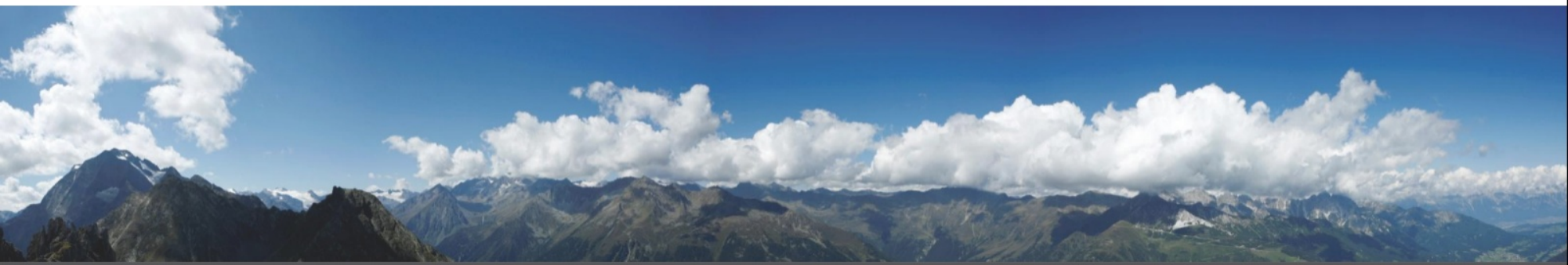
STI · INNSBRUCK

703657 PS/2 Web Engineering

Task 5 - Testing and Usability

Wednesday, 2015-04-29

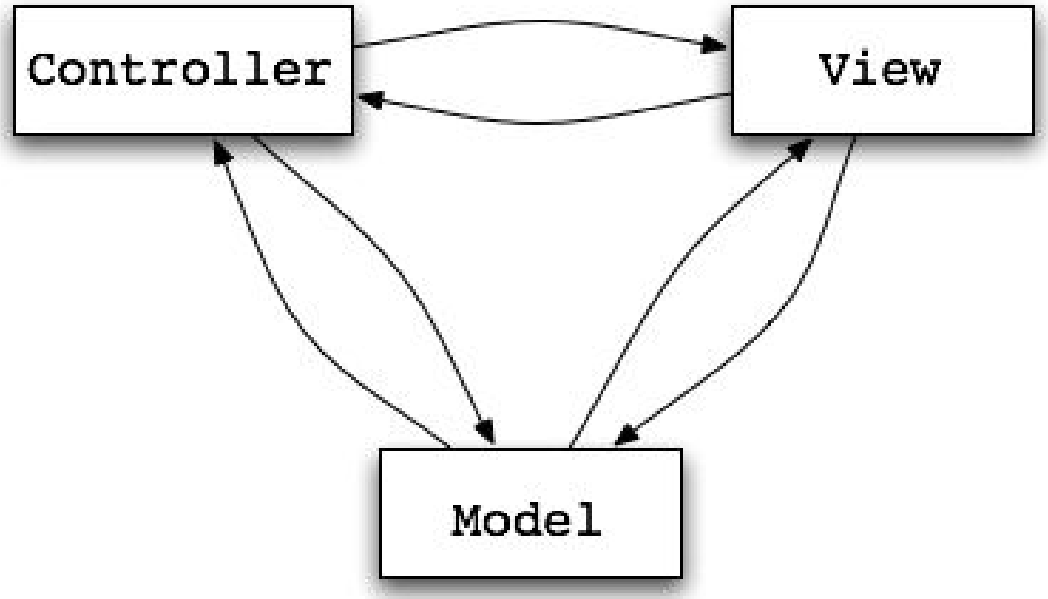


Schedule - Where are we now?

STI · INNSBRUCK

Session	Date	Task
1	Wed, 2015-03-11	Project idea
2	Wed, 2015-03-18	Requirements engineering
3	Wed, 2015-03-25	Web Application Modeling
4	Wed, 2015-04-15	Progress / Presentation
5	Wed, 2015-04-22	Web Application Modeling II
6	**Wed, 2015-04-29**	**Testing and Usability**
7	Wed, 2015-05-06	Mid-Term Report / Presentation
8	Wed, 2015-05-13	-
9	Wed, 2015-05-20	-
10	Wed, 2015-05-27	-
11	Wed, 2015-06-03	Progress / Presentation
12	Wed, 2015-06-10	-
13	Wed, 2015-06-17	-
14	Wed, 2015-06-24	Final Report / Presentation

Recap (1) - Model-View-Controller



Separate the concerns of the application: modelling, presentation and actions.

1. Model, deals with the data
2. View, deals with the visualization of the data
3. Controller, deals with the data flow

Model-View-Controller - Frameworks

STI · INNSBRUCK

Java:

i. Apache Struts (`https://struts.apache.org/`)
ii. Google Web Toolkit (`http://www.gwtproject.org/`)
iii. JavaServer Faces (`https://javaserverfaces.java.net/`)
iv. Spring (`https://spring.io/`)

JavaScript:

i. AngularJS (`http://angularjs.org/`)
ii. Backbone.js (`http://backbonejs.org/`)
iii. Ember.js (`http://emberjs.com/`)
iv. Ext JS (`http://www.sencha.com/products/extjs/`)
v. Kendo UI (`http://www.kendoui.com/`)

Model-View-Controller - Frameworks

STI · INNSBRUCK

- Comparison of web application frameworks (`http://en.wikipedia.org/wiki/Comparison_of_web_application_frameworks`)
- Notes:
  - By now, you should be able to determine which framework is suitable for your project

Recap (2) - Testing and Usability

- Testing: Functional requirements validation
- Usability: Non-functional requirements validation

1. Unit tests, testing the ”atomic” units - classes, Web pages, etc. - independently. (Developer)
2. Integration tests, test the interaction of units (Tester & Developer)
3. System tests, testing the whole, integrated system (Dedicated team)
4. Acceptance tests, ”Real-world” tests - testing under conditions that are as close to the ”live” environment as possible (Client)
5. Beta tests, informal, product-wide tests conducted by ”friendly” users.

Testing Tool

STI · INNSBRUCK

i. JUnit.org (`http://www.junit.org`)
ii. TestNG (`http://testng.org/`)
iii. Spring TestContext Framework (`https://spring.io/`)
iv. Python Unittest (`https://docs.python.org/2/library/unittest.html`)
v. Robot Framework (`http://robotframework.org/`)
vi. PHPUnit (`https://phpunit.de/`)
vii. Canoo WebTest (`http://webtest.canoo.com`)
viii. Selenium (`http://www.seleniumhq.org/`)
ix. Ruby Unit Test Frameworks (`https://www.ruby-toolbox.com/categories/testing_frameworks`)

Usability Tool

STI · INNSBRUCK

i. http://chrispederick.com/work/web-developer/
ii. http://webusability.com/usability-tools/
iii. http://wave.webaim.org/

Evaluator / Validator:

- Web Accessibility Evaluation Tools List (http://www.w3.org/WAI/ER/tools/)
- Quality Assurance Tools (http://www.w3.org/QA/Tools/)
  1. The MarkUp Validator (http://validator.w3.org/)
  2. The Link Checker (http://validator.w3.org/checklink)
  3. The CSS Validator (http://jigsaw.w3.org/css-validator/)
  4. The Mobile-friendliness Checker (https://validator.w3.org/mobile/)

Example - Unit Test

STI · INNSBRUCK

- Is a piece of code written by a developer that executes a specific functionality in the code to be tested
- Targets a small unit of code, e.g., a method or a class
- Unit tests ensure that code works as intended.

JUnit [1] - a unit testing framework for the Java programming language.

```
@Test //Annotation
public void testIt() {
  Book book = new Book(...);

  //Test
  assertEquals("N24229", book.getId());
  assertEquals("Web␣Engineering", book.getTitle());
}
```

[1] http://junit.org/

JUnit - Annotations

STI · INNSBRUCK

1. **@Test** - Identifies a method as a test method.
2. **@Test (expected = Exception.class)** - Fails if the method does not throw the named exception.
3. **@Test (timeout=100)** - Fails if the method takes longer than 100 milliseconds.
4. **@Before** - This method is executed before each test.
5. **@After** - This method is executed after each test.
6. **@BeforeClass** - This method is executed once, before the start of all tests.
7. **@AfterClass** - This method is executed once, after all tests have been finished.
8. **@Ignore** - Ignores the test method.

JUnit - Test Methods

STI · INNSBRUCK

1. **assertTrue**([message], boolean condition)
2. **assertFalse**([message], boolean condition)
3. **assertEquals**([String message], expected, actual)
4. **assertEquals**([String message], expected, actual, tolerance)
5. **assertNull**([message], object)
6. **assertNotNull**([message], object)
7. **assertSame**([String], expected, actual)
8. **assertNotSame**([String], expected, actual)
9. **fail**(String)

JUnit - Example

STI · INNSBRUCK

```
@Test
public void testCRUD() throws Exception {
  DAOFactory daoFactory = DAOFactory.getDAOFactory();
  BookDAO bookDAO = daoFactory.getBookDAO();

  String bookId = "N24229";
  assertNull(bookDAO.findBook(bookId));

  bookDAO.insertBook(bookId, "Web␣Engineer");
  assertNotNull(bookDAO.findBook(bookId));

  bookDAO.updateBook(bookId, "Web␣Engineering");
  assertEquals("Web␣Engineering",
      bookDAO.findBook(bookId).getTitle());

  bookDAO.deleteBook(bookId);
  assertNull(bookDAO.findBook(bookId));
}
```

JUnit - Hamcrest

STI · INNSBRUCK

Library of matchers for building test expressions[2]

1. **Core**: anything, describedAs, is
2. **Logical**: allOf, anyOf, not
3. **Object**: equalTo, hasToString, instanceOf, isCompatibleType, notNullValue, nullValue, sameInstance
4. **Beans**: hasProperty
5. **Collections**: array, hasEntry, hasKey, hasValue, hasItem, hasItems, hasItemInArray
6. **Number**: closeTo, greaterThan, greaterThanOrEqualTo, lessThan, lessThanOrEqualTo
7. **Text**: equalToIgnoringCase, equalToIgnoringWhiteSpace, containsString, endsWith, startsWith

[2] http://hamcrest.org/

JUnit - Hamcrest

STI · INNSBRUCK

```
import static org.hamcrest.MatcherAssert.assertThat;
import static org.hamcrest.Matchers.*;
import junit.framework.TestCase;

public class BookTest extends TestCase {

  public void testIt() {
    Book book = new Book("N24229", "Web␣Engineering");

    Book dbBook = bookDAO.findBook("N24229");

    assertThat(book, equalTo(dbBook));
    assertThat(dbBook.getTitle(), startsWith("Web"));
    assertThat(dbBook.getTitle(),
        equalToIgnoringCase(book.getTitle()));
  }
}
```

Task 5 - Project Design Report

STI · INNSBRUCK

Group assignment:

1. Prepare your project design report, use the provided template.
   i) Project Model obtained from WebML
      a) Domain Model
      b) Hypertext/Navigation Model
      c) Presentation Model
   ii) Application Architectures
      - Determine a framework for your application
   iii) Testing and Usability
      - Determine a unit testing framework for your application
      - Create a plan to test a CRUD operation of your applications
2. Send the files to the tutor by the next session (Wednesday, 2015-05-06, 08.15) at the latest

2. Progress / Presentation

STI · INNSBRUCK

Next session (Wednesday, 2015-05-06):

1. Present your project design to the class
   - For each group, allocated time is 8 minutes + 2 minutes for Questions and Answers
   - Try to distribute your slides to all members of your group equally