- This will be the main focus of lab sessions 6-11
- Choice of what to build is up to you
  - but it will have to be designed well!
  - design specification assesses this
- You will present your application to your lab group on 26 / 27 March
- And submit the code by 27 March

In-class quizzes

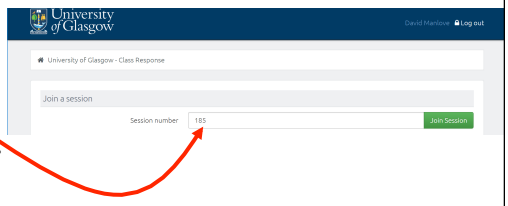
- Lecture quizzes will aim to reinforce concepts, promote interaction, feedback
- Not about attendance monitoring
- But, strong correlation between attendance and performance in the final exam
- Typically one or two quizzes per lecture
- Designed to be simple (and unobtrusive)
- Can only be answered during scheduled classes
- Similar to types of questions asked in the final exam

In-class quizzes (cont)

- Correct answer: 1 mark; incorrect / missing answer: 0 marks
- Best 70% of marks for the quizzes will be aggregated to give a band
- Quiz questions will not be available retrospectively or on Moodle
- Responses should be made by smartphone / laptop / tablet, via Wi-Fi / 4G
- If ill, see the course handbook (available on Moodle)

Quiz: dry run (unassessed!)

- Via your smartphone, laptop or tablet, connect to the internet using Wi-Fi (e.g., eduroam) or 3G/4G
- Navigate to the YACRS system:
  https://classresponse.gla.ac.uk
- Login using your GUID (e.g., 2029999z)
- Enter session number 578 under "join a session"



- You should see 4 choices for Q1: A, B, C and D
- Select one as your answer to the question on the next slide…

Quiz Question 0

How many of the following 10 technologies have you used before?

- Python
- Django
- HTML
- CSS
- HTTP GET/POST
- XML/JSON
- XHTML
- JavaScript
- JQuery
- AJAX

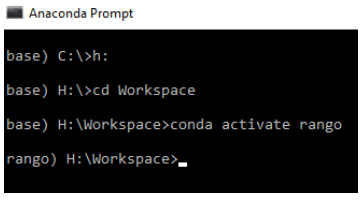
A. 0-2
B. 3-5
C. 6-8
D. 9-10

Reminder:

- URL: https://classresponse.gla.ac.uk
- Login using your GUID (e.g., 2029999z)
- Enter session 578 under “join a session”

Command-Line Interfaces

- Control software or an Operating System by issuing text-based commands
- Alternative to Graphical User Interface (GUI)
  - An OS might have both, and you can choose which to use
- Can be used to perform many common tasks
  - Navigate around directory (folder) structure
  - Create/edit files
  - Run applications
  - Lots more!



Command-Line Interfaces

- We'll be using Command Line a lot in labs
  - Anaconda Prompt
- Common commands:
  - **dir** list files in current directory
  - **mkdir <name>** create new directory called 'name'
  - **cd <name>** change directory/navigate to named destination
  - **cd ..** move 'up' one level to current directory's parent
- Many Django/Anaconda/Git-specific commands
  - see future lectures

QUICK INTRO TO HTML

What is HTML?

- HTML stands for **HyperText Markup Language**
- It's the language web browsers use to interpret what gets displayed when you view a web page
- A mark-up language is a set of tags which describe document content
- Hyperlinks are connections between documents
- HTML documents (web pages) contain HTML mark-up tags and plain text

Basic HTML Example

```
<html>      ← Begin HTML now
<head>
<title> The title </title>
</head>
<body>

<h1>WAD2</h1>

<p>My first paragraph.</p>

</body>
</html>     ← End HTML now
```

Tags:

- Keywords (tag names) surrounded by <>
- Normally have opening and closing tags

Plain text:

- Between tags
- Are the content displayed in the browser

Basic HTML Example

```
<html>
  <head>
    <title> The title</title>
  </head>
  <body>

    <h1>WAD2</h1>

    <p>My first paragraph.</p>

  </body>
</html>
```

HTML Document Structure:

- Nested tags
- Starts with <html> tag
- <head> tag contains information about the document such as title and other things
- <body> tag contains the html to be displayed

Basic HTML Example

```
<html>
  <head>
    <title>The title</title>
  </head>
  <body>

    <h1>Things to do</h1>

    <p>My first paragraph.</p>

  </body>
</html>
```

Elements

- From an opening tag to a closing tag is called an element
- The plain text between opening and closing tags is called the element content

Basic HTML Example

<!DOCTYPE html>

```
<html>
    <head>
      <title>The title</title>
    </head>
    <body>
      <h1>Things to Do</h1>
       Make an HTML page<br>
       Add a paragraph
      <p>My first paragraph.</p>
    </body>
</html>
```

Empty Elements

- There are some tags which have no content
- They also have no end tag
- E.g.
 which forces a line break

Basic HTML Example

<!DOCTYPE html>

```
<html>
    <head>
      <title> The title</title>
    </head>
    <body>

      <h1>Things to To</h1>

      <p>My first paragraph.</p>

    </body>
</html>
```

Two Basic Tags:

- <h1> - "header 1"
  - Used just once
  - Defines the most important heading
  - Search engines use H1 to determine the content of your web pages
  - There are h1,…,h6 headers. H1 being the most important

Basic HTML Example

```
<html>
    <head>
      <title>The title</title>
    </head>
    <body>

      <h1>WAD2</h1>

      <p>My first paragraph.</p>

    </body>
</html>
```

Two Basic Tags:

- <p> is the paragraph tag
- Browsers add space (margin) before and after each <p> element
- They ignore your own formatting – collapse whitespace

Other useful HTML tags (1)

- Anchor tags – provide HTML hyperlinks

  Syntax:

  link text

  Example:

  Visit the WAD2 Moodle page

- Unordered list / list items

```
<ul>
    <li> List item one </li>
    <li> List item two </li>
</ul>
```

Visit the WAD2 Moodle page

- List item one
- List item two

Other useful HTML tags (2)

- Div elements let you create sections to divide up the page in different ways when coupled with CSS

- Span elements are used to group inline-elements in a document, again when coupled with CSS

  Example: I have

  blue eyes.

  I have blue eyes.

Lab session for this week

- Download the Lab Briefing Sheet for Weeks 1-5 from the course Moodle page
- Work your way through the sheet
- Make sure you finish the section “Lab session for week 1” this week
- Move onto the next section (“Lab sessions for weeks 2-5”) if you finish early
- Important to set up Git repo properly!
  - Read the appendix on Git from the course text
- Your lab group tutor will be there if you have any questions

Lab groups

- Note your lab group number, time and venue from MyCampus and tutor from the table below

Lab Grp	Lab Time	Venue	Tutor
1	Thu 1700-1800	Jura	Joshua Mitton
2		Islay	Michael McKay
3	Fri 0900-1000	Jura	Tom Wallis
4		Islay	Adrian Ramsingh
5	Fri 1000-1100	Jura	Joshua Mitton
6		Islay	Michael McKay
7	Fri 1500-1600	Jura	Francesco Perrone
8		Islay	Pierre Ezard
9	Fri 1600-1700	Jura	Adrian Ramsingh
10		Islay	Francesco Perrone
11	Fri 1700-1800	Jura	Adrian Ramsingh
12		Islay	Francesco Perrone
13	Fri 1400-1500	Jura	Alistair Morrison