Web-technológia II.

XML feldolgozás: expat, DOM, SimpleXML

Hatwágner F. Miklós

Széchenyi István Egyetem, Győr

2019. április 11.

XML feldolgozók

Megközelítések:

- eseményvezérelt: expat könyvtáron alapul
  - XML elemek → eseménykezelő fv. hívás
  - SAX (Simple API for XML) konvenciók
  - memóriatakarékos, de bonyolult kód
- fa alapú:
  - az egész dokumentum a memóriába kerül → gyors, egyszerű kód, de memóriaigényes
    - DOM
    - SimpleXML

expat

- Elemző létrehozása:
  `xml_parser_create()`
  Bemeneti kódolás: automatikus felismerés, kimeneti: paraméterként megadható, aé.: UTF-8. Erőforrást ad vissza (nincs type hinting)
- Eseménykezelők regisztrálása:
  `xml_set_element_handler()`
  Paraméterek:

parser	elemző erőf.
start_element	nyitó elem eseménykezelője
end_element	záró elem eseménykezelője

- Nyitó elem eseménykezelője:
  startElementHandler()
  - parser elemző erőf.
  - name elem neve, alapértelmezetten NAGYBETŰSRE alakítva
  - attribs attribútumok asszociatív tömbben
- Záró elem eseménykezelője:
  endElementHandler()
  - parser elemző erőf.
  - name elem neve

- Karakteres adatok (CDATA) eseménykezelőjének regisztrálása:
  xml_set_character_data_handler()
  - parser elemző erőf.
  - handler eseménykezelő fv.
- Eseménykezelő alakja:
  handler()
  - parser elemző erőf.
  - data adatok string-ként. Lehet üres string, de az is lehet, hogy egy node teljes szövege csak többszöri fv. hívással érhető el.

expat

- Feldolgozási utasítások (Processing Instruction, PI) elhelyezése XML-ben:
  <? *utasítás;* ?>
  Pl. egy PHP kód elhelyezésére:
  <?php echo 5*2; ?>
- Eseménykezelő regisztrálása:
  xml_set_processing_instruction_handler()
  - parser elemző erőf.
  - handler eseménykezelő fv.
- Eseménykezelő alakja:
  handler()
  - parser elemző erőf.
  - target PI célja (pl. PHP)
  - data a végrehajtandó utasítások. Kiértékelhető lenne pl. eval() fv.-nyel, de veszélyes!

expat

- Entitások: a legtöbb automatikusan feloldásra kerül,
- kivéve: *külső* és *nem értelmezett* $\rightarrow$ eseménykezelő fv.
- Külső entitás eseménykezelőjének regisztrálása:
  xml_set_external_entity_ref_handler()

parser	elemző erőf.
handler	eseménykezelő fv.

- Eseménykezelő alakja:
  handler()

parser	elemző erőf.
openEntityNames	nyitott entitások szóközzel tagolt listája, beleértve a hivatkozott entitást
base	a külső entitás rendszer-azonosítójának (systemID) feloldásához használt bázis; mindig üres string
systemID	rendszer azonosító, ált. fájlnév
publicID	nyilvános azonosító, ha van ilyen, különben üres string

- Ha vt. érték nincs vagy FALSE, az elemzés XML_ERROR_EXTERNAL_ENTITY_HANDLING hibával leáll.

- Nem értelmezett entitásokat jelölés deklaráció kíséri, pl.:

Külső entitás jelöléssel

```
<!DOCTYPE doc [
  <!NOTATION jpeg SYSTEM "image/jpeg">
  <!ENTITY logo SYSTEM "sze-logo.jpg" NDATA jpeg>
]>
```

- Jelölés deklaráció eseménykezelőjének regisztrálása:
  xml_set_notation_decl_handler()

parser	elemző erőf.
handler	eseménykezelő fv.

- Eseménykezelő alakja:
  handler()

parser	elemző erőf.
notationName	jelölésnév
base	üres string
systemID	rendszer azonosító
publicID	nyilvános azonosító

- Nem értelmezett entitás kezelőjének regisztrálása:
  `xml_set_unparsed_entity_decl_handler()`

parser	elemző erőf.
handler	eseménykezelő fv.

- Eseménykezelő alakja:
  `handler()`

parser	elemző erőf.
entity	entitás neve
base	üres string
systemID	rendszer azonosító
publicID	nyilvános azonosító
notation	az entitáshoz kapcsolt jelölésdeklarációra hivatkozik

- Alapértelmezett eseménykezelő: minden egyéb esetben (XML deklaráció, DTD) erre kerül a végrehajtás. Regisztrálása:
  xml_set_default_handler()
  - parser elemző erőf.
  - handler eseménykezelő fv.
- Eseménykezelő alakja:
  handler()
  - parser elemző erőf.
  - text szöveges adat
- Ha az eseménykezelő fv.-ek metódusok → objektum regisztrálása:
  xml_set_object()
  - parser elemző erőf., ezt is illik expliciten felszabadítani unset-tel
  - object az eseménykezelőt tartalmazó objektum

expat

- Az elemző beállításainak lekérdezése:
  xml_parser_get_option()
  parser elemző erőf.
  option beállítás azonosítója
  Vt. érték: beállítás értéke
- Beállítások módosítása:
  xml_parser_set_option()
  parser elemző erőf.
  option beállítás azonosítója
  value beállítás értéke
- Fontosabb beállítás azonosítók:
  XML_OPTION_CASE_FOLDING elem- és attribútumnevek nagybetűsre alakítása
  XML_OPTION_TARGET_ENCODING kimenet karakterkódolása, alapesetben egyezik a bemenetével

expat

- Beállítások megtörténtek, eseménykezelők regisztrálva → elemzés megkezdése:
  xml_parse()
  parser elemző erőf.
  data adatcsomag; a fv. többször egymás után hívható egy fájl egymást követő darabjaival
  [final] ez az utolsó adatcsomag?
  vt. érték: elemzés sikeres?
- Hibakezelés: hibakód lekérdezése:
  xml_get_error_code()
  parser elemző erőf.
  Vt. érték: hibakód, ha minden OK: XML_ERROR_NONE
- Hibaüzenet lekérdezése a kód alapján:
  xml_error_string()
  code hibakód

generateXML.php

```
<?php
  header("Content-Type: application/xml");

  // XML deklaráció
  $xml = "<?xml version=\"1.0\" encoding=\"UTF-8\" ?>\n";
  // Feldolgozási utasítás, Processing Instruction
  $xml .= "<?php echo 5*2; ?>\n";

  // receptek tömbjének összeállítása
  $receptek = array(
    array(
      "nev" => "Palacsinta",
      "hozzavalok" => array(
        "125 g liszt",
        "3,5 dl tej",
        "2 tojás"
      ),
      "elkeszites" => "Egy tálba lisztet szitálunk, majd beleütjük ".
        "a tojásokat..."
    ),
```

generateXML.php

```
    array(
      "nev" => "Mustáros házinyúl",
      "hozzavalok" => array(
        "1,5 kg nyúlhús",
        "4 evőkanál olaj",
        "2 dl tejföl"
      ),
      "elkeszites" => "A nyulat nagyobb darabokra vágjuk, és sóval, ".
        "mustárral alaposan bedörzsöljük..."
    )
);

// a tömb alapján xml előállítása
$xml .= "<szakacskonyv encoding=\"UTF-8\">\n";
foreach($receptek as $id => $recept) {
  $xml .= "\t<recept id=\"$id\">\n".
          "\t\t<nev>{$recept['nev']}</nev>\n".
          "\t\t<hozzavalok>\n";
  foreach ($recept['hozzavalok'] as $hozzavalo) {
    $xml .= "\t\t\t<hozzavalo>$hozzavalo</hozzavalo>\n";
  }
```

generateXML.php

```
    $xml .= "\t\t</hozzavalok>\n".
            "\t\t<elkeszites>{$recept['elkeszites']}</elkeszites>\n".
            "\t</recept>\n";
  }
  $xml .= "</szakacskonyv>\n";

  // eredmény megjelenítése, fájlba írása
  echo $xml;
  file_put_contents("/home/feltoltes/www/szakacskonyv.xml", $xml);
?>
```

parseXML.php

```
<?php declare(strict_types = 1); ?>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Szakácskönyv</title>
  </head>
  <body>
    <?php
    class Szakacskonyv {
```

parseXML.php

```
const MERET = 4192;

private $elemzo;    // XML elemző, parser
private $fh;        // File handle
private $bejegyzes; // Aktuális recept adatai
private $verem;     // Egymásba ágyazott XML elemek
private $id;        // Keresett recept azonosítója
private $cdata;     // Szöveges adatok összevonása

public function __construct(string $fajl) {
  $this->elemzo = xml_parser_create();
  xml_set_object($this->elemzo, $this);
  $this->fh = fopen($fajl, "r");
}

public function __destruct() {
  fclose($this->fh);
  xml_parser_free($this->elemzo);
}
```

parseXML.php

```
private function feldolgozas() {
  $this->bejegyzes = array();
  $this->verem = array();
  while ($adat = fread($this->fh, self::MERET)) {
    if (!xml_parse($this->elemzo, $adat, feof($this->fh))) {
      echo "Elemzési hiba: ".xml_error_string($this->elemzo).
           " Sor: ".xml_get_current_line_number($this->elemzo);
    }
  }
}

private function kezdoElem(string $elem, array &$attributumok) {
  array_push($this->verem, $elem);
  if(array_key_exists("ID", $attributumok)) {
    $this->bejegyzes["ID"] = $attributumok["ID"];
  }
  if(array_key_exists("ENCODING", $attributumok)) {
    xml_parser_set_option($this->elemzo,
      XML_OPTION_TARGET_ENCODING, $attributumok["ENCODING"]);
  }
}
```

parseXML.php

```
private function zaroElem(string $elem) {
  array_pop($this->verem);
  if($elem == "RECEPT") {
    $this->bejegyzes = array();
  }
}

public function showTartalom() {
  // Rossz ötlet, ki ne próbáld!
  // xml_set_processing_instruction_handler($this->elemzo, "pi");
  xml_set_element_handler($this->elemzo, "tartKezdoElem",
    "tartZaroElem");
  xml_set_character_data_handler($this->elemzo, "tartCdata");
  echo "<h1>Tartalomjegyzék</h1>\n\t\t<ul>\n";
  $this->feldolgozas();
  echo "\t\t</ul>\n";
}
```

parseXML.php

```
public function pi($elemzo, $cel, $adat) {
  if($cel == "php") {
    echo "<h2>Vajon mi ";
    eval($adat);
    echo "?</h2>\n";
  }
}

public function tartKezdoElem($elemzo,
  string $elem, array &$attributumok) {
  $this->kezdoElem($elem, $attributumok);
}

public function tartZaroElem($elemzo, string $elem) {
  if($elem == "RECEPT") {
    echo "\t\t\t<li><a href=\"".$_SERVER["PHP_SELF"].
         "?id=".$this->bejegyzes["ID"].
         "\">".$this->bejegyzes["NEV"]."</a></li>\n";
  }
  $this->zaroElem($elem);
}
```

parseXML.php

```
public function tartCdata($elemzo, string $szoveg) {
  if(end($this->verem) == "NEV") {
    if(array_key_exists("NEV", $this->bejegyzes)) {
      $this->bejegyzes["NEV"] .= $szoveg;
    } else {
      $this->bejegyzes["NEV"] = $szoveg;
    }
  }
}

public function showRecept(string $id) {
  $this->id = $id;
  xml_set_element_handler($this->elemzo, "receptKezdoElem",
    "receptZaroElem");
  xml_set_character_data_handler($this->elemzo, "receptCdata");
  $this->feldolgozas();
}
```

parseXML.php

```
public function receptKezdoElem($elemzo, string $elem,
  array &$attributumok) {
  $this->kezdoElem($elem, $attributumok);
  $this->cdata = false;
  if($elem == "HOZZAVALOK" &&
     $this->bejegyzes["ID"] == $this->id) {
    $this->bejegyzes["HOZZAVALOK"] = array(); } }
public function receptZaroElem($elemzo, string $elem) {
  $this->cdata = false;
  if($elem == "RECEPT" && $this->bejegyzes["ID"] == $this->id) {
    echo "\t\t<h1>".$this->bejegyzes["NEV"]."</h1>\n".
         "\t\t<h2>Hozzávalók:</h2>\n\t\t<ul>\n";
    foreach ($this->bejegyzes["HOZZAVALOK"] as $hv) {
      echo "\t\t\t<li>$hv</li\n>";
    }
    echo "\t\t</ul>\n\t\t<h2>Elkészítés:</h2>\n".
         "\t\t<p>".$this->bejegyzes["ELKESZITES"]."</p>\n".
         "\t\t<p><a href=\"".$_SERVER["PHP_SELF"]."\">".
         "Vissza a tartalomjegyzékhez</a></p>\n";
  }
  $this->zaroElem($elem); }
```

parseXML.php

```
public function receptCdata($elemzo, string $szoveg) {
  if(array_key_exists("ID", $this->bejegyzes) &&
     $this->bejegyzes["ID"] == $this->id) {
    $utolso = end($this->verem);
    if($utolso == "NEV" || $utolso == "ELKESZITES") {
      if(array_key_exists($utolso, $this->bejegyzes)) {
        $this->bejegyzes[$utolso] .= $szoveg;
      } else {
        $this->bejegyzes[$utolso] = $szoveg;
      }
    } elseif($utolso == "HOZZAVALO") {
      if($this->cdata) {
        $hv = array_pop($this->bejegyzes["HOZZAVALOK"]);
        array_push($this->bejegyzes["HOZZAVALOK"],
                $hv.$szoveg);
      } else {
        $this->bejegyzes["HOZZAVALOK"][] = $szoveg;
      }
    }
  }
  $this->cdata = true; } }
```

parseXML.php

```
    $horvathRozi = new Szakacskonyv(
      "/home/feltoltes/www/szakacskonyv.xml");
    if(isset($_GET["id"])) {
      $horvathRozi->showRecept($_GET["id"]);
    } else {
      $horvathRozi->showTartalom();
    }
    ?>
  </body>
</html>
```

DOM referencia

Néhány fontosabb osztály:

- DOMDocument a teljes XML fát ez reprezentálja (gyökér elemet külön kell létrehozni)
- DOMNode egy általános csúcs a fában, többnyire a leszármazottjaival dolgozunk (pl. DOMDocument, DOMElement)
- DOMNodeList DOMElement-ek bejárható listája (*Traversable* interfész)
- DOMElement egy XML elemet reprezentál
- DOMAttr egy attribútumot reprezentál

DOMDocument néhány fontosabb metódusa

- Konstruktor opcionális paraméterei: verziószám, karakterkódolás
- PI csúcs létrehozása: createProcessingInstruction()
- Elem létrehozása: createElement()
- Attribútum létrehozása: createAttribute()
- Elem keresése ID alapján: getElementById(), de csak a DTD/XSD tudja megmondani, mi az ID attribútum neve! → DOMElement::setIdAttribute()
- Elemek keresése *name* attribútum alapján: getElementsByTagName()
- Betöltés: load()
- Mentés: save()
- XML szövegként: saveXML()

DOMElement néhány fontosabb metódusa

- Konstruktor paraméterei: név, opcionálisan érték
- Attribútum értékének lekérdezése: getAttribute(), beállítása: setAttribute()
- Tartalmazó DOMDocument elérése: ownerDocument adattaggal

DOMNode néhány fontosabb metódusa

- Gyerek hozzáadása: appendChild()
- Gyerek beszúrása: insertBefore(), paraméterek: mit, mi elé
- Gyerek eltávolítása: removeChild()
- Gyerekek elérése: $e->childNodes->item($index)

generateDOM.php

```
<?php
  header("Content-Type: application/xml");

  // XML dokumentum létrehozása
  $xml = new DOMDocument("1.0", "UTF-8");
  $xml->xmlStandalone = true;
  // Feldolgozási utasítás, Processing Instruction
  $pi = $xml->createProcessingInstruction("php", "echo 5*2;");
  if($pi) {
    $xml->appendChild($pi);
  }
```

generateDOM.php

```
$receptek = array(
  array(
    "nev" => "Palacsinta",
    "hozzavalok" => array(
      "125 g liszt",
      "3,5 dl tej",
      "2 tojás"
    ),
    "elkeszites" => "Egy tálba lisztet szitálunk, majd beleütjük ".
      "a tojásokat..."
  ),
  array(
    "nev" => "Mustáros házinyúl",
    "hozzavalok" => array(
      "1,5 kg nyúlhús",
      "4 evőkanál olaj",
      "2 dl tejföl"
    ),
    "elkeszites" => "A nyulat nagyobb darabokra vágjuk, és sóval, ".
      "mustárral alaposan bedörzsöljük..."
  ) );
```

generateDOM.php

```
// a tömb alapján xml előállítása
$szakacskonyv = $xml->createElement("szakacskonyv");
foreach($receptek as $id => $r) {
  $recept = $xml->createElement("recept");
  $recept->setAttribute("id", $id);
  $nev = $xml->createElement("nev", $r['nev']);
  $recept->appendChild($nev);
  $hozzavalok = $xml->createElement("hozzavalok");
  foreach ($r['hozzavalok'] as $h) {
    $hozzavalo = $xml->createElement("hozzavalo", $h);
    $hozzavalok->appendChild($hozzavalo);
  }
  $recept->appendChild($hozzavalok);
  $elkeszites = $xml->createElement("elkeszites", $r['elkeszites']);
  $recept->appendChild($elkeszites);
  $szakacskonyv->appendChild($recept);
}
$xml->appendChild($szakacskonyv);
// eredmény megjelenítése, fájlba írása
echo $xml->saveXML();
$xml->save("/home/feltoltes/www/szakacskonyv.xml"); ?>
```

parseDOM.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Szakácskönyv</title>
  </head>
  <body>
    <?php
    class Szakacskonyv {
      private $konyv;

      public function __construct($fajl) {
        $this->konyv = new DOMDocument();
        $this->konyv->load($fajl);
      }
```

parseDOM.php

```
public function showTartalom() {
  echo "<h1>Tartalomjegyzék</h1>\n\t\t<ul>\n";
  foreach($this->konyv->getElementsByTagName("recept")
    as $recept) {
    echo "\t\t\t<li><a href=\"".$_SERVER["PHP_SELF"].
         "?id=".$recept->getAttribute("id").
         "\">".$recept->getElementsByTagName("nev")->
           item(0)->nodeValue.
         "</a></li>\n";
  }
  echo "\t\t</ul>\n";
}
```

parseDOM.php

```
public function showRecept($id) {
  $osszes = $this->konyv->getElementsByTagName("recept");
  // DTD hiánya miatt
  foreach($osszes as $recept) {
    $recept->setIdAttribute("id", true);
  }
  $recept = $this->konyv->getElementById($id);
  echo "\t\t<h1>".$recept->getElementsByTagName("nev")->
    item(0)->nodeValue."</h1>\n";
  echo "\t\t<h2>Hozzávalók:</h2>\n\t\t<ul>\n";
  $lista = $recept->getElementsByTagName("hozzavalo");
  foreach($lista as $hv) {
    echo "\t\t\t<li>$hv->nodeValue</li\n>";
  }
  echo "\t\t</ul>\n";
  echo "\t\t<h2>Elkészítés:</h2>\n";
  echo "\t\t<p>".$recept->getElementsByTagName("elkeszites")->
       item(0)->nodeValue."</p>\n";
  echo "\t\t<p><a href=\"".$_SERVER["PHP_SELF"]."\">".
       "Vissza a tartalomjegyzékhez</a></p>\n";
} }
```

parseDOM.php

```
  $horvathRozi = new Szakacskonyv(
    "/home/feltoltes/www/szakacskonyv.xml");
  if(isset($_GET["id"])) {
    $horvathRozi->showRecept($_GET["id"]);
  } else {
    $horvathRozi->showTartalom();
  }
  ?>
  </body>
</html>
```

SimpleXML

SimpleXML: elsősorban ismert szerkezetű fájlok feldolgozásához, de új XML is készíthető vele

- Konstruktor
- Létező fájl betöltése:
  $xml = new SimpleXMLElement($location, null, true);
  $xml = simplexml_load_file($location);
- XML string elemzése:
  $xml = new SimpleXMLElement($xmlAsString);
  $xml = simplexml_load_string($xmlAsString);
- Új fájl készítés:
  $xml = new SimpleXMLElement('<root/>');

SimpleXMLElement legfontosabb metódusai:

- Feldolgozási utasítást nem kezeli, de SimpleXMLElement → DOMElement: dom_import_simplexml()
- Gyerek hozzáadása: addChild(), paraméterek: név, opcionális érték, vissza: gyerek csúcs
- Attribútum hozzáadása: addAttribute(), paraméterek: név, opcionális érték
- Lekérés stringként: asXML()

generateSimpleXML.php

```php
<?php
  header("Content-Type: application/xml");

  // XML dokumentum létrehozása
  $xml = new SimpleXMLElement(
    '<?xml version="1.0" encoding="UTF-8" standalone="yes"?>'.
    '<szakacskonyv/>');

  // Feldolgozási utasítás, Processing Instruction
  $domDoc = dom_import_simplexml($xml)->ownerDocument;
  $pi = $domDoc->createProcessingInstruction("php", "echo 5*2;");
  $domDoc->insertBefore($pi, $domDoc->childNodes->item(0));

  // receptek tömbjének összeállítása
  $receptek = array( //...
```

generateSimpleXML.php

```
  foreach($receptek as $id => $r) {
    $recept = $xml->addChild("recept");
    $recept->addAttribute("id", $id);
    $recept->addChild("nev", $r['nev']);
    $hozzavalok = $recept->addChild("hozzavalok");
    foreach ($r["hozzavalok"] as $h) {
      $hozzavalok->addChild("hozzavalo", $h);
    }
    $recept->addChild("elkeszites", $r["elkeszites"]);
  }

  // eredmény megjelenítése, fájlba írása
  echo $xml->asXML();
  file_put_contents(
    "/home/feltoltes/www/szakacskonyv.xml",
    $xml->asXML());
?>
```

Gyerek elemek elérése:

- adattagok az elemével egyező névvel (*Traversable*)
- Pl. echo $xml->childElements[0]->childElement; // tartalom
- children()

Attribútumok lekérése:

- mintha tömbelemek lennének
- Pl. echo $xml['attribute'];
- attributes()

Elem neve: getName()

parseSimpleXML.php

```
<html>
  <head>
    <meta charset="utf-8" />
    <title>Szakácskönyv</title>
  </head>
  <body>
    <?php
    class Szakacskonyv {
      private $konyv;
      public function __construct($fajl) {
        $this->konyv = new SimpleXMLElement($fajl, 0, true);
      }
      public function showTartalom() {
        echo "<h1>Tartalomjegyzék</h1>\n\t\t<ul>\n";
        foreach($this->konyv->recept as $recept) {
          echo "\t\t\t<li><a href=\"".$_SERVER["PHP_SELF"].
               "?id=".$recept["id"].
               "\">".$recept->nev."</a></li>\n";
        }
        echo "\t\t</ul>\n";
      }
```

parseSimpleXML.php

```
public function showRecept($id) {
  foreach($this->konyv->recept as $recept) {
    if($recept["id"] == $id) {
      // XPath használatával közvetlenül is kiválasztható lenne
      // $recept = $this->konyv->xpath(
      //   "/szakacskonyv/recept[@id=\"$id\"]");
      // $recept = $recept[0];
      echo "\t\t<h1>".$recept->nev."</h1>\n";
      echo "\t\t<h2>Hozzávalók:</h2>\n";
      echo "\t\t<ul>\n";
      foreach($recept->hozzavalok->hozzavalo as $hv) {
        echo "\t\t\t<li>$hv</li\n>";
      }
      echo "\t\t</ul>\n";
      echo "\t\t<h2>Elkészítés:</h2>\n";
      echo "\t\t<p>".$recept->elkeszites."</p>\n";
      echo "\t\t<p><a href=\"".$_SERVER["PHP_SELF"]."\">".
           "Vissza a tartalomjegyzékhez</a></p>\n";
      break;
    }
  } } }
```

parseSimpleXML.php

```
    $horvathRozi = new Szakacskonyv(
      "/home/feltoltes/www/szakacskonyv.xml");
    if(isset($_GET["id"])) {
      $horvathRozi->showRecept($_GET["id"]);
    } else {
      $horvathRozi->showTartalom();
    }
    ?>
  </body>
</html>
```

Feladat

Készítsen weboldalt egy rendezvény lehetséges időpontjának egyeztetéséhez! Az oldalnak tartalmaznia kell

- a rendezvény címét,
- rövid ismertetését,
- a lehetséges időpontokat, amelyek közül tetszőleges számú megjelölhető,
- a résztvevő nevének megadására szolgáló szövegbeviteli mezőt (kötelező kitölteni).

Az adatok megadását követően az adatokat gombnyomásra egy jól formázott (well-formed), de egyébként tetszőlegesen megválasztott szerkezetű XML fájlba kell menteni.
Megjelenítendő továbbá áttekinthető formában az is, hogy eddig ki milyen időpontokat jelölt meg számára megfelelőként.
Adjon lehetőséget a résztvevők által korábban megadott időpontok megváltoztatására, és a leadott szavazatainak teljes törlésére is (minden időpont törlését követően már a neve se látszódjék)!