Agile Software Development

Produced by

Eamonn de Leastar (edeleastar@wit.ie)

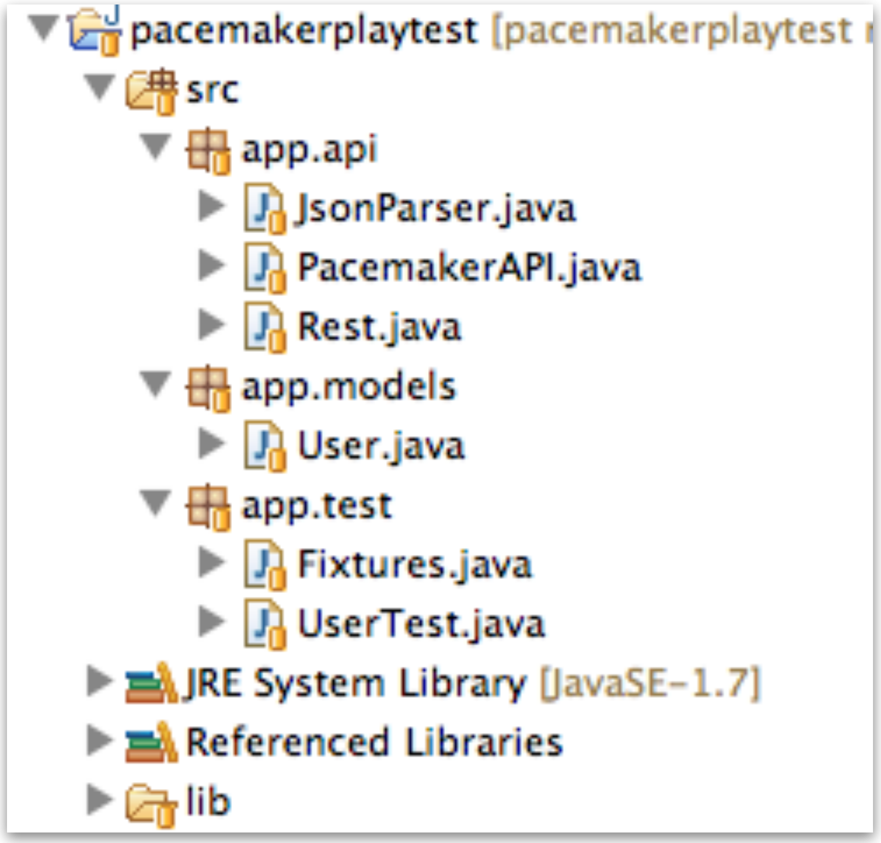
Department of Computing, Maths & Physics
Waterford Institute of Technology

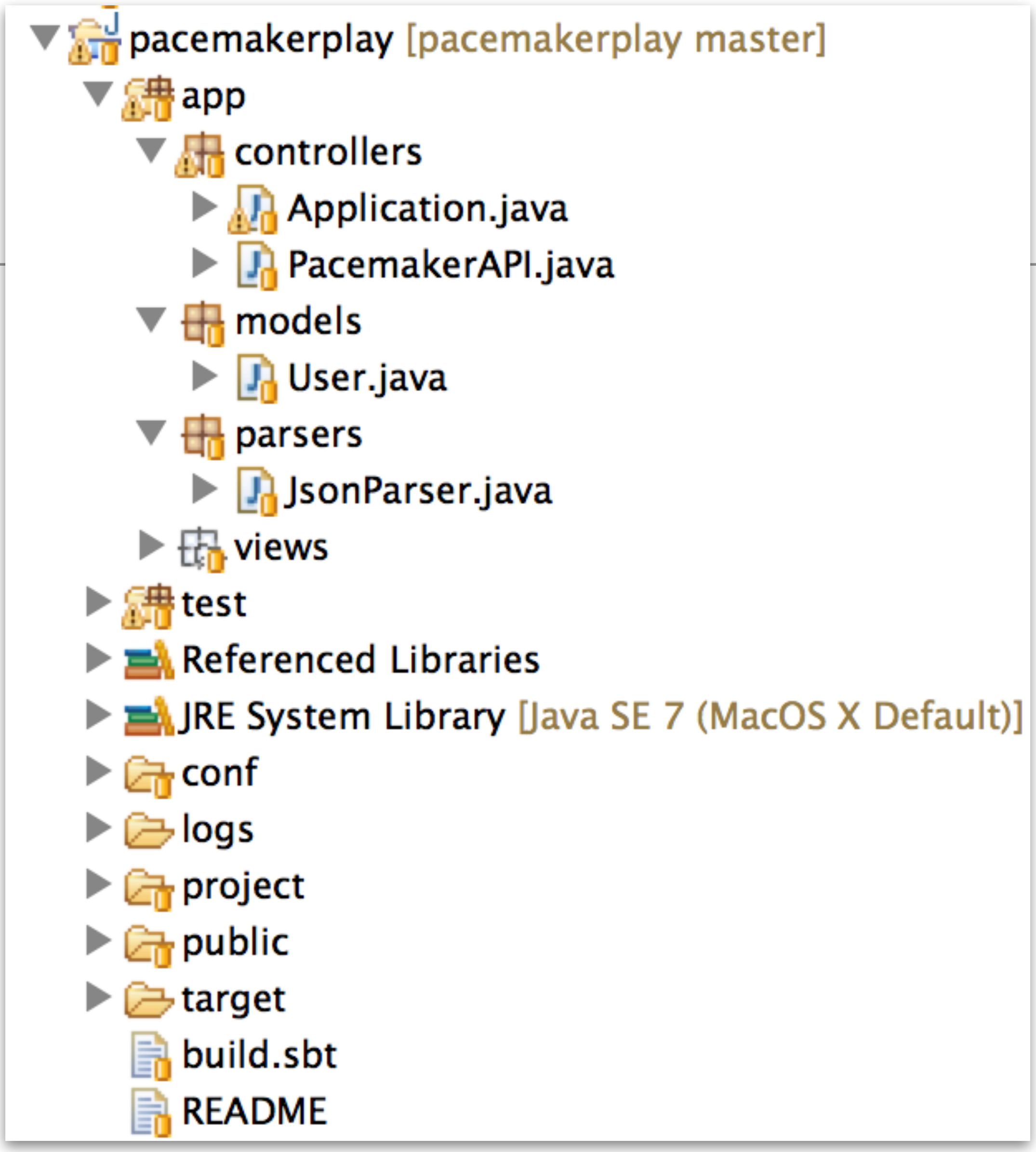
http://www.wit.ie

http://elearning.wit.ie

Testing Pacemakerplay

Two Projects


pacemakerplaytest [pacemakerplaytest
- src
  - app.api
    - JsonParser.java
    - PacemakerAPI.java
    - Rest.java
  - app.models
    - User.java
  - app.test
    - Fixtures.java
    - UserTest.java
- JRE System Library [JavaSE-1.7]
- Referenced Libraries
- lib

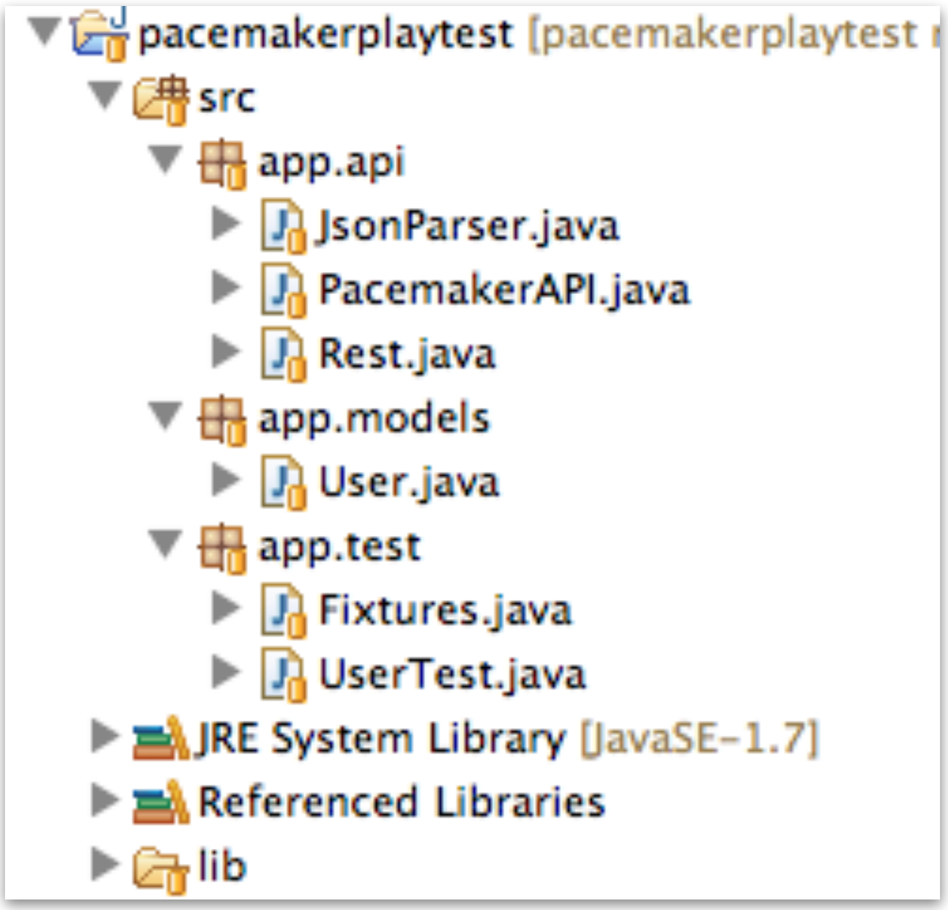

pacemakerplay [pacemakerplay master]
- app
  - controllers
    - Application.java
    - PacemakerAPI.java
  - models
    - User.java
  - parsers
    - JsonParser.java
  - views
- test
- Referenced Libraries
- JRE System Library [Java SE 7 (MacOS X Default)]
- conf
- logs
- project
- public
- target
- build.sbt
- README

- System Under Test (SUT)

- Test Project

pacemakerplaytest

- Test application runs as a separate process (may be on a different machine).
- Tests written using standard JUnit conventions
- Exercises pacemakerplay over http - as it is indented to be used.
- Considerably expanded scope of the tests:
  - the model
  - the model’s Object Relational Mapping (ORM) to the database (+ evolutions?)
  - the ‘business logic’ in the server
  - the exposure of the API over Restful http
- + security? Performance? etc…



```
pacemakerplaytest [pacemakerplaytest
  src
    app.api
      JsonParser.java
      PacemakerAPI.java
      Rest.java
    app.models
      User.java
    app.test
      Fixtures.java
      UserTest.java
  JRE System Library [JavaSE-1.7]
  Referenced Libraries
  lib
```

pacemakerplaytest - models

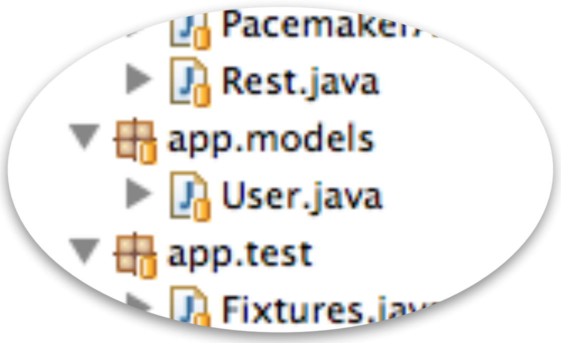
- Model classes are ‘shadowed’ in test project

```
@Entity
@Table(name="my_user")
public class User extends Model
{
  @Id
  @GeneratedValue
  public Long   id;
  public String firstname;
  public String lastname;
  public String email;
  public String password;
  public String nationality;
  …
}
```

play

```
public class User
{
  public Long   id;
  public String firstname;
  public String lastname;
  public String email;
  public String password;



 …
}
```

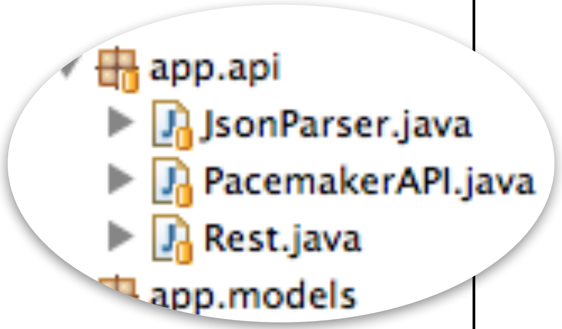
test



pacemakerplaytest - api

- Encapsulate the API into a single class
- Class exposes Json and Model variants of API
- Uses same JsonParser class as pacemaker play
- Use Rest class to make blocking calls to server
- Rely on Exceptions to convey errors

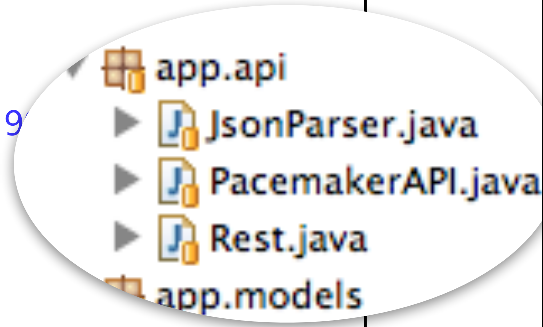
```java
public class PacemakerAPI
{
  public static List<User> getUsers () throws Exception
  {
    String response =  Rest.get("/api/users");
    List<User> userList = renderUsers(response);
    return userList;
  }

  public static User createUser(String userJson) throws Exception
  {
    String response = Rest.post ("/api/users", userJson);
    return renderUser(response);
  }

  public static User createUser(User user) throws Exception
  {
    return createUser(renderUser(user));
  }

  public static User getUser(Long id) throws Exception
  {
    String response = Rest.get ("/api/users/" + id);;
    User user = renderUser(response);
    return user;
  }
  public static void deleteUsers() throws Exception
  {
    Rest.delete("/api/users");
  }
  public static void deleteUser(Long userId) throws Exception
  {
   Rest.delete("/api/users/" + userId );
  }

  public static void updateUser(Long userId, String userJson) throws Exception
  {
    Rest.put("/api/users/" + userId, userJson);
  }

  public static void updateUser(Long userId, User user) throws Exception
  {
    Rest.put("/api/users/" + userId, renderUser(user));
  }
}
```



pacemakerplaytest - api

- Make http requests, assuming Json payloads.
- Block until response
- Generate exceptions on failure
- Uses apache httpcomponent library (compatible with android)

```
public class Rest
{
  private static DefaultHttpClient httpClient = null;
  private static final String URL = "http://localhost:9

  private static DefaultHttpClient httpClient()
  {
    if (httpClient == null)
    {
      HttpParams httpParameters = new BasicHttpParams();
      HttpConnectionParams.setConnectionTimeout(httpParameters, 10000);
      HttpConnectionParams.setSoTimeout(httpParameters, 10000);
      httpClient = new DefaultHttpClient(httpParameters);
    }
    return httpClient;
  }

  public static String get(String path) throws Exception
  {
    HttpGet getRequest = new HttpGet(URL + path);
    getRequest.setHeader("accept", "application/json");
    HttpResponse response = httpClient().execute(getRequest);
    return new BasicResponseHandler().handleResponse(response);
  }

  public static String delete(String path) throws Exception
  {
    HttpDelete deleteRequest = new HttpDelete(URL + path);
    HttpResponse response = httpClient().execute(deleteRequest);
    return new BasicResponseHandler().handleResponse(response);
  }

  public static String post(String path, String json) throws Exception
  {
    HttpPost putRequest = new HttpPost(URL + path);
    putRequest.setHeader("Content-type", "application/json");
    putRequest.setHeader("accept", "application/json");

    StringEntity s = new StringEntity(json);
    s.setContentEncoding("UTF-8");
    s.setContentType("application/json");
    putRequest.setEntity(s);

    HttpResponse response = httpClient().execute(putRequest);
    return new BasicResponseHandler().handleResponse(response);
  }
}
```

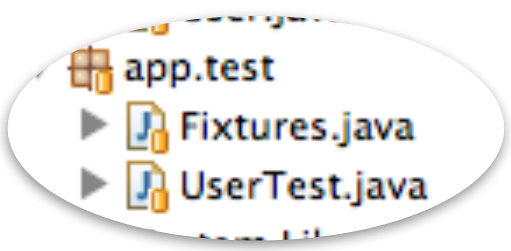


pacemakerplaytest - api

- Filter Json output to specifically exclude ‘class’ metadata in serialised form

```
public class JsonParser
{
  private static JSONSerializer  userSerializer = new JSONSerializer().exclude("class");

  public static User renderUser(String json)
  {
    return new JSONDeserializer<User>().deserialize(json, User.class);
  }

  public static String renderUser(Object obj)
  {
    return userSerializer.serialize(obj);
  }

  public static List<User> renderUsers(String json)
  {
    return new JSONDeserializer<ArrayList<User>>().use("values", User.class).deserialize(json);
  }
}
```

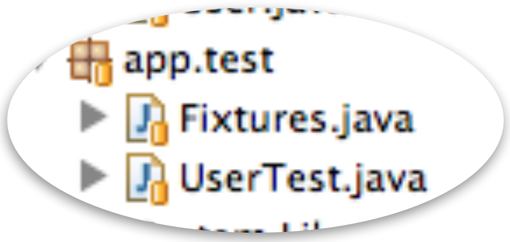
pacemakerplaytest - test



- Similar to pacemaker-1.0 tests
- Extra fixture to test json serializer

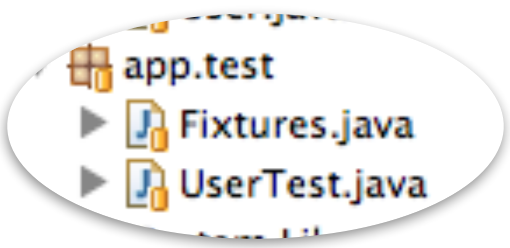
```
public class Fixtures
{
  static String userJson = "{\n"
                                 +  "\"email\"    : \"jim@simpson.com\" ,\n"
                                 +  "\"firstName\": \"Jim\"             ,\n"
                                 +  "\"lastName\" : \"Simpson\"         ,\n"
                                 +  "\"password\" : \"secret\"           \n"
                            + "}";

  static User users[] = {
                            new User ("homer",  "simpson", "homer@simpson.com",  "secret"),
                            new User ("lisa",   "simpson", "lisa@simpson.com",   "secret"),
                            new User ("maggie", "simpson", "maggie@simpson.com", "secret"),
                            new User ("bart",   "simpson", "bart@simpson.com",   "secret"),
                            new User ("marge",  "simpson", "marge@simpson.com",  "secret"),
                          };
}
```

pacemakerplaytest - test


app.test
Fixtures.java
UserTest.java

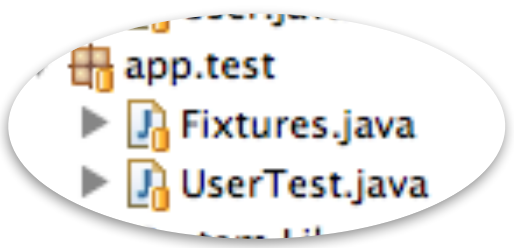
```
public class UserTest
{
  static User users[] =
  {
    new User ("homer",  "simpson", "homer@simpson.com",  "secret"),
    new User ("lisa",   "simpson", "lisa@simpson.com",   "secret"),
    new User ("maggie", "simpson", "maggie@simpson.com", "secret"),
    new User ("bart",   "simpson", "bart@simpson.com",   "secret"),
    new User ("marge",  "simpson", "marge@simpson.com",  "secret"),
 };

  User user;

  @Before
  public void setUp() throws Exception
  {
    user = new User ("mark", "simpson", "mark@simpson.com", "secret");
    PacemakerAPI.deleteUsers();
  }

  @After
  public void tearDown() throws Exception
  {
    PacemakerAPI.deleteUsers();
  }

  @Test
  public void createUserJson() throws Exception
  {
    User user1 = PacemakerAPI.createUser(Fixtures.userJson);
    User user2 = PacemakerAPI.getUser(user1.id);
    assertEquals(user1, user2);
    PacemakerAPI.deleteUser(user1.id);
  }

  @Test
  public void createUserObj() throws Exception
  {
    User user2 = PacemakerAPI.createUser(user);

    assertTrue(user.equals(user2));
    PacemakerAPI.deleteUser(user2.id);
  }
```

pacemakerplaytest - test



```
@Test
public void createUserObjs() throws Exception
{
  for (User user : Fixtures.users)
  {
    User user2 = PacemakerAPI.createUser(user);
    user.id = user2.id;
  }

  List <User> users = PacemakerAPI.getUsers();
  assertEquals(users.size(), Fixtures.users.length);

  for (User user : Fixtures.users)
  {
    PacemakerAPI.deleteUser(user.id);
  }
  List <User> users2 = PacemakerAPI.getUsers();
  assertEquals(0, users2.size());
}

@Test
public void updateUser() throws Exception
{
  User user2 = PacemakerAPI.createUser(user);
  user2.email = "NEWNAME@simpson.com";
  PacemakerAPI.updateUser(user2.id, user2);

  User user3 = PacemakerAPI.getUser(user2.id);
  assertEquals (user3.email, "NEWNAME@simpson.com");
  assertEquals (user3.id, user2.id);

  PacemakerAPI.deleteUser(user2.id);
}
```

pacemakerplaytest - test



```
  @Test
  public void updateNonExistantUser() throws Exception
  {
    try
    {
      Rest.put("/api/users/4000", Fixtures.userJson);
      fail ("put error");
    }
    catch(HttpResponseException e)
    {
      assertTrue (404 == e.getStatusCode());
    }
  }

  @Test
  public void deleteeNonExistantUser() throws Exception
  {
    try
    {
      Rest.delete("/api/users/4000");
      fail ("delete error");
    }
    catch(HttpResponseException e)
    {
      assertTrue (404 == e.getStatusCode());
    }
  }
}
```

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

eLearning support unit