AICC content delivery

AICC Communication Basics

- AICC standards are used to enable communication between an LMS and content hosted on different servers, allowing for tracking learner progress, scores, and completion.
- AICC courses rely on HTTP-based messaging between the course content and the LMS. Common AICC commands include `GetParam`, `PutParam`, `PutInteractions`, and `ExitAU`.

Setting Up AICC Course Files

- **AICC Descriptor Files**: These are .au, .crs, .cst, des, and .cst files that define course structure and properties. They allow the LMS to recognize and launch AICC-compliant content:
  - `.crs` the main course descriptor and should include essential information like the course ID, title, and version.
  - `.au` file lists the assignable units, linking them to specific URLs (e.g., the location of HTML files containing the course material).
  - `.cst` file: Lists course structure and completion rules.
  - `.des` file defines the components (Assignable Units) of the course and how they link together. In the example below we have just one unit for simplicity.

Sample .crs file

```
[Course]
Course_ID=SampleCourse01
Course_Title=Introduction to Sample AICC Course
Version=1.0
System_Vendor=Sample Company
Course_Creator=Sample Author
Max_Score=100
Mastery_Score=80
Time_Limit=00:20:00
System_ID=sample_course_system
Description=A sample AICC course for demonstration purposes.
```

Note: mastery_score is the minimum score required to pass

Sample .au file

```
File_Name=http://www.sampledomain.com/courses/sample_course/introduction.html
AU_Type=Web-based
Description=Introduction to the sample course.
Max_Score=100
Mastery_Score=80
System_Vendor=Sample Company
```

Sample .cst file

```
[Course Structure]
Course_ID=SampleCourse01
Structure=Module01,Module02

[Module01]
Title=Introduction Module
Description=This is the first module, providing an overview of AICC.
AU=IntroAU01
Mandatory=Yes
Prerequisite=None
Completion=All

[Module02]
Title=Advanced Concepts
Description=This module covers advanced topics in AICC.
AU=AdvancedAU01
Mandatory=Yes
Prerequisite=Module01
Completion=All
```

Sample .des file

```
[Course Structure]
Course_ID=SampleCourse01
Units=IntroModule01

[IntroModule01]
AU=IntroAU01
AU_Title=Introduction Module
Description=Basic introduction module with a quiz.
Mandatory=Yes
```

JavaScript Functions for AICC LMS Communication

- AICC-compliant courses often use JavaScript to communicate with the LMS. The core functions for AICC communication are `AICC_GetParam` and `AICC_PutParam`.
- **Setup LMS Communication**:
  - Capture query parameters from the LMS in the URL (like `AICC_URL` and `AICC_SID`) to facilitate communication.
  - Use JavaScript to send HTTP POST or GET requests to the LMS's AICC URL.
- **Basic JavaScript Structure**:

```
// Example: Setting up AICC session
let aiccURL = new URL(window.location.href).searchParams.get("AICC_URL");
let aiccSID = new URL(window.location.href).searchParams.get("AICC_SID");

// Function to GET data from LMS
function getAICCData() {
    fetch(`${aiccURL}`, {
        method: 'POST',
        headers: { 'Content-Type': 'application/x-www-form-urlencoded' },
        body: `command=GetParam&AICC_SID=${aiccSID}`
    })
    .then(response => response.text())
    .then(data => console.log("AICC GetParam Data:", data));
}

// Function to PUT data back to LMS
function putAICCData(score, status) {
    fetch(`${aiccURL}`, {
        method: 'POST',
        headers: { 'Content-Type': 'application/x-www-form-urlencoded' },
        body: `command=PutParam&AICC_SID=${aiccSID}&score=${score}&status=${status}`
    })
    .then(response => response.text())
    .then(data => console.log("AICC PutParam Data:", data));
}
```

Implementing AICC Functions

- **Get and Set Course Data**: Use `getAICCData()` to retrieve data about learner progress and `putAICCData()` to submit data back to the LMS.
- **Track User Progress**: AICC allows tracking of learner status, score, and interactions. Use the LMS’s data storage to keep track of these metrics.

- **Handling Completion and Exiting**: Set up an exit function that ensures completion status is sent to the LMS.

Open-Source Libraries for Simplification

To avoid building from scratch, consider using existing libraries, like ajaicc or sample-aicc-package, which provide functions to handle AICC communication through JavaScriptThese libraries simplify managing AICC requests and allow you to focus more on course content.

Working with AICC using PHP

Displaying an AICC course in PHP involves setting up a mechanism to handle AICC HACP (HTTP AICC Communication Protocol) communication. This protocol enables communication between the course content and an LMS. Here’s a basic example in PHP to help you set up an AICC-compliant display and tracking solution.

Here's what the process generally involves:

1. **Prepare AICC Configuration and Initialization**: Define parameters for launching and tracking.
2. **Handle AICC Requests**: Use PHP to parse and respond to AICC HACP requests.
3. **Launch the Course Content**: Embed or redirect to the actual course content.

Here’s some example PHP code that provides a simple implementation:

1. Setup an AICC Configuration File (aicc.php)

This PHP code example will set up basic handling for AICC data requests, simulate launching the course, and return status.

aicc.php

```
<?php
// aicc.php - AICC launch and tracking handler


// Configuration for AICC (usually stored in a database or configuration file)
$aicc_data = [
    'course_id' => '12345',
    'student_id' => 'student1',
    'lesson_location' => '',
    'lesson_status' => 'incomplete',
    'score' => '0',
];

// Process incoming AICC requests
if (isset($_POST['command'])) {
    $command = $_POST['command'];
    switch ($command) {
        case 'GetParam':
            // Send course data
            echo get_aicc_params($aicc_data);
            break;
        case 'PutParam':
            // Receive updates from course and store them
            update_aicc_data($aicc_data);
            echo "error=0\r\nerror_text=No Error";
            break;
        case 'GetParamEnc':
            // Encrypted command - not implemented here
            echo "error=1\r\nerror_text=Not Implemented";
            break;
        default:
            echo "error=1\r\nerror_text=Unknown Command";
    }
}

// Function to simulate returning AICC data in format for GetParam command
function get_aicc_params($data) {
    return "course_id={$data['course_id']}\r\n" .
           "student_id={$data['student_id']}\r\n" .
           "lesson_location={$data['lesson_location']}\r\n" .
           "lesson_status={$data['lesson_status']}\r\n" .
           "score={$data['score']}\r\n";
}

// Function to update AICC data based on PutParam command (simulates tracking)
function update_aicc_data(&$data) {
    $params = $_POST['aicc_data'];
    parse_str($params, $parsed_data);

    // Update simulated AICC data (this would be saved in a database in production)
    if (isset($parsed_data['lesson_location'])) {
        $data['lesson_location'] = $parsed_data['lesson_location'];
    }
    if (isset($parsed_data['lesson_status'])) {
        $data['lesson_status'] = $parsed_data['lesson_status'];
    }
    if (isset($parsed_data['score'])) {
        $data['score'] = $parsed_data['score'];
    }
}
```

2. Course Launch Page

Create a separate PHP page to handle the course launch. This page will redirect the user to the course content, and when the course calls back to `aicc.php`, it will handle status updates.

launch_course.php

```
<?php
// launch_course.php - Redirect to AICC-compliant course content

// Prepare AICC launch URL with required parameters
$aicc_url = "aicc.php";
$course_url = "path/to/your/course/content";

// Launch course and pass AICC data URL for communication
echo "<html><body>";
echo "<h1>Launching Course...</h1>";
echo "<iframe src='{$course_url}?aicc_url={$aicc_url}' width='100%' height='600px'></iframe>";
echo "</body></html>";
```

3. Course Content HTML (Simulated)

If you don’t have an AICC course package, you can simulate the content with a basic HTML page that communicates with the `aicc.php` script.

course_content.html (Example Course Content)

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <title>Sample AICC Course</title>
    <script>
        function sendAiccData(command, data) {
            const xhr = new XMLHttpRequest();
            xhr.open("POST", "<?php echo $_GET['aicc_url']; ?>", true);
            xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");

            xhr.onreadystatechange = function() {
                if (xhr.readyState === 4 && xhr.status === 200) {
                    console.log("AICC Response: " + xhr.responseText);
                }
            };

            let params = "command=" + command;
            if (data) {
                params += "&aicc_data=" + encodeURIComponent(data);
            }

            xhr.send(params);
        }

        // Simulate completing the course and reporting progress
        function completeCourse() {
            let aiccData = "lesson_location=Module1&lesson_status=completed&score=100";
            sendAiccData("PutParam", aiccData);
            alert("Course Completed!");
        }
    </script>
</head>
<body>
    <h1>Welcome to the AICC Course</h1>
    <p>Content goes here...</p>
    <button onclick="completeCourse()">Complete Course</button>
</body>
</html>
```

Example of an AICC quiz:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>AICC Quiz</title>
    <style>
        body { font-family: Arial, sans-serif; margin: 20px; }
        .question { margin-bottom: 15px; }
        .question p { font-weight: bold; }
        .feedback { margin-top: 20px; font-size: 1.2em; color: green; }
    </style>
</head>
<body>
    <h1>Army Leadership Fundamentals Quiz</h1>

    <!-- Question 1 -->
    <div class="question">
        <p>1. What are the three core attributes of an Army leader as defined in the Army Leadership
Requirements Model?</p>
        <input type="radio" name="q1" value="A"> A) Responsibility, Respect, Resilience<br>
        <input type="radio" name="q1" value="B"> B) Character, Presence, Intellect<br>
        <input type="radio" name="q1" value="C"> C) Courage, Confidence, Collaboration<br>
        <input type="radio" name="q1" value="D"> D) Competence, Integrity, Selflessness<br>
    </div>

    <!-- Question 2 -->
    <div class="question">
        <p>2. Which of the following is NOT one of the Army Values?</p>
        <input type="radio" name="q2" value="A"> A) Loyalty<br>
        <input type="radio" name="q2" value="B"> B) Duty<br>
        <input type="radio" name="q2" value="C"> C) Responsibility<br>
        <input type="radio" name="q2" value="D"> D) Honor<br>
    </div>

    <!-- Question 3 -->
    <div class="question">
        <p>3. Which leadership style involves the leader giving orders and expecting them to be followed without
question?</p>
        <input type="radio" name="q3" value="A"> A) Transformational<br>
    <input type="radio" name="q3" value="B"> B) Autocratic<br>
    <input type="radio" name="q3" value="C"> C) Participative<br>
    <input type="radio" name="q3" value="D"> D) Delegative<br>
</div>

<!-- Add more questions as needed -->

<button onclick="submitQuiz()">Submit Quiz</button>

<div id="feedback" class="feedback"></div>

<script>
    // Correct answers for each question
    const correctAnswers = {
        q1: "B",
        q2: "C",
        q3: "B",
    };

    function submitQuiz() {
        let score = 0;
        let totalQuestions = Object.keys(correctAnswers).length;

        // Loop through each question and check answers
        for (let question in correctAnswers) {
            const answer = document.querySelector(`input[name="${question}"]:checked`);
            if (answer && answer.value === correctAnswers[question]) {
                score++;
            }
        }

        // Calculate score percentage
        let scorePercentage = (score / totalQuestions) * 100;

        // assume we have loaded a js library to connect to the aicc api

          aiccApi.LMSInitialize("");
          aiccApi.SetValue("cmi.core.score.raw", scorePercentage.toFixed(2)); // Set raw score
          aiccApi.SetValue("cmi.core.score.min", "0"); // Minimum score
          aiccApi.SetValue("cmi.core.score.max", "100"); // Maximum score
```

```
            if (scorePercentage >= 70) { // Passing score
                aiccApi.SetValue("cmi.core.lesson_status", "passed");
                feedbackElement.innerHTML += "<br>Good job! You passed.";
            } else {
                aiccApi.SetValue("cmi.core.lesson_status", "failed");
                feedbackElement.innerHTML += "<br>Keep studying! Try again to improve your score.";
            }
            aiccApi.Finish("");

        }
    </script>
</body>
</html>
```

Summary of AICC Workflow:

1. **Launch Page** (`launch_course.php`): Redirects or embeds the course content.
2. **Course Content** (`course_content.html`): Communicates progress to `aicc.php` by calling `sendAiccData`.
3. **AICC Handler** (`aicc.php`): Receives, stores, and responds to progress and tracking information.

Important Notes:

- In production, AICC data should be stored in a secure database rather than using static PHP variables.
- Proper error handling should be added for a robust implementation.
- Customize `course_content.html` for real AICC content to interact with your LMS.

This setup provides a simple AICC-compatible structure for course tracking and display with PHP.

Revision #11
Created 30 October 2024 02:39:08 by peterd
Updated 1 November 2024 02:16:38 by peterd