Kampus/KATBook Integration with Moodle/Canvas/BlackBoard etc.

Draft Document: Version 0.1 Dated 24/03/2023

Steps to add an external web page in Moodle:

In Moodle login as Admin or Manager or Teacher.

Select the course.

Enable the edit mode and click the Add an activity or resources.

Select "**External Tool**" from the "**Add an activity or resource"**

What is External Tool:

Many content providers share materials and interactive learning exercises different from and complementary to Moodle's own resources and activities. The external tool offers a way for teachers to link to these activities from within their Moodle course page and where available to have grades sent back into Moodle. Students only need to log in to Moodle; they do not have to log in a second time to the connecting content site.

Content sites which allow connection to Moodle in this way are known as LTI compliant and are called tools.

The External tool enables participants to interact with LTI-compliant learning resources and activities on other web sites. (LTI is an IMS standard for Learning Tool Interoperability.) For example, an external tool could provide access to a new activity type or learning materials from a publisher.

To create an external tool activity, a tool provider which supports LTI (Learning Tools Interoperability) is required. A teacher can create an external tool activity or make use of a tool configured by the site administrator.

External tool activities differ from URL resources in a few ways:

- External tools are context aware i.e. they have access to information about the user who launched the tool, such as institution, course and name
- External tools support reading, updating, and deleting grades associated with the activity instance
- External tool configurations create a trust relationship between your site and the tool provider, allowing secure communication between them

How to setup an External Tool in Moodle:

1 In a course, with the editing turned on, choose 'External tool' from the activity chooser.

2 Give it a name and, if needed, a description.
3 Add the tool URL, either one configured by your sits admin or create your own external tool activity, for which you need the Consumer key and Shared secret, both provided by the external tool provider.
4 Specify the Launch container (how the external tool will display)
5 Select the services you want.
6 Specify the privacy settings.
7 Save the changes.

Moodle Help pages to explore:

External tool
https://docs.moodle.org/401/en/External_tool

External tool settings
https://docs.moodle.org/401/en/External_tool_settings

LTI Advantage Automatic Registration
https://moodlelti.theedtech.dev/dynreg/

Using External tool
https://docs.moodle.org/401/en/Using_External_tool

LTI Specification and Sample Code

LTI stands for Learning Tools Interoperability, which is a specification developed by the IMS Global Learning Consortium to facilitate the integration of learning tools with learning management systems (LMS).

Learning Tool Interoperability (LTI) is a set of specifications that allow learning tools to be integrated with learning management systems (LMS) or other education platforms. Developing LTI for a product involves creating a set of interfaces and protocols that allow the product to interact with different LMSs and other education platforms that support LTI.

With LTI, instructors can easily integrate third-party tools such as quizzes, assignments, and simulations into their LMS courses without the need for additional logins or manual gradebook entries.

Here are some of the key features of LTI:

Single Sign-On: LTI enables users to log in to their LMS and access third-party learning tools without the need for separate usernames and passwords.

Resource Sharing: LTI allows LMS users to easily share content, data, and resources with third-party tools.

Gradebook Integration: LTI supports the integration of third-party learning tools with the LMS gradebook, making it easy for instructors to manage grades and track student progress.

Security: LTI provides secure access to third-party tools and ensures the protection of user data.

The architecture of an LTI-compliant product typically includes the following components:

1. LTI Provider: This component provides the interface for integrating with other platforms, such as an LMS. It should be able to authenticate users and communicate with the LMS using the LTI protocol.
2. LTI Consumer: This component consumes the services provided by the LTI Provider. It should be able to launch the LTI Provider in a new window and pass user credentials to it.
3. LTI Launch URL: This is the URL that the LTI Consumer uses to launch the LTI Provider. It includes the necessary parameters to authenticate the user and establish the connection.
4. LTI Advantage: This is a set of extensions to the basic LTI specification that provide additional functionality, such as improved security and user management.
5. Outcome Service: The outcome service is a protocol that allows the learning tool to send student performance data back to the LMS.

To convert an existing solution to LTI, the following things are required:

1. Understand the LTI specification: You need to have a good understanding of the LTI specification and how it works.

2. Identify the LTI components: You need to identify the LTI components that are required for your solution and integrate them into your product.
3. Implement LTI Provider: You need to implement the LTI Provider component in your product. This involves creating an interface that can communicate with the LMS using the LTI protocol.
4. Implement LTI Consumer: You need to implement the LTI Consumer component in your product. This involves creating a way to launch the LTI Provider and pass user credentials to it.
5. Implement LTI Launch URL: You need to implement the LTI Launch URL in your product. This involves creating a URL that includes the necessary parameters to authenticate the user and establish the connection.
6. Implement LTI Advantage: If you want to take advantage of the additional functionality provided by LTI Advantage, you need to implement the necessary extensions in your product.
7. Test and validate: You need to test and validate your LTI integration to ensure that it works correctly and is compatible with different LMSs and other education platforms that support LTI.

Here is some sample code in Java for developing an application with the LTI specification:

1 Implementing LTI Provider:

```
import org.imsglobal.lti.launch.LtiVerificationResult;
import org.imsglobal.lti.launch.LtiVerifier;
import org.imsglobal.lti.launch.LtiVerificationException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class MyLtiProvider {

  public void handleLaunch(HttpServletRequest request,
HttpServletResponse response) throws Exception {
    // Get the LTI parameters from the request
    Map<String, String> parameters = new HashMap<>();
    Enumeration<String> parameterNames =
request.getParameterNames();
    while (parameterNames.hasMoreElements()) {
      String name = parameterNames.nextElement();
      String value = request.getParameter(name);
      parameters.put(name, value);
    }

    // Verify the LTI launch request
    LtiVerifier ltiVerifier = new LtiVerifier();
```

```
    LtiVerificationResult verificationResult =
ltiVerifier.verify(request, parameters);

    if (!verificationResult.getSuccess()) {
      throw new LtiVerificationException("LTI launch
verification failed");
    }

    // Process the LTI launch request
    // ...
  }
}
```

2 Implementing LTI Consumer:

```
public class MyLtiConsumer {

  public void launchLtiProvider(String launchUrl, String key,
String secret, String userId) throws Exception {
    // Construct the LTI launch request
    Map<String, String> parameters = new HashMap<>();
    parameters.put("lti_version", "LTI-1p0");
    parameters.put("lti_message_type", "basic-lti-launch-
request");
    parameters.put("lti_consumer_key", key);
    parameters.put("lti_user_id", userId);
    // Add additional LTI parameters as needed

    // Sign the LTI launch request
    LtiSigner signer = new LtiSigner();
    String signature = signer.sign(parameters, launchUrl,
"POST", secret);

    // Add the signature to the LTI parameters
    parameters.put("oauth_signature", signature);

    // Launch the LTI provider
    String launchQueryString =
LtiUtils.convertMapToQueryString(parameters);
    String launchUrlWithQueryString = launchUrl + "?" +
launchQueryString;
    // Open the launchUrlWithQueryString in a new window or
iframe
  }
}
```

3 Implementing LTI Launch URL:

```
public class MyLtiLaunchUrl {

  public String generateLaunchUrl(String baseUrl, String key,
String secret, String userId) throws Exception {
    // Construct the LTI launch URL
    String launchUrl = baseUrl + "/lti-launch";
    Map<String, String> parameters = new HashMap<>();
    parameters.put("lti_version", "LTI-1p0");
    parameters.put("lti_message_type", "basic-lti-launch-
request");
    parameters.put("lti_consumer_key", key);
    parameters.put("lti_user_id", userId);
    // Add additional LTI parameters as needed

    // Sign the LTI launch URL
    LtiSigner signer = new LtiSigner();
    String signature = signer.signUrl(launchUrl, "POST",
parameters, secret);

    // Add the signature to the LTI launch URL
    String launchQueryString =
LtiUtils.convertMapToQueryString(parameters);
    String launchUrlWithQueryString = launchUrl + "?" +
launchQueryString + "&oauth_signature=" + signature;

    return launchUrlWithQueryString;
  }
}
```

4 Implementing LTI Advantage:

```
public class MyLtiAdvantage {

  public void handleDeepLinkingRequest(HttpServletRequest
request, HttpServletResponse response) throws Exception {
    // Get the LTI Advantage JWT
```

Here are some additional resources that may be helpful for implementing LTI in Java:

IMS Global Learning Consortium LTI Developer Resources: This page provides links to the LTI Advantage specifications and other resources for developers.

LTI Advantage Test Tool: This tool allows you to test your LTI Advantage implementation against a set of test cases.

Java SDK for LTI Advantage: This is a Java SDK for implementing LTI Advantage APIs, developed by the Apereo Foundation.

Learning Tools Interoperability Core Specification 1.3

https://www.imsglobal.org/spec/lti/v1p3

IMS Global - LTI Utilities - Git Repo

https://github.com/1EdTech/basiclti-util-java

Here is the source code for org.imsglobal.lti.toolProvider.ContentItem.java

http://www.java2s.com/example/java-src/pkg/org/imsglobal/lti/toolprovider/contentitem-0abe7.html

Recipe for Making LTI 1 Tool Providers

https://www.imsglobal.org/recipe-making-lti-1-tool-providers

Sample LTI Java Web Application:

Here is a simple Java web application that demonstrates the LTI components based on the LTI specification. This application implements a basic "Hello World" message exchange between a Tool Provider and a Tool Consumer using LTI 1.3.

First, you will need to add the LTI 1.3 libraries to your project dependencies. You can do this by adding the following Maven dependencies to your project's pom.xml file:

```
<dependency>
    <groupId>org.imsglobal</groupId>
    <artifactId>lti-tool-provider-api</artifactId>
    <version>1.3.4</version>
</dependency>

<dependency>
    <groupId>org.imsglobal</groupId>
    <artifactId>lti-tool-consumer-api</artifactId>
    <version>1.3.4</version>
</dependency>
```

Next, create a simple Java class that represents the Tool Provider. This class will handle the launch request from the Tool Consumer and respond with a “Hello World” message. Here’s an example:

```
import org.imsglobal.lti.toolProvider.*;
import org.imsglobal.lti.launch.*;

public class HelloWorldProvider implements ToolProvider {

    public void onLaunch(LtiLaunch ltiLaunch) {
        String message = "Hello, " + ltiLaunch.getUserName() +
"!";
        ltiLaunch.getResponse().setParameter("message",
message);
    }

}
```

In this example, the HelloWorldProvider class implements the ToolProvider interface from the LTI Tool Provider API. The onLaunch method is called when the Tool Consumer sends a launch request to the Tool Provider. The method constructs a "Hello World" message using the user's name from the launch parameters, and sets the message as a response parameter.

Next, create a simple Java class that represents the Tool Consumer. This class will send a launch request to the Tool Provider and display the response. Here's an example:

```
import org.imsglobal.lti.toolConsumer.*;
import org.imsglobal.lti.launch.*;
public class HelloWorldConsumer {
    public static void main(String[] args) {
        ToolConsumer consumer = new ToolConsumerImpl();
        LtiLaunch ltiLaunch =
consumer.getLtiLaunch("http://localhost:8080/launch");
        String message =
ltiLaunch.getResponse().getParameter("message");
        System.out.println(message);
    }
}
```

In this example, the HelloWorldConsumer class creates a ToolConsumerImpl object from the LTI Tool Consumer API. The getLtiLaunch method sends a launch request to the Tool Provider at the specified URL (http://localhost:8080/launch). The response from the Tool Provider is stored in the ltiLaunch object. Finally, the message parameter from the response is retrieved and displayed in the console.

To run this example, you will need to deploy the Tool Provider application to a web server and start it up. Then, you can run the Tool Consumer application from the command line, and it will send a launch request to the Tool Provider and display the "Hello World" message.

Note that this is a very basic example, and there are many more components and configurations that can be included in an LTI implementation. The IMS Global website has more resources and documentation for developing LTI applications.